NEC

Preliminary User’s Manual

V850E/Dx3

32-bit Single-Chip Microcontroller

Hardware

µPD70F3420, µPD703420
µPD70F3421, µPD703421
µPD70F3422, µPD703422
µPD70F3423
µPD70F3424
µPD70F3425
µPD70F3426
µPD70F3427

Document No. U17566EE1V2UM00

Date Published 18/7/06

Printed in Germany

Notes for CMOS Devices

1. **Precaution against ESD for semiconductors**
   Strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it once, when it has occurred. Environmental control must be adequate. When it is dry, humidifier should be used. It is recommended to avoid using insulators that easily build static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work bench and floor should be grounded. The operator should be grounded using wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with semiconductor devices on it.

2. **Handling of unused input pins for CMOS**
   No connection for CMOS device inputs can be cause of malfunction. If no connection is provided to the input pins, it is possible that an internal input level may be generated due to noise, etc., hence causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using a pull-up or pull-down circuitry. Each unused pin should be connected to VDD or GND with a resistor, if it is considered to have a possibility of being an output pin. All handling related to the unused pins must be judged device by device and related specifications governing the devices.

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

Legal Notes

- The information in this document is current as of 18/7/06. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC sales representative for availability and additional information.
- No part of this document may be copied or reproduced in any form or by any means without prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such NEC Electronics products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of customer's equipment shall be done under the full responsibility of customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: “Standard”, “Special” and “Specific”.

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

  "Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.
  "Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support).
  "Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

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

Note

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

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

3. SuperFlash® is a registered trademark of Silicon Storage Technology, Inc. in several countries including the United States and Japan. This product uses SuperFlash® technology licensed from Silicon Storage Technology, Inc.

Regional Information

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

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

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

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Europe) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 01
Fax: 0211-65 03 327

Sucursal en España
Madrid, Spain
Tel: 091- 504 27 87
Fax: 091- 504 28 60

Succursale Française
Vélizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

Filiale Italiana
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

Branch The Netherlands
Eindhoven, The Netherlands
Tel: 040-244 58 45
Fax: 040-244 45 80

Branch Sweden
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

United Kingdom Branch
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
Singapore
Tel: 65-6253-8311
Fax: 65-6250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

NEC do Brasil S.A.
Electron Devices Division
Guarulhos, Brasil
Tel: 55-11-6465-6810
Fax: 55-11-6465-6829

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.eu.necel.com.

Table of Contents

Chapter 3 CPU System Functions 103

Chapter 4 Clock Generator 129

Chapter 7 Bus and Memory Control (BCU, MEMC) 249

Chapter 8 DMA Controller (DMAC) 309

Chapter 15 Watchdog Timer (WDT) . . . 497

Chapter 16 Asynchronous Serial Interface (UARTA) . . . 507

Chapter 17 Clocked Serial Interface (CSIB) . . . 541

Chapter 18 I^{2}C Bus (IIC)

Chapter 19 CAN Controller (CAN) 639

Chapter 21 Stepper Motor Controller/Driver (Stepper-C/D) 795

Chapter 22 LCD Controller/Driver (LCD-C/D) 809

Chapter 23 LCD Bus Interface (LCD-I/F) 823

Chapter 24 Sound Generator (SG) 841

Chapter 1 Introduction

The V850E/Dx3 is a product line in NEC Electronics' V850 family of single-chip microcontrollers designed for automotive applications.

1.1 General

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

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

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

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

(1) V850E CPU

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

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

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

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

(2) On-chip flash memory

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

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

(3) A full range of software development tools

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

1.2 Features Summary

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

Table 1-1 V850E/Dx3 features summary (1/4)

CPU	
Core	V850E1
Number of instructions	81
Minimum instruction execution time	• 19.841 ns (@ ϕ = 50.4 MHz) (µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427) • 39.683 ns (@ ϕ = 25.2 MHz) (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423)
General registers	32 registers (32 bits each)
Instruction set	
V850E (compatible with V850 plus additional powerful instructions for reducing code and increasing execution speed)	
Signed multiplication (16 bits × 16 bits → 32 bits or 32 bits × 32 bits → 64 bits): 1 to 2 clocks	
Saturated operation instructions (with overflow/underflow detection) 32-bit shift instructions: 1 clock	
Bit manipulation instructions	
Load/store instructions with long/short format	
Signed load instructions	
Internal flash memory	
Size	• 2 MB (µPD70F3426) • 1 MB (µPD70F3427, µPD70F3425) • 512 KB (µPD70F3424, µPD70F3423) • 384 KB (µPD70F3422) • 256 KB (µPD70F3421) • 128KB (µPD70F3420)
Flash protection	• N-Wire security function • external programmer security function
Secure self programming	
Internal mask ROM memory	
Size	• 384 KB (µPD703422) • 256 KB (µPD703421) • 128 KB (µPD703420)

Table 1-1 V850E/Dx3 features summary (2/4)

Internal data RAM	
Size	• 84 KB (µPD70F3426) • 60 KB (µPD70F3427) • 32KB (µPD70F3425) • 24 KB (µPD70F3424) • 20 KB (µPD70F3423) • 16 KB (µPD70(F)3422) • 12 KB (µPD70(F)3421) • 6 KB (µPD70(F)3420)
Clock Generator	
Internal spread-spectrum PLL	• 48 MHz ± 5 % (µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427) • 24 MHz ± 5 % (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423)
Internal PLL (peripheral clock supply)	8-fold PLL
CPU frequency range	• up to 50.4 MHz (µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427) • up to 25.2 MHz (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423)
Peripheral frequency range	up to 16 MHz
Main crystal frequency range (main oscillator)	4 MHz
Sub oscillator	32 KHz (typ.)
Ring oscillator	240 KHz (typ.)
Clock supervision	2 channels: • main oscillator monitor • sub oscillator monitor
Auxiliary frequency output	
Built-in power saving modes	
HALT / IDLE / WATCH / Sub-WATCH / STOP	
External memory bus interface (µPD70F3427 only)	
Address/data separated busses	24/32-bit
Chip select signals	4
DMA Controller	
Number of channels	4
I/O ports	
Input/output ports	• µPD70F3427: 101 • all others: 98
Input ports	16

Table 1-1 V850E/Dx3 features summary (3/4)

A/D Converter	
Number of channels	• 16 (µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427) • 12 (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423)
Resolution	10-bit
Conversion modes	• Continuous select mode • Continuous scan mode • Timer trigger mode • Software trigger mode
Analog input channels shared with digital input port functionality	
Serial interfaces	
Synchroneous: CSI (CSIB)	• 3 channels (µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427) • 2 channels (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423)
Asynchroneous: UART (UARTA)	2 channels with LIN support
I^2C (IIC)	2 channels
CAN (CAN)	2 channels with 32 message buffers each
Timers	
16-bit multi purpose timer/event counter (TMP)	4 channels
16-bit multi purpose timer/counter (TMG)	3 channels
16-bit multi purpose timer/counter (TMZ)	• 10 channels (µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427) • 6 channels (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423)
Watch Timer (WT)	1 channel
Watch Calibration Timer (WCT)	1 channel
Watchdog Timer (WDT)	1 channel
LCD Controller/Driver (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423)	
Segment signal output	max. 40
Common signal output	max. 4
Modes	1/4 duty, 1/3 bias
LCD Bus Interface	
Bus width	8-bit parallel
Bus control modes	2 modes: • RD strobe and WR strobe (“mod80”) • RD/WR signal and data strobe (“mod68”)
Transfer speed	100 KHz to 3.2 MHz
Stepper Motor Controller/Driver	
Number of channels	6
Resolution	8-bit and 8-bit + 1

Table 1-1 V850E/Dx3 features summary (4/4)

Sound Generator	
Number of channels	1
Volume	9-bit volume level accuracy
Sound frequency	245 Hz to 6 KHz with min. resolution of ± 20 Hz
Sound duration	256 steps
Interrupts and exceptions	
Non-maskable interrupts	2 sources
Maskable interrupts	• 92 sources (µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427) • 84 sources (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423)
Software exceptions	32 sources
Exception trap	2 sources
ROM Correction	
Number of channels	8 channels by DBTRAP
On-chip debug interface	
Number of interfaces	1
Connection of an external N-Wire emulator	
Internal Voltage Comparators	
Number of channels	2
Power supply supervision	
Power-On-Clear	Generates reset at power-up and in case of power loss
Single supply operating voltage	
Range	4.0 V to 5.5 V (refer to Electrical Target Specification)
Temperature range	
Range	T_a = –40 to +85°C • @ ϕ = 48 MHz) (µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427) • @ ϕ = 24 MHz) (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423)
Package	
Package	• µPD70F3427: 208-pin QFP • all others: 144-pin QFP
Package size	• µPD70F3427: 28 mm × 28 mm • all others: 20 mm × 20 mm
Pin pitch	0.5 mm
CMOS technology	

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

1.3 Product Series Overview

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

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

Table 1-2 V850E/Dx3 product series overview (1/2)

Part number		µPD70F3427	µPD70F3426	µPD70F3425	µPD70F3424	µPD70F3423	µPD70F3422	µPD703422	µPD70F3421	µPD703421	µPD70F3420	µPD703420
Internal memory	Flash	1 MB	1 MB + 1 MB[a]	1 MB	512 KB		384 KB	none	256 KB	none	128 KB	none
	ROM	none						384 KB	none	256 KB	none	128 KB
	RAM	60 KB	60 KB + 24 KB[a]	32 KB	24 KB	20 KB	16 KB		12 KB		6 KB	
External memory interface		provided	–									
DMA		4 ch										
Operating clock	Main oscillator with SSCG[b]	48 MHz typ., 50.4 MHz max.[c]				24 MHz typ., 25.2 MHz max.[c]						
	Ring oscillator	240 KHZ typ.										
	Sub oscillator	32 KHZ typ.										
I/O ports	Input/Output	101	98									
	Input	16										
A/D converter		16 channels				12 channels						
Timers	TMZ	10 channels				6 channels						
	TMP	4 channels										
	TMG	3 channels										
	WDT	1 channel										
	Watch	provided										
	Watch calibration	provided										
Serial interfaces	CAN	2 channels										
	UARTA	2 channels										
	CSIB	3 channels				2 channels						
	I^2C	2 channels										

Table 1-2 V850E/Dx3 product series overview (2/2)

Part number		**µPD70F3427**	**µPD70F3426**	**µPD70F3425**	**µPD70F3424**	**µPD70F3423**	**µPD70F3422**	**µPD703422**	**µPD70F3421**	**µPD703421**	**µPD70F3420**	**µPD703420**
Interrupts	External	8 channels				7 channels						
	Internal	91 channels	87 channels		91 channels	83 channels	79 channels	75 channels	79 channels	75 channels		
	NMI	2 channels										
Other functions	ROM Correction by DBTRAP	8 channels	2 x 8 channels	8 channels								
	Power-On-Clear	provided										
	Voltage Comparator	2 channels										
	Clock supervision	2 channels										
	Sound Generator	1 channel										
	Stepper Motor Controller/Driver	6 channels										
	LCD-Controller/Driver	none				40 x 4						
	LCD Bus Interface	provided										
	Auxiliary frequency output	provided										
	On-Chip debug	provided										
Operating voltage		3.5 V to 5.5 V[c]										
Package		208-pin QFP	144-pin QFP									

[a] The additional 1 MB flash memory respectively 24KB RAM is accessible via the VSB, and thus requires an additional CPU clock cycle.

[b] SSCG: spread spectrum clock generator

[c] Refer to the Electrical Target Specification

1.4 Description

Figure 1-1 provides a functional block diagram of the V850E/DJ3 µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423, µPD70F3424, µPD70F3425, µPD70F3426 microcontrollers.

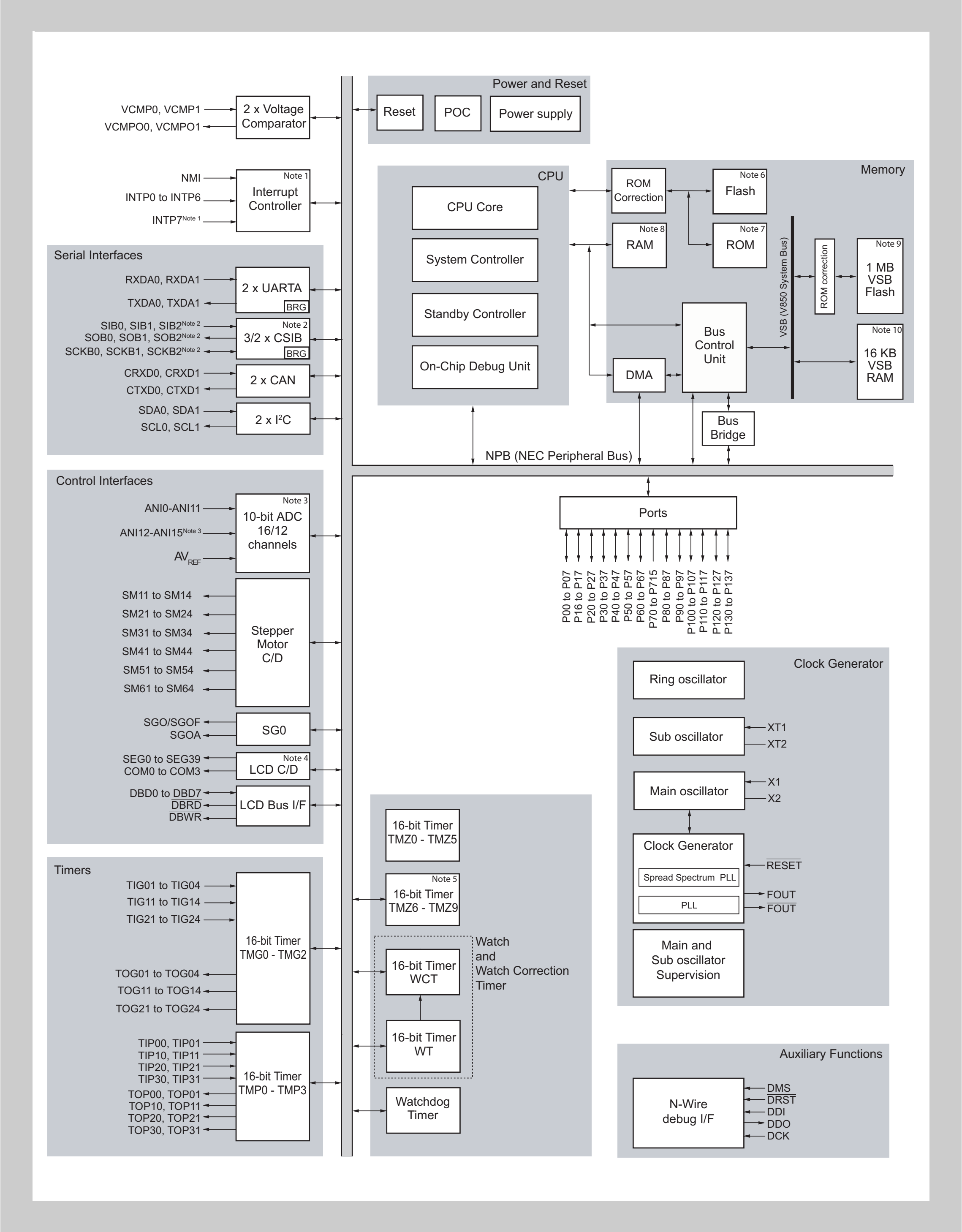


Figure 1-1 **V850E/DJ3 µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423, µPD70F3424, µPD70F3425, µPD70F3426 block diagram**

Table 1-3 summarizes the different features of the of the V850E/DJ3 µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423, µPD70F3424, µPD70F3425, µPD70F3426 microcontrollers, marked as “Notes” in *Figure 1-1*.

Table 1-3 Feature set differences

Note	Feature	‘F3426	‘F3425	‘F3424	‘F3423	‘F3422	‘3422	‘F3421	‘3421	‘F3420	‘3420
1	INTP7	√	√	√	—	—	—	—	—	—	—
2	CSIB2	√	√	√	—	—	—	—	—	—	—
3	ANI12 to ANI15	√	√	√	—	—	—	—	—	—	—
4	LCD-C/D	—	—	—	√	√	√	√	√	√	√
5	TMZ6 to TMZ9	√	√	√	—	—	—	—	—	—	—
6	Flash	1 MB	1 MB	512 KB	512 KB	384 KB	—	256 KB	—	128 KB	—
7	ROM	—	—	—	—	—	384 KB	—	256 KB	—	128 KB
8	RAM	60 KB	32 KB	24 KB	20 KB	16 KB	16 KB	12 KB	12 KB	6 KB	6 KB
9	VSB Flash	1 MB	—	—	—	—	—	—	—	—	—
10	VSB RAM	24 KB	—	—	—	—	—	—	—	—	—

Figure 1-2 provides a functional block diagram of the V850E/DL3 µPD70F3427 microcontroller.

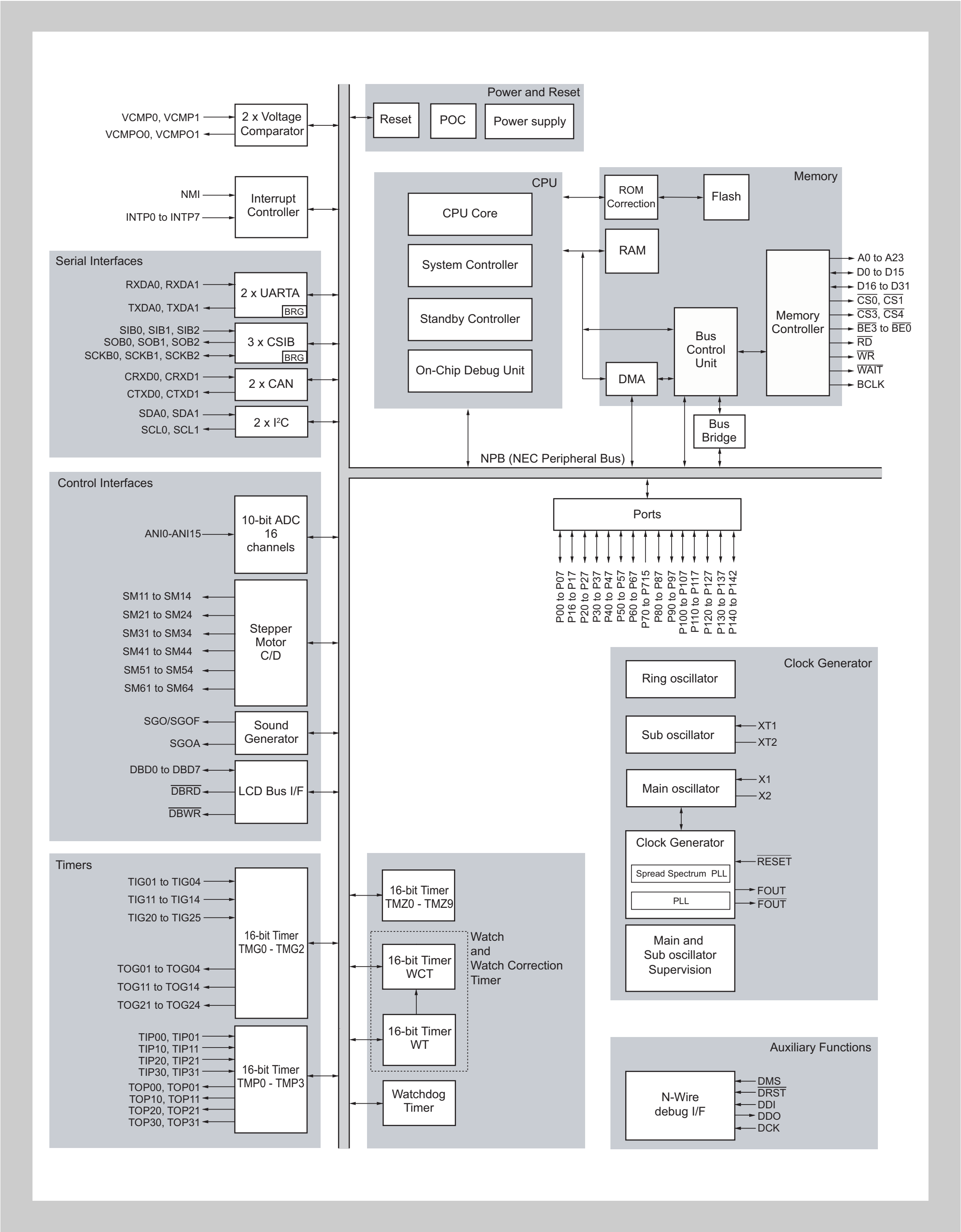


Figure 1-2 V850E/DL3 µPD70F3427 block diagram

Structure of the diagram

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

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

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

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

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

Structure of the manual

This manual explains how to use the V850E/Dx3 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 33
  “CPU System Functions“ on page 103
  “Clock Generator“ on page 129
  “Interrupt Controller (INTC)“ on page 187

- Memory access

  “Flash Memory“ on page 229
  “Bus Control Unit (BCU)“ on page 227
  “DMA Controller (DMAC)“ on page 309
  “ROM Correction Function (ROMC)“ on page 331
  “Code Protection and Security“ on page 339

- Timers

  “16-bit Timer/Event Counter P (TMP)“ on page 343
  “16-bit Interval Timer Z (TMZ)“ on page 429
  “16-bit Multi-Purpose Timer G (TMG)“ on page 437
  “Watch Timer (WT)“ on page 477
  “Watchdog Timer (WDT)“ on page 497

- Serial interfaces

  “Asynchronous Serial Interface (UARTA)“ on page 507
  “Clocked Serial Interface (CSIB)“ on page 541
  “I2C Bus (IIC)“ on page 573
  “CAN Controller (CAN)“ on page 639

- Control interfaces

  “A/D Converter (ADC)“ on page 769
  “Stepper Motor Controller/Driver (Stepper-C/D)“ on page 795
  “LCD Controller/Driver (LCD-C/D)“ on page 809
  “LCD Bus Interface (LCD-I/F)“ on page 823
  “Sound Generator (SG)“ on page 841

- Power and reset

  “Power Supply Scheme“ on page 855
  “Reset“ on page 861
  “Voltage Comparator“ on page 871

- Auxiliary functions

  “On-Chip Debug Unit“ on page 877

1.5 Ordering Information

Table 1-4 V850E/Dx3 ordering information

NEC order code	Pin/package	Memory size	Remarks
UPD703420GJ(A)-GAE-QS-AX	144 pin LQFP	128 KB ROM	–
UPD70F3420GJ(A)-GAE-QS-AX	144 pin LQFP	128 KB flash	–
UPD703421GJ(A)-GAE-QS-AX	144 pin LQFP	256 KB ROM	–
UPD70F3421GJ(A)-GAE-QS-AX	144 pin LQFP	256 KB flash	–
UPD703422GJ(A)-GAE-QS-AX	144 pin LQFP	384 KB ROM	–
UPD70F3422GJ(A)-GAE-QS-AX	144 pin LQFP	384 KB flash	–
UPD70F3423GJ(A)-GAE-QS-AX	144 pin LQFP	512 KB flash	–
UPD70F3424GJ(A)-GAE-QS-AX	144 pin LQFP	512 KB flash	–
UPD70F3425GJ(A)-GAE-QS-AX	144 pin LQFP	1 MB flash	–
UPD70F3426GJ(A)-GAE-QS-AX	144 pin LQFP	2 MB flash	VSB flash and RAM
UPD70F3427GD(A)-LML-QS-AX	208 pin QFP	1 MB flash	External bus I/F

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 39*.

Features summary

- Number of ports and port groups:

Device	Number of ports		Number of port groups
	I/O ports	Input ports	
µPD70F3427	101	16	15
all others	98	16	14

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

2.1.1 Description

This microcontroller has the port groups shown below.

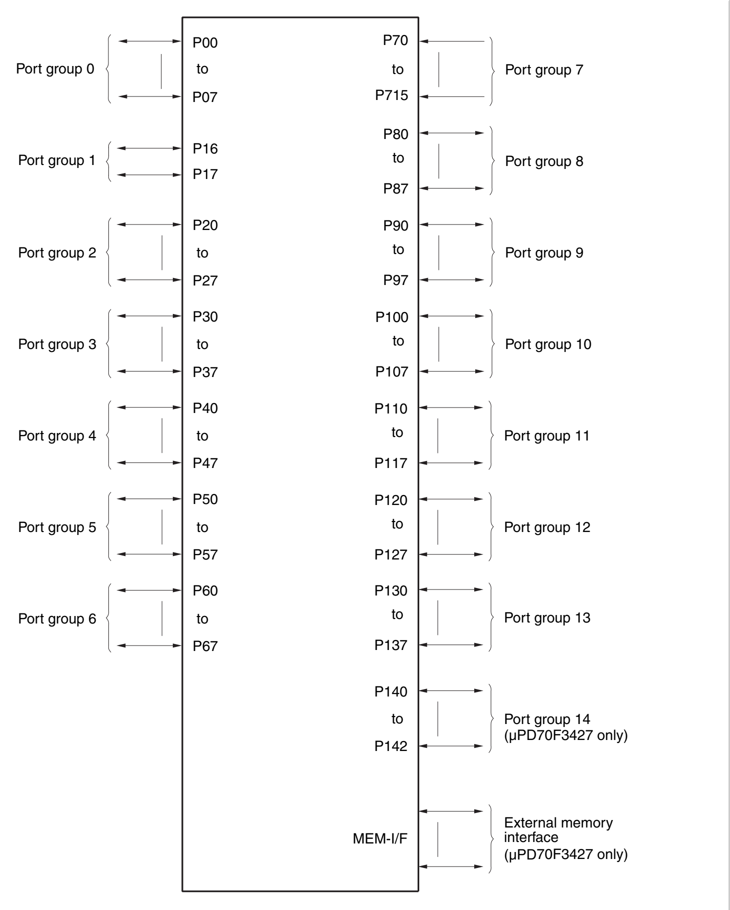


Figure 2-1 Port groups

Port group overview *Table 2-1* gives an overview of the port groups. For each port group it shows the supported functions in port mode and in alternative mode. Any port group can operate in 8-bit or 1-bit units. Port group 7 can additionally operate in 16-bit units.

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

Port group name	Function	
	Port mode	**Alternative mode**
0	8-bit input/output	• External interrupt 0 to 6 • Non maskable interrupt • N-Wire debug interface reset • Output state of internal Voltage Comparators 0
1	2-bit input/output	• I^{2}C0 data/clock line
2	8-bit input/output	• Timer TMG0 to TMG1 channels • I^{2}C1 data/clock line • LCD controller segment signal output (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only)
3	8-bit input/output	• UARTA0 transmit/receive data, • UARTA1 transmit/receive data • I^{2}C1 data/clock line • LCD controller segment signal output (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only) • Timer TMG2 channels • Timer TMP0 to TMP3 channels
4	8-bit input/output	• Clocked Serial Interface CSIB0 data/clock line • Clocked Serial Interface CSIB1 data/clock line • LCD controller segment signal output (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only) • CAN0 transmit/receive data
5	8-bit input/output	• External interrupt 7 (µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 only) • Sound Generator outputs • Frequency output • N-Wire interface signals • CAN1 transmit/receive data • UARTA1 transmit/receive data
6	8-bit input/output	• Timer TMP0 to TMP3 channels • Timer TMG2 channels • LCD controller segment signal output (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only) • I^{2}C0 data/clock line
7	16-bit input	• A/D Converter input - µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427: 16 channels - µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423: 12 channels

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

Port group name	Function	
	Port mode	Alternative mode
8	8-bit input/output	• Clocked Serial Interface CSIB2 data/clock line (µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 only) • LCD controller segment signal output (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only) • Frequency output • UARTA0 transmit/receive data
9	8-bit input/output	• LCD Bus I/F data lines • LCD controller segment/common signal output (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only)
10	8-bit input/output	• Timer TMP0 to TMP3 channels • LCD Bus I/F read/write strobe • LCD controller segment signal output (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only) • Clocked Serial Interface CSIB0 data/clock line
11	8-bit input/output	• Stepper Motor Controller/Driver outputs
12	8-bit input/output	• Stepper Motor Controller/Driver outputs
13	8-bit input/output	• Stepper Motor Controller/Driver outputs • Timer TMG0 to TMG1 channels
14	3-bit input/output	External memory interface (µPD70F3427 only): • Bus clock • Byte enable 2, 3
MEM-I/F	–	External memory interface (µPD70F3427 only): • Address lines 0 to 23 • Chip selects 0, 1, 3, 4 • Read/write strobe • Data wait request • Byte enable 0, 1 • Data lines 0 to 31

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 40*.
- For every port, detailed information on the configuration registers is given in *“Port Group Configuration“ on page 56*.

There are three types of control circuits, defined as port types. For a description of the port types, see *“Port Types 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, P07 denotes port 7 of port group 0. It is referenced as “port P07”.

- **Alternative mode**

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

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

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

- **Port type**

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

2.1.3 Noise elimination

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

See *“Noise Elimination“ on page 93* 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 41*
- *“Pin data input/output“ on page 46*
- *“Configuration of electrical characteristics“ on page 48*
- *“Alternative input selection“ on page 50*

2.2.1 Overview

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

Table 2-2 Registers for port group configuration

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

n = 0 to 14

2.2.2 Pin function configuration

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

- input mode or output mode
- port mode or alternative mode
- selection of one of the alternative output functions ALT1-OUT/ALT2-OUT
- pin usage for LCD Controller/Driver output LCD_OUT
- 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	PLCDC	PMC	PFC	PM	
Port mode (output)	0	0	0	X	0	O
Port mode (input)				X	1	I
Alternative output 1 mode			1	0	0	O
Alternative output 2 mode				1	0	O
Alternative input mode				X	1	I
LCD signal output (segment or common signal)		1	X	X	X	O
On-chip debug mode[a]	1	X	X	X	X	I/O

a) 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.

(1) PMn - Port mode register

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

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

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

Address see *"Port Group Configuration" on page 56*

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

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

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

Table 2-4 PMn register contents

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

(2) PMCn - Port mode control register

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

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

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

Address see *“Port Group Configuration“ on page 56*

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

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

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

Table 2-5 PMCn register contents

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

(3) PFCn - Port function control register

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

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

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

Address see *“Port Group Configuration“ on page 56*

Initial Value PFC0: 20$_H$

other PFCn: 00$_H$

This register is initialized by any reset.

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

Table 2-6 PFCn register contents

Bit position	Bit name	Function
7 to 0	PFCn[7:0]	Specifies the output function of the pin 0: Alternative output mode 1 (ALT1-OUT) 1: Alternative output mode 2 (ALT2-OUT) See *“Port Group Configuration“ on page 56* for a list of the possible output modes.

(4) PLCDCn - Port LCD control register

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

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

Address see *"Port Group Configuration" on page 56*

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

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

Table 2-7 PLCDCn register contents

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

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

(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 877*.

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

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 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
0	0	0	0	0	0	0	OCDM0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	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). 1: Pins are used in on-chip debug mode.

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.

2.2.3 Pin data input/output

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

(1) Pn - Port register

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

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

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

Address see *“Port Group Configuration“ on page 56*

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

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

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

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

Table 2-9 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* for details.

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

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

Table 2-10 Writing/reading register Pn

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

(2) PPRn - Port pin read register

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

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

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

Address see *“Port Group Configuration“ on page 56*

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

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

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

Table 2-11 PPRn register contents

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

2.2.4 Configuration of electrical characteristics

The registers for the configuration of electrical characteristics are briefly described in the following. For details refer to the Electrical Target Specification.

(1) PDSCn - Port drive strength control register

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

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

Address see *“Port Group Configuration“ on page 56*

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

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

Table 2-12 PDSCn register contents

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

For the detailed specification of "Limit 1" and "Limit 2" refer to the Electrical Target Specification.

(2) PICCn - Port input characteristic control register

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

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

Address see *“Port Group Configuration“ on page 56*

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

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

Table 2-13 PICCn register contents

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

(3) PILCn - Port input level control register

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

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

Address see *“Port Group Configuration“ on page 56*

Initial Value 00_H

This register is initialized by any reset.

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

Table 2-14 PILCn register contents

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

(4) PODCn - Port open drain control register

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

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

Address see *“Port Group Configuration“ on page 56*

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

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

Table 2-15 PODCn register contents

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

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

2.2.5 Alternative input selection

Alternative input functions of CSIB0, UART0, UART1, I^{2}C0 and I^{2}C1 are provided on two pins each. Thus you can select on which pin the alternative function should appear. For this purpose, four peripheral function select registers PFSRk (k = 0, 3) are provided.

Note The selection of the alternative *input* function is done by a different circuit than the selection of the alternative *output* function. Therefore, the registers for selecting the alternative input functions (PFSR) are not reflected in the block diagrams of the port types in chapter *“Port Types Diagrams“ on page 52*.

(1) PFSR0 - Peripheral function select register

The 8-bit PFSR0 register selects the alternative input paths for the peripheral functions CSIB0, I^{2}C0 and I^{2}C1.

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

Address FFFF F720$_H$

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

7	6	5	4	3	2	1	0
0[a]	0[a]	PFSR05	PFSR04	0[a]	0[a]	0[a]	PFSR00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

a) These bits must not be changed!

Table 2-16 PFSR0 register contents

Bit position	Bit name	Function
5	PFSR05	Specifies the alternative input path for I^{2}C1: 0: SCL1 is input from P21 (SCL1_0) SDA1 is input from P20 (SDA1_0) 1: SCL1 is input from P31 (SCL1_1) SDA1 is input from P30 (SDA1_1)
4	PFSR04	Specifies the alternative input path for I^{2}C0: 0: SCL0 is input from P17 (SCL0_0) SDA0 is input from P16 (SDA0_0) 1: SCL0 is input from P64 (SCL0_1) SDA0 is input from P65 (SDA0_1)
0	PFSR00	Specifies the alternative input path for CSIB0: 0: SCKB0 is input from P42 (SCKB0_0) SIB0 is input from P40 (SIB0_0) 1: SCKB0 is input from P107 (SCKB0_1) SIB0 is input from P105 (SIB0_1)

(2) PFSR3 - Peripheral function select register

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

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

Address FFFF $F726_H$

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

7	6	5	4	3	2	1	0
0[a]	0[a]	PFSR35	PFSR34	0[a]	0[a]	0[a]	1[a]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

a) These bits must not be changed!

Table 2-17 PFSR3 register contents

Bit position	Bit name	Function
5	PFSR35	Specifies the alternative input path for UARTA1: 0: RXDA1 is input from P33 (RXDA1_0) 1: RXDA1 is input from P56 (RXDA1_1)
4	PFSR34	Specifies the alternative input path for UARTA0: 0: RXDA0 is input from P31 (RXDA0_0) 1: RXDA0 is input from P87 (RXDA0_1)

2.3 Port Types Diagrams

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

(1) Port type M

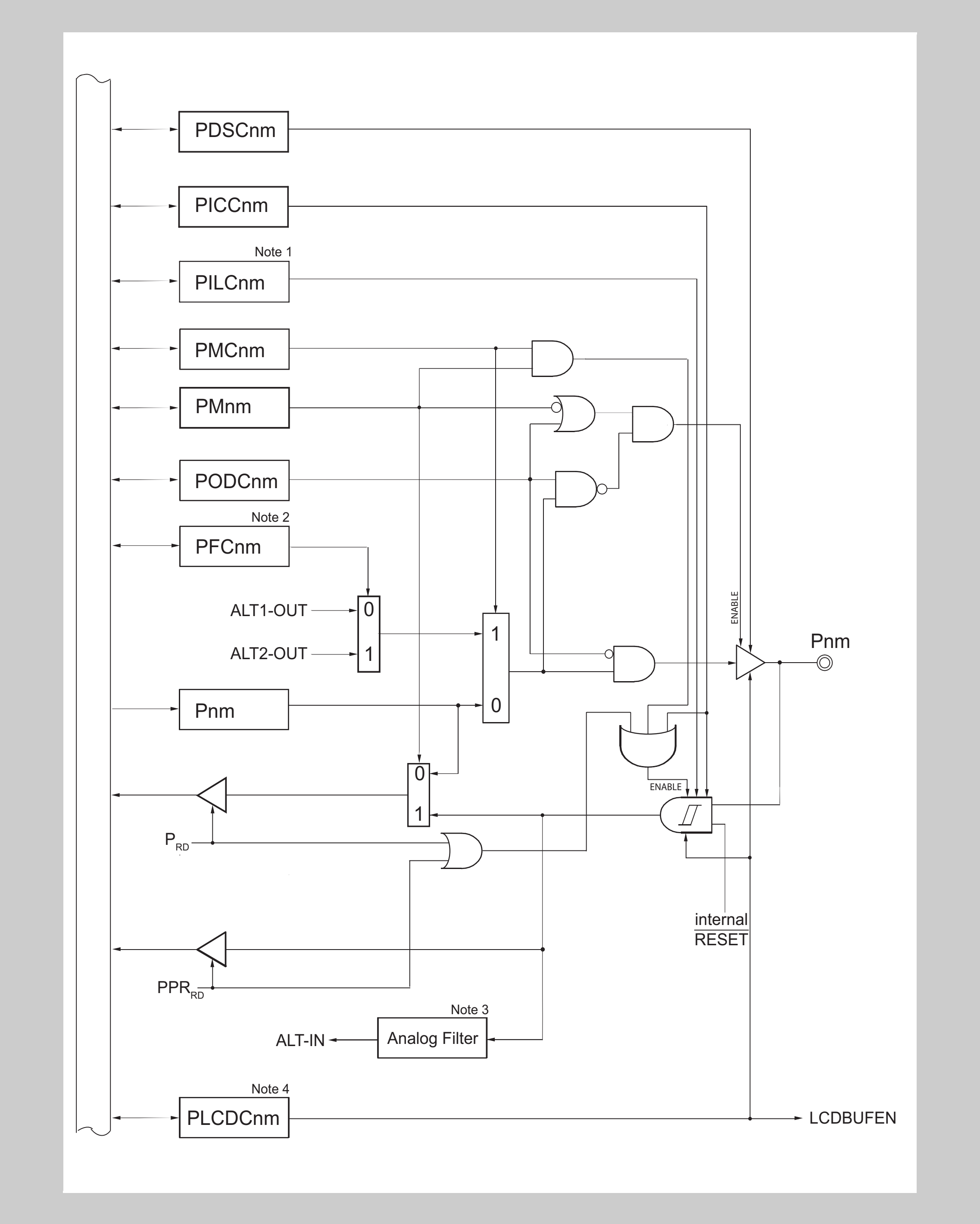


Figure 2-2 Block diagram: port type M

Note

1. Bits PILCn.PILCnm are available only for port group 8.
2. The PFCn register is only available for port groups P0, P3, P5, P6 and P13.
   Note that PFC0 does not select between ALT1-OUT and ALT2-OUT but has a different function, refer to *“Port type R“ on page 55*.
3. The analog filter is provided only for alternative external interrupt ports P00–04, P06, P07.
   The µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 provides an additional analog filter at P50..
4. Bits PLCDCn.PLCDCnm are only provided with µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 for pins with an alternative function as LCD Controller/Driver output ports P20–27, P32–37, P43–45, P60–67, P80–83, P85–87, P90–97, P104–107.

(2) Port type Q

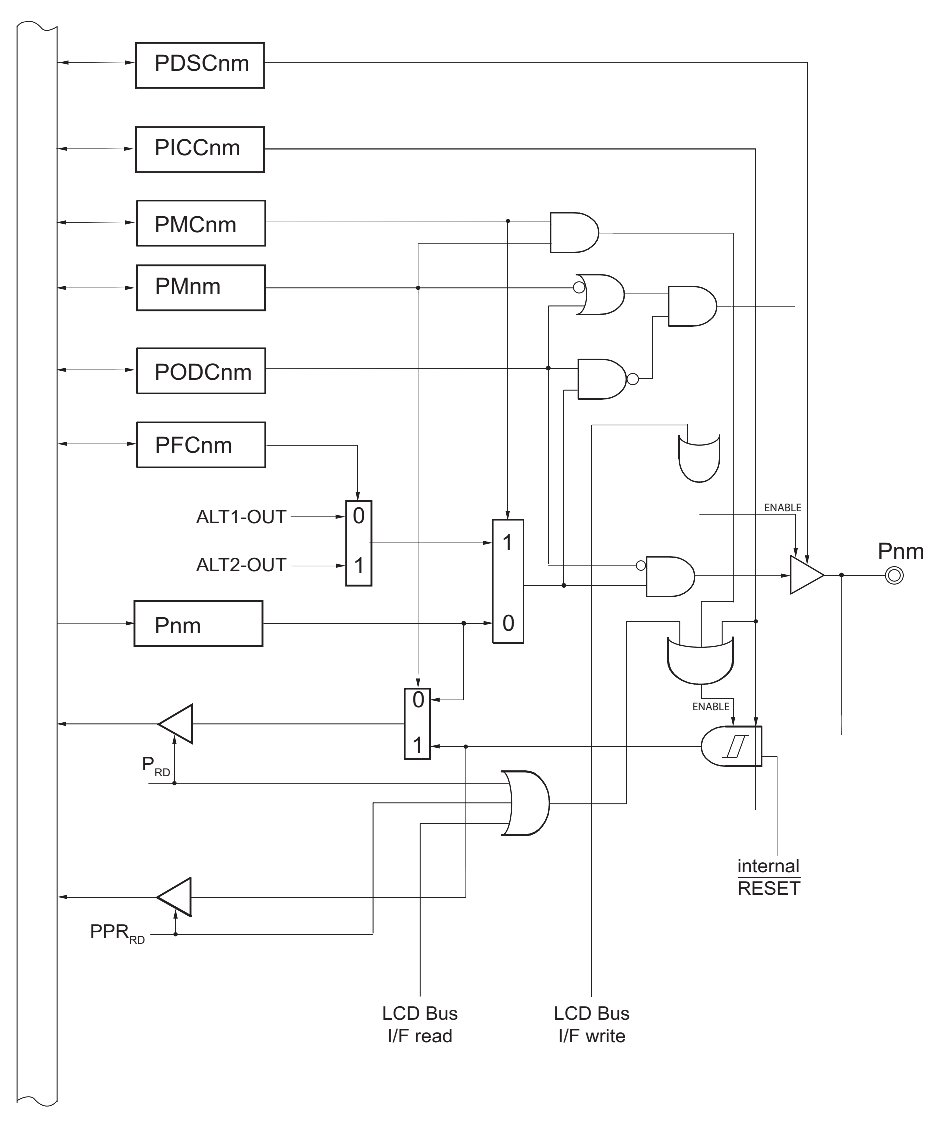


Figure 2-3 Block diagram: port type Q

(3) Port type R

This port type holds for pins that can be used for on-chip debugging with the N-Wire interface.

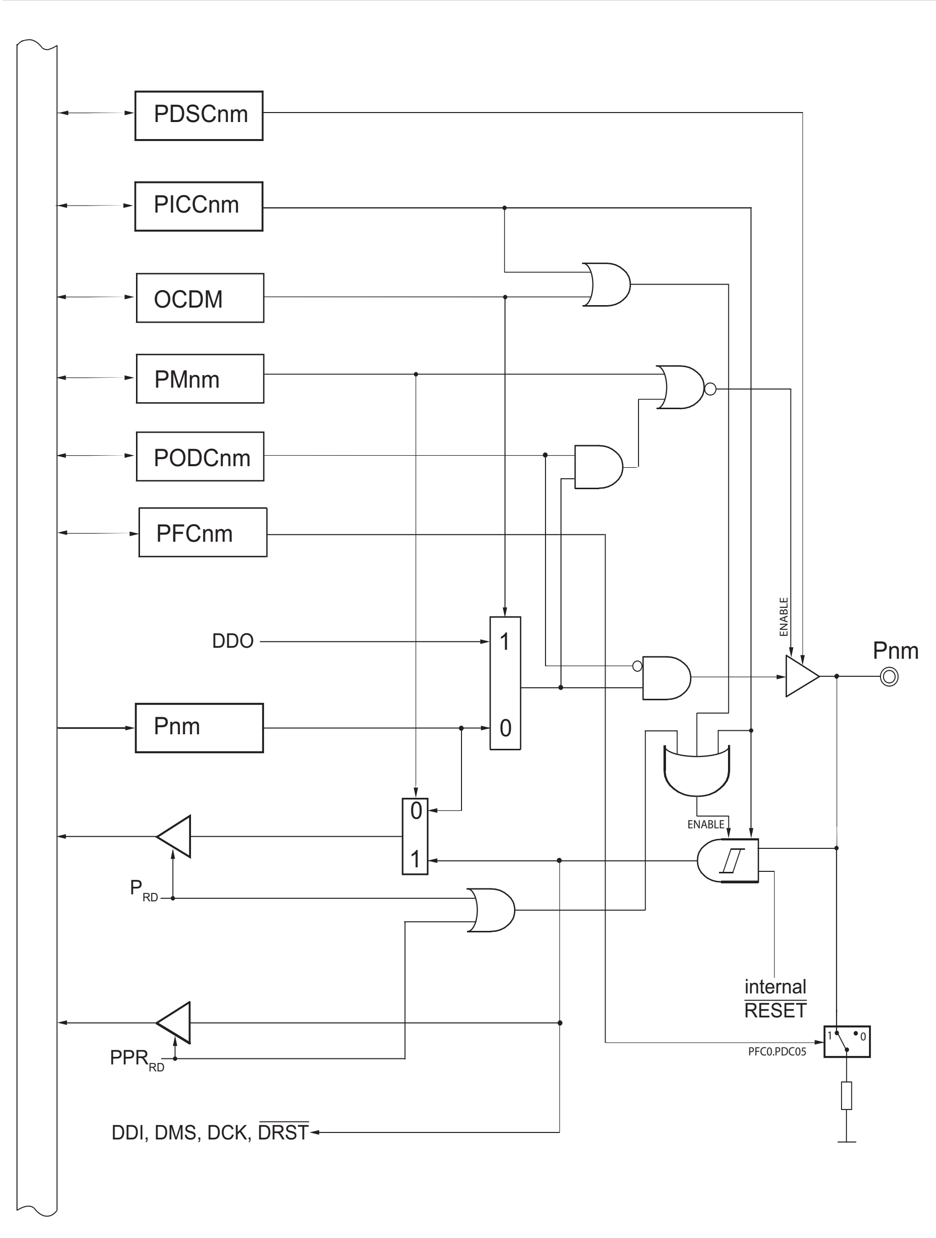


Figure 2-4 Block diagram: port type R

Note If OCDM.OCDM0 = 1, the corresponding pins are operating in on-chip debug mode. The 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 877*.

(4) Port type B

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

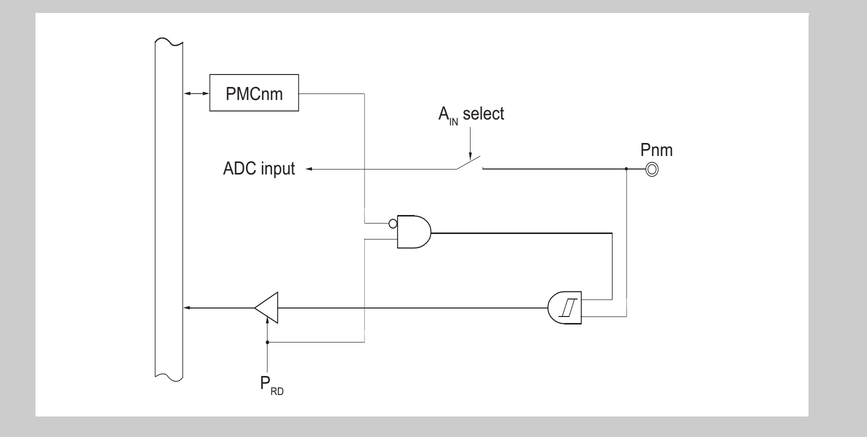


Figure 2-5 Block diagram: port type B

2.4 Port Group Configuration

This section provides an overview of the port groups (*Table 2-18*, *Table 2-19*) and of the pin functions (*Table 2-20 on page 65*). In *Table 2-58 on page 97* it is listed how the pin functions change if the microcontroller is reset or if it is in one of the standby modes.

In the subsections, for every port group the settings of the configuration registers is listed. Further, the addresses and initial values of the configuration registers are given. See *“Port group 0“ on page 72* to *“Port group 13“ on page 91*.

2.4.1 Port group configuration lists

Following tables provide overviews of the functions available at each port pin:

- *Table 2-18* for µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423
- *Table 2-19* for µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427

Table 2-18 Port group list for µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 (1/4)

Port group name	Port name	Alternative outputs ALT1_OUT/ALT2_OUT/ LCD_OUT	Alternative inputs	Port type
0	P00	–	INTP0/NMI	M
	P01	–	INTP1	M
	P02	–	INTP2	M
	P03	–	INTP3	M
	P04	–	INTP4	M
	P05	–	$\overline{\text{DRST}}$	R
	P06	–	INTP5	M
	P07	VCMPO0	INTP6	M
1	P16	SDA0	SDA0	M
	P17	SCL0	SCL0	M
2	P20	SDA1/SEG0	SDA1	M
	P21	SCL1/SEG1	TIG02/SCL1	M
	P22	SEG2	TIG03	M
	P23	SEG3	TIG04	M
	P24	SEG4	TIG11	M
	P25	SEG5	TIG12	M
	P26	SEG6	TIG13	M
	P27	SEG7	TIG14	M
3	P30	TXDA0/SDA1	SDA1	M
	P31	SCL1	RXDA0/SCL1	M
	P32	TXDA1/SEG31	–	M
	P33	SEG29	RXDA1	M
	P34	SEG8/TOG21	–	M
	P35	SEG9/TOG22	TIG22	M
	P36	SEG10/TOG23	TIG23	M
	P37	SEG11/TOG24	TIG24	M
4	P40	–	SIB0	M
	P41	SOB0	–	M
	P42	SCKB0	SCKB0	M
	P43	SEG22	SIB1	M
	P44	SOB1/SEG21	–	M
	P45	SCKB1/SEG20	SCKB1	M
	P46	–	CRXD0	M
	P47	CTXD0	–	M

Table 2-18 Port group list for µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 (2/4)

Port group name	Port name	Alternative outputs ALT1_OUT/ALT2_OUT/ LCD_OUT	Alternative inputs	Port type
5	P50	FOUT/SGOA	–	M
	P51	SGO	–	M
	P52	–	DDI	R
	P53	DDO	–	R
	P54	–	DCK	R
	P55	–	DMS	R
	P56	–	RXDA1/CRXD1	M
	P57	TXDA1/CTXD1	–	M
6	P60	SEG12	TIG20	M
	P61	SEG13	TIP01/TIG21	M
	P62	TOP10/SEG14	TIP10/TIG25	M
	P63	TOP11/SEG15	TIP11	M
	P64	SCL0/SEG16	TIP20/SCL0	M
	P65	SDA0/TOP30/SEG17	TIP30/SDA0	M
	P66	SEG18	TIP21	M
	P67	TOP31/SEG19	TIP31	M
7	P70	–	ANI0	B
	P71	–	ANI1	B
	P72	–	ANI2	B
	P73	–	ANI3	B
	P74	–	ANI4	B
	P75	–	ANI5	B
	P76	–	ANI6	B
	P77	–	ANI7	B
	P78	–	ANI8	B
	P79	–	ANI9	B
	P710	–	ANI10	B
	P711	–	ANI11	B
	P712	–	–	B
	P713	–	–	B
	P714	–	–	B
	P715	–	–	B

Table 2-18 Port group list for µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 (3/4)

Port group name	Port name	Alternative outputs ALT1_OUT/ALT2_OUT/ LCD_OUT	Alternative inputs	Port type
8	P80	SEG26	–	M
	P81	SEG25	–	M
	P82	SEG24	–	M
	P83	SEG23	–	M
	P84	–	–	M
	P85	FOUT/SEG27	–	M
	P86	TXDA0/SEG30	–	M
	P87	SEG28	RXDA0	M
9	P90	DB0/SEG36	–	Q
	P91	DB1/SEG37	–	Q
	P92	DB2/SEG38	–	Q
	P93	DB3/SEG39	–	Q
	P94	DB4/COM0	–	Q
	P95	DB5/COM1	–	Q
	P96	DB6/COM2	–	Q
	P97	DB7/COM3	–	Q
10	P100	TOP00	TIP00	M
	P101	TOP01	–	M
	P102	TOP20	–	M
	P103	TOP21	–	M
	P104	DBRD/SEG35	–	M
	P105	DBWR/SEG34	SIB0	M
	P106	SOB0/SEG33	–	M
	P107	SCKB0/SEG32	SCKB0	M
11	P110	SM11	–	M
	P111	SM12	–	M
	P112	SM13	–	M
	P113	SM14	–	M
	P114	SM21	–	M
	P115	SM22	–	M
	P116	SM23	–	M
	P117	SM24	–	M

Note: Port group 11 is equipped with high drive buffers for stepper motor control.

Table 2-18 Port group list for µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 (4/4)

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

Table 2-19 Port group list for µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 (1/5)

Port group name	Port name	Alternative outputs ALT1_OUT/ALT2_OUT/ LCD_OUT	Alternative inputs	Port type
0	P00	–	INTP0/NMI	M
	P01	–	INTP1	M
	P02	–	INTP2	M
	P03	–	INTP3	M
	P04	–	INTP4	M
	P05	–	$\overline{\text{DRST}}$	R
	P06	–	INTP5	M
	P07	VCMPO0	INTP6	M
1	P16	SDA0	SDA0	M
	P17	SCL0	SCL0	M

Table 2-19 Port group list for μPD70F3424, μPD70F3425, μPD70F3426, μPD70F3427 (2/5)

Port group name	Port name	Alternative outputs ALT1_OUT/ALT2_OUT/ LCD_OUT	Alternative inputs	Port type
2	P20	SDA1	SDA1	M
	P21	SCL1	TIG02/SCL1	M
	P22	–	TIG03	M
	P23	–	TIG04	M
	P24	–	TIG11	M
	P25	–	TIG12	M
	P26	–	TIG13	M
	P27	–	TIG14	M
3	P30	TXDA0/SDA1	SDA1	M
	P31	SCL1	RXDA0/SCL1	M
	P32	TXDA1	–	M
	P33	–	RXDA1	M
	P34	TOP01/TOG21	–	M
	P35	TOP21/TOG22	TIG22	M
	P36	TOP31/TOG23	TIG23	M
	P37	TOP11/TOG24	TIG24	M
4	P40	–	SIB0	M
	P41	SOB0	–	M
	P42	SCKB0	SCKB0	M
	P43	–	SIB1	M
	P44	SOB1	–	M
	P45	SCKB1	SCKB1	M
	P46	–	CRXD0	M
	P47	CTXD0	–	M
5	P50	FOUT/SGOA	INTP7	M
	P51	SGO	–	M
	P52	–	DDI	R
	P53	DDO	–	R
	P54	–	DCK	R
	P55	–	DMS	R
	P56	–	RXDA1/CRXD1	M
	P57	TXDA1/CTXD1	–	M

Table 2-19 Port group list for µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 (3/5)

Port group name	Port name	Alternative outputs ALT1_OUT/ALT2_OUT/ LCD_OUT	Alternative inputs	Port type
6	P60	–	TIG20	M
	P61	–	TIP01/TIG21	M
	P62	TOP10	TIP10TIG25	M
	P63	TOP11	TIP11	M
	P64	SCL0	TIP20/SCL0	M
	P65	SDA0/TOP30	TIP30/SDA0	M
	P66	–	TIP21	M
	P67	TOP31	TIP31	M
7	P70	–	ANI0	B
	P71	–	ANI1	B
	P72	–	ANI2	B
	P73	–	ANI3	B
	P74	–	ANI4	B
	P75	–	ANI5	B
	P76	–	ANI6	B
	P77	–	ANI7	B
	P78	–	ANI8	B
	P79	–	ANI9	B
	P710	–	ANI10	B
	P711	–	ANI11	B
	P712	–	–	B
	P713	–	–	B
	P714	–	–	B
	P715	–	–	B
8	P80	–	SIB2	M
	P81	SOB2	–	M
	P82	SCKB2	SCKB2	M
	P83	–	–	M
	P84	–	–	M
	P85	FOUT	–	M
	P86	TXDA0	–	M
	P87	–	RXDA0	M

Table 2-19 Port group list for µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 (4/5)

Port group name	Port name	Alternative outputs ALT1_OUT/ALT2_OUT/ LCD_OUT	Alternative inputs	Port type
9	P90	DBD0	–	Q
	P91	DBD1	–	Q
	P92	DBD2	–	Q
	P93	DBD3	–	Q
	P94	DBD4	–	Q
	P95	DBD5	–	Q
	P96	DBD6	–	Q
	P97	DBD7	–	Q
10	P100	TOP00	TIP00	M
	P101	TOP01	–	M
	P102	TOP20	–	M
	P103	TOP21	–	M
	P104	DBRD	–	M
	P105	DBWR	SIB0	M
	P106	SOB0	–	M
	P107	SCKB0	SCKB0	M
11	P110	SM11	–	M
	P111	SM12	–	M
	P112	SM13	–	M
	P113	SM14	–	M
	P114	SM21	–	M
	P115	SM22	–	M
	P116	SM23	–	M
	P117	SM24	–	M
Note: Port group 11 is equipped with high drive buffers for stepper motor control.				
12	P120	SM51	–	M
	P121	SM52	–	M
	P122	SM53	–	M
	P123	SM54	–	M
	P124	SM61	–	M
	P125	SM62	–	M
	P126	SM63	–	M
	P127	SM64	–	M
Note: Port group 12 is equipped with high drive buffers for stepper motor control.				

Table 2-19 Port group list for µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 (5/5)

Port group name	Port name	Alternative outputs ALT1_OUT/ALT2_OUT/ LCD_OUT	Alternative inputs	Port type
13	P130	SM31/TOG01	TIG01	M
	P131	SM32/TOG02	–	M
	P132	SM33/TOG03	–	M
	P133	SM34/TOG04	–	M
	P134	SM41/TOG11	–	M
	P135	SM42/TOG12	–	M
	P136	SM43/TOG13	–	M
	P137	SM44/TOG14	–	M
Note: Port group 13 is equipped with high drive buffers for stepper motor control.				
14[a]	P140	BCLK	–	M
	P141	$\overline{\text{BE2}}$	–	M
	P142	$\overline{\text{BE3}}$	–	M
MEM-I/F[a]	–	A[23:0]	–	
	–	$\overline{\text{CS0}}$, $\overline{\text{CS1}}$, $\overline{\text{CS3}}$, $\overline{\text{CS4}}$	–	
	–	$\overline{\text{WR}}$	–	
	–	$\overline{\text{RD}}$	–	
	–	$\overline{\text{BE0}}$, $\overline{\text{BE1}}$	–	
	–	D[15:0]	–	

[a] µPD70F3427 only

2.4.2 Alphabetic pin function list

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

The right columns show also the port, the concerned signal is shared with:
- “Pnm”: signal is shared with port Pnm
- “no ports”: signal has no shared port function
- “–”: signal is not available

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

Pin name	I/O	Pin function	Port		
			‘3420, ‘3421 ‘3422, ‘3423	‘3424, ‘3425 ‘3426	‘3427
A0 to A23	O	External memory interface address lines 0 to 23	–		no ports
ANI0 to ANI11	I	A/D Converter input 0 to 11	P70 to P711		
ANI12 to ANI15	I	A/D Converter input 12 to 15	–	P712 to P715	
AVDD	–	ADC supply voltage	no ports		
AVREF	–	ADC reference voltage input	no ports		
AVSS	–	ADC ground	no ports		
$\overline{\text{BE0}}$, $\overline{\text{BE1}}$	O	External memory interface byte enable signals 0, 1	–		no ports
$\overline{\text{BE2}}$	O	External memory interface byte enable signals 2	–		P141
$\overline{\text{BE3}}$	O	External memory interface byte enable signals 3	–		P142
BCLK	O	External memory interface clock signal	–		P140
BVDD50, BVDD51	–	I/O buffer supply voltage	no ports		
BVSS50, BVSS51	–	I/O buffer supply ground	no ports		
COM0 to COM3	O	LCD common lines 0 to 3	P94 to P97	–	
CRXD0	I	CAN0 receive data	P46		
CRXD1	I	CAN1 receive data	P56		
$\overline{\text{CS0}}$, $\overline{\text{CS1}}$, $\overline{\text{CS3}}$, $\overline{\text{CS4}}$	O	External memory interface chip select signals	–		no ports
CTXD0	O	CAN0 transmit data	P47		
CTXD1	O	CAN1 transmit data	P57		
D0 to D15	I/O	External memory interface data lines 0 to 15	–		no ports

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

Pin name	I/O	Pin function	Port		
			‘3420, ‘3421 ‘3422, ‘3423	‘3424, ‘3425 ‘3426	‘3427
D16	I/O	External memory interface data lines 16 to 31	–		P87
D17					P86
D18					P32
D19					P33
D20					P104
D21					P105
D22					P106
D23					P107
D24					P90
D25					P91
D26					P92
D27					P93
D28					P94
D29					P95
D30					P96
D31					P97
DBD0 to DBD7	I/O	LCD Bus I/F data lines 0 to 7	P90 to P97		
$\overline{\text{DBRD}}$	O	LCD Bus I/F read strobe	P104		
$\overline{\text{DBWR}}$	O	LCD Bus I/F write strobe	P105		
DCK	I	N-Wire interface clock	P54		
DDI	I	N-Wire interface debug data input	P52		
DDO	O	N-Wire interface debug data output	P53		
DMS	I	N-Wire interface debug mode select input	P55		
$\overline{\text{DRST}}$	I	N-Wire debug interface reset	P05		
DVDD50	–	LCD Bus I/F supply voltage	no ports		
DVSS50	–	LCD Bus I/F supply ground	no ports		
DVDD51	–	LCD Bus I/F, D[31:16] ports supply voltage	–		no ports
DVSS51	–	LCD Bus I/F, D[31:16] ports supply ground	–		no ports
FLMD0	I	Primary operating mode select pin	no ports		
FLMD1	I	Secondary operating mode select pin (flash memory devices only)	P07		
FOUT	O	Frequency output	P50, P85		
INTP0 to INTP4	I	External interrupts 0 to 6	P00 to P04		
INTP5 to INTP6			P06 to P07		
INTP7	I	External interrupt 7	–	P50	
MVDD50 to MVDD54	–	External memory interface supply voltage	–		no ports

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

Pin name	I/O	Pin function	Port		
			‘3420, ‘3421 ‘3422, ‘3423	‘3424, ‘3425 ‘3426	‘3427
MVSS50 to MVSS54	–	External memory interface supply ground	–		no ports
NMI	I	Non-maskable interrupt	P00		
$\overline{\text{RD}}$	I/O	External memory interface read strobe	–		no ports
REGC0 to REGC2	–	External capacitor connection	no ports		
$\overline{\text{RESET}}$	I	Reset input	no ports		
RXDA0	I	UARTA0 receive data	P31, P87		
RXDA1	I	UARTA1 receive data	P33, P56		
SCKB0	I/O	Clocked Serial Interface CSIB0 clock line	P42, P107		
SCKB1	I/O	Clocked Serial Interface CSIB1 clock line	P45		
SCKB2	I/O	Clocked Serial Interface CSIB2 clock line	–	P82	
SCL0	I/O	I^{2}C0 clock line	P17, P64		
SCL1	I/O	I^{2}C1 clock line	P21, P31		
SDA0	I/O	I^{2}C0 data line	P16, P65		
SDA1	I/O	I^{2}C1 data line	P20, P30		
SEG0 to SEG7	O	LCD segment lines 0 to 39	P20 to P27	–	
SEG8 to SEG11			P34 to P37		
SEG12 to SEG19			P60 to P67		
SEG20 to SEG22			P45 to P43		
SEG23 to SEG26			P83 to P80		
SEG27			P85		
SEG28			P87		
SEG29			P33		
SEG30			P86		
SEG31			P32		
SEG32 to SEG35			P107 to P104		
SEG36 to SEG39			P90 to P93		
SGO	O	Sound Generator output	P51		
SGOA	O	Sound Generator amplitude PWM output	P50		
SIB0	I	Clocked Serial Interface CSIB0 data input	P40, P105		
SIB1	I	Clocked Serial Interface CSIB1 data input	P43		
SIB2	I	Clocked Serial Interface CSIB2 data input	–	P80	
SM11	O	Stepper motor 1 output sin +	P110		
SM12	O	Stepper motor 1 output sin –	P111		
SM13	O	Stepper motor 1 output cos +	P112		

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

Pin name	I/O	Pin function	Port		
			‘3420, ‘3421 ‘3422, ‘3423	‘3424, ‘3425 ‘3426	‘3427
SM14	O	Stepper motor 1 output cos –	P113		
SM21	O	Stepper motor 2 output sin +	P114		
SM22	O	Stepper motor 2 output sin –	P115		
SM23	O	Stepper motor 2 output cos +	P116		
SM24	O	Stepper motor 2 output cos –	P117		
SM31	O	Stepper motor 3 output sin +	P130		
SM32	O	Stepper motor 3 output sin –	P131		
SM33	O	Stepper motor 3 output cos +	P132		
SM34	O	Stepper motor 3 output cos –	P133		
SM41	O	Stepper motor 4 output sin +	P134		
SM42	O	Stepper motor 4 output sin –	P135		
SM43	O	Stepper motor 4 output cos +	P136		
SM44	O	Stepper motor 4 output cos –	P137		
SM51	O	Stepper motor 5 output sin +	P120		
SM52	O	Stepper motor 5 output sin –	P121		
SM53	O	Stepper motor 5 output cos +	P122		
SM54	O	Stepper motor 5 output cos –	P123		
SM61	O	Stepper motor 6 output sin +	P124		
SM62	O	Stepper motor 6 output sin –	P125		
SM63	O	Stepper motor 6 output cos +	P126		
SM64	O	Stepper motor 6 output cos –	P127		
SMVDD50, SMVDD51	–	Stepper Motor Controller/Driver supply voltage	no ports		
SMVSS50, SMVSS51	–	Stepper Motor Controller/Driver ground	no ports		
SOB0	O	Clocked Serial Interface CSIB0 data output	P41, P106		
SOB1	O	Clocked Serial Interface CSIB1 data output	P44		
SOB2	O	Clocked Serial Interface CSIB2 data output	–	P81	
TIG01	I	Timer TMG0 channels 1 to 4 input	P130		
TIG02 to TIG04			P21 to P23		
TIG11 to TIG14	I	Timer TMG1 channels 1 to 4 input	P24 to P27		
TIG20 to TIG21	I	Timer TMG0 channels 0 to 5 input	P60 to P61		
TIG22 to TIG24			P35 to P37		
TIG25			P62		
TIP00	I	Timer TMP0 channel 0 capture input	P100		
TIP01	I	Timer TMP0 channel 1 capture input	P61		

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

Pin name	I/O	Pin function	Port		
			‘3420, ‘3421 ‘3422, ‘3423	‘3424, ‘3425 ‘3426	‘3427
TIP10	I	Timer TMP1 channel 0 capture input	P62		
TIP11	I	Timer TMP1 channel 1 capture input	P63		
TIP20	I	Timer TMP2 channel 0 capture input	P64		
TIP21	I	Timer TMP2 channel 1 capture input	P66		
TIP30	I	Timer TMP3 channel 0 capture input	P65		
TIP31	I	Timer TMP3 channel 1 capture input	P67		
TOG01 to TOG04	O	Timer TMG0 channels 1 to 4 output	P130 to P133		
TOG11 to TOG14	O	Timer TMG1 channels 1 to 4 output	P134 to P137		
TOG21 to TOG24	O	Timer TMG2 channels 1 to 4 output	P34 to P37		
TOP00	O	Timer TMP0 channel 0 output	P100		
TOP01	O	Timer TMP0 channel 1 output	P34, P101		
TOP10	O	Timer TMP1 channel 0 output	P62		
TOP11	O	Timer TMP1 channel 1 output	P37, P63		
TOP20	O	Timer TMP2 channel 0 output	P102		
TOP21	O	Timer TMP2 channel 1 output	P35, P103		
TOP30	O	Timer TMP3 channel 0 output	P65		
TOP31	O	Timer TMP3 channel 1 output	P36, P67		
TXDA0	O	UARTA0 transmit data	P30, P86		
TXDA1	O	UARTA1 transmit data	P32, P57		
VCMP0	I	Voltage Comparator 0 input	no ports		
VCMP1	I	Voltage Comparator 1 input	no ports		
VCMPO0	O	Output state of internal Voltage Comparator 0	P07		
VDD0 to VDD2	–	Core supply voltage	no ports		
VSS0 to VSS2	–	Core supply ground	no ports		
$\overline{\text{WAIT}}$	I/O	External memory interface data wait request	–		no ports
$\overline{\text{WR}}$	I/O	External memory interface write strobe	–		no ports
X1, X2	–	Main oscillator terminals	no ports		
XT1, XT2	–	Sub-oscillator terminals	no ports		

Note Alternative *input* functions of CSIB0, UART0 and UART1 are provided on two pins each. Thus you can select on which pin the alternative function should appear.
Refer to *“Alternative input selection“ on page 50*.

2.4.3 External memory interface of µPD70F3427

The µPD70F3427 is equipped with an external memory interface. The data bus width can be chosen between 16-bit D[15:0] and 32-bit D[31:0].

The signals of the external memory interface are partly shared with ports respectively alternative functions and are controlled by different means, as listed in *Table 2-21*.

The upper data bus lines D[31:16] are made available by setting PMC.PMC143 = 1. Thus this bit changes the interface from 16- to 32-bit mode.

All other bus interface signals are available via group 14 (also usable as 3-bit I/O port) and the permanent MEM-I/F group.

Table 2-21 External memory interface pin control

Port		Ext. memory I/F signal	Mode control	
Group	**Name**		**Port/alternative**	**Ext. Memory I/F**
3	P32	D18	PMC.PMC143 = 0	PMC.PMC143 = 1
	P33	D19		
8	P86	D17		
	P87	D18		
9	P90	D24		
	P91	D25		
	P92	D26		
	P93	D27		
	P94	D28		
	P95	D29		
	P96	D30		
	P97	D31		
10	P104	D20		
	P105	D21		
	P106	D22		
	P107	D23		
14	P140	BCLK	PMC.PMC140 = 0	PMC.PMC140 = 1
	P141	$\overline{\text{BE2}}$	PMC.PMC141 = 0	PMC.PMC141 = 1
	P142	$\overline{\text{BE3}}$	PMC.PMC142 = 0	PMC.PMC142 = 1
MEM-I/F	–	A[23:0]	–	permanent
		D[15:0]		
		$\overline{\text{WR}}$		
		$\overline{\text{CS0}}$, $\overline{\text{CS1}}$, $\overline{\text{CS3}}$, $\overline{\text{CS4}}$		
		$\overline{\text{RD}}$		
		$\overline{\text{WAIT}}$		
		$\overline{\text{BE0}}$, $\overline{\text{BE1}}$		

2.4.4 Port group 0

- Port group 0 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:
- External interrupt (INTP0 to INTP6)
- Non-maskable interrupt (NMI)
- N-Wire debug interface reset ($\overline{\text{DRST}}$)
- Output state of internal Voltage Comparator 0 (VCMPO0)

Port group 0 includes the following pins:

Table 2-22 Port group 0: pin functions and port types

Pin functions in different modes				Pin function after reset	Port type
Port mode (PMCnm = 0)	**Alternative mode (PMCnm = 1)**		**On-chip debug mode (OCDM0 = 1)**		
	Output mode (PMnm = 0)	**Input mode (PMnm = 1)**			
P00 (I/O)	–	INTP0/NMI	–	P00 (I)	M
P01 (I/O)	–	INTP1	–	P01 (I)	M
P02 (I/O)	–	INTP2	–	P02 (I)	M
P03 (I/O)	–	INTP3	–	P03 (I)	M
P04 (I/O)	–	INTP4	–	P04 (I)	M
P05 (I/O)	–	–	$\overline{\text{DRST}}$ (I)	P05 or $\overline{\text{DRST}}$ (I)[a]	R
P06 (I/O)	–	INTP5	–	P06 (I)	M
P07 (I/O)	VCMPO0	INTP6	–	P07 (I)	M

a) 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 45* and to the *"On-Chip Debug Unit" on page 877*.

Note

1. The setting of bit OCDM.OCDM0 applies only to pins of port type R.
2. For configuring P00 as NMI and/or INTP0 refer also to *"Edge and Level Detection Configuration" on page 218*.

Table 2-23 Port group 0: configuration registers

Register	Address	Initial value	Used bits							
PM0	FFFF F420$_H$	FF$_H$	PM07	PM06	PM05	PM04	PM03	PM02	PM01	PM00
PMC0	FFFF F440$_H$	00$_H$	PMC07	PMC06	X	PMC04	PMC03	PMC02	PMC01	PMC00
PFC0	FFFF F460$_H$	20$_H$	0[a]	X	PDC05[b]	X	X	X	X	X
OCDM	FFFF F9FC$_H$	00$_H$/ 01$_H$[c]	0	0	0	0	0	0	0	OCDM0
P0	FFFF F400$_H$	00$_H$	P07	P06	P05	P04	P03	P02	P01	P00
PPR0	FFFF F3C0$_H$	00$_H$	PPR07	PPR06	PPR05	PPR04	PPR03	PPR02	PPR01	PPR00
PDSC0	FFFF F300$_H$	00$_H$	PDSC07	PDSC06	PDSC05	PDSC04	PDSC03	PDSC02	PDSC01	PDSC00
PICC0	FFFF F380$_H$	00$_H$	PICC07	PICC06	PICC05	PICC04	PICC03	PICC02	PICC01	PICC00
PODC0	FFFF F360$_H$	00$_H$	PODC07	PODC06	PODC05	PODC04	PODC03	PODC02	PODC01	PODC00

a) The default value “0” of this bit must not be changed!

b) Note that PDC05 is used to connect/disconnect the internal pull-down resistor at pin P05/$\overline{\text{DRST}}$.

c) Depends on the reset source (Refer to *“OCDM - On-chip debug mode register“ on page 45* and to *“On-Chip Debug Unit“ on page 877*.

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

2.4.5 Port group 1

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

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

Port group 1 includes the following pins:

Table 2-24 Port group 1: pin functions and port types

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

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

Note Alternative *input* functions SDA0 and SCL0 are provided on two pins each. Thus you can select on which pin the alternative function should appear. If alternative functions SDA0/SCL0 are used at P16/17 make sure to set also PFSR0.PFSR04 = 0.
Refer to *"Alternative input selection" on page 50*.

Table 2-25 Port group 1: configuration registers

Register	Address	Initial value	Used bits							
PM1	FFFF F422$_H$	FF$_H$	PM17	PM16	X	X	X	X	X	X
PMC1	FFFF F442$_H$	00$_H$	PMC17	PMC16	X	X	X	X	X	X
P1	FFFF F402$_H$	00$_H$	P17	P16	X	X	X	X	X	X
PPR1	FFFF F3C2$_H$	00$_H$	PPR17	PPR16	X	X	X	X	X	X
PDSC1	FFFF F302$_H$	00$_H$	PDSC17	PDSC16	X	X	X	X	X	X
PICC1	FFFF F382$_H$	00$_H$	PICC17	PICC16	X	X	X	X	X	X
PODC1	FFFF F362$_H$	00$_H$	PODC17	PODC16	X	X	X	X	X	X

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

2.4.6 Port group 2

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

- Timer TMG0 to TMG1 channels (TIG02 to TIG04, TIG11 to TIG14)
- I^2C1 data/clock line (SDA1, SCL1)
- LCD controller segment signal output (SEG0 to SEG7) (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only)

Port group 2 includes the following pins:

Table 2-26 Port group 2: pin functions and port types

Pin functions in different modes				Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)		LCD mode (PLCDCnm = 1)[a]		
	Output mode (PMnm = 0)	Input mode (PMnm = 1)			
P20 (I/O)	SDA1[b]	SDA1	SEG0[a]	P20 (I)	M
P21 (I/O)	SCL1[b]	TIG02/SCL1	SEG1[a]	P21 (I)	M
P22 (I/O)	–	TIG03	SEG2[a]	P22 (I)	M
P23 (I/O)	–	TIG04	SEG3[a]	P23 (I)	M
P24 (I/O)	–	TIG11	SEG4[a]	P24 (I)	M
P25 (I/O)	–	TIG12	SEG5[a]	P25 (I)	M
P26 (I/O)	–	TIG13	SEG6[a]	P26 (I)	M
P27 (I/O)	–	TIG14	SEG7[a]	P27 (I)	M

a) µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only

b) In I^2C function mode open drain emulation has to be enabled (PODC2.PODC20 = 1 and PODC2.PODC21 = 1). Thus output function is enabled automatically, although PMnm = 1.

Note Alternative *input* functions of I^2C1 (SDA1, SCL1) are provided on two pins each. Thus you can select on which pin the alternative function should appear. If alternative functions SDA1/SCL1 are used at P20/21 make sure to set also PFSR0.PFSR05 = 0.
Refer to *“Alternative input selection“ on page 50*.

Table 2-27 Port group 2: Configuration registers

Register	Address	Initial value	Used bits							
PM2	FFFF $F424_H$	FF_H	PM27	PM26	PM25	PM24	PM23	PM22	PM21	PM20
PMC2	FFFF $F444_H$	00_H	PMC27	PMC26	PMC25	PMC24	PMC23	PMC22	PMC21	PMC20
PLCDC2[a]	FFFF $F344_H$	00_H	PLCDC27	PLCDC26	PLCDC25	PLCDC24	PLCDC23	PLCDC22	PLCDC21	PLCDC20
P2	FFFF $F404_H$	00_H	P27	P26	P25	P24	P23	P22	P21	P20
PPR2	FFFF $F3C4_H$	00_H	PPR27	PPR26	PPR25	PPR24	PPR23	PPR22	PPR21	PPR20
PDSC2	FFFF $F304_H$	00_H	PDSC27	PDSC26	PDSC25	PDSC24	PDSC23	PDSC22	PDSC21	PDSC20
PICC2	FFFF $F384_H$	00_H	PICC27	PICC26	PICC25	PICC24	PICC23	PICC22	PICC21	PICC20
PODC2	FFFF $F364_H$	00_H	PODC27	PODC26	PODC25	PODC24	PODC23	PODC22	PODC21	PODC20

a) µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only

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

2.4.7 Port group 3

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

- UARTA0 transmit/receive data (TXDA0, RXDA0)
- UARTA1 transmit/receive data (TXDA1, RXDA1)
- I^2C1 data/clock line (SDA1, SCL1)
- LCD controller segment signal output (SEG8 to SEG11, SEG29, SEG31) (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only)
- Timer TMG2 channels (TIG21 to TIG24, TOG21 to TOG24)
- Timer TMP0 to TMP3 channels (TOP01 to TOP31) (µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 only)

Port group 3 includes the following pins:

Table 2-28 Port group 3: pin functions and port types

Pin functions in different modes					Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)			LCD mode (PLCDCnm = 1)[a]		
	Output mode (PMnm = 0)		Input mode (PMnm = 1)			
	PFCnm = 0 ALT1-OUT	PFCnm = 1 ALT2-OUT				
P30 (I/O)	TXDA0	SDA1[b]	SDA1	–	P30 (I)	M
P31 (I/O)	SCL1[b]		RXDA0/SCL1	–	P31 (I)	M
P32 (I/O)	TXDA1		–	SEG31[a]	P32 (I)	M
P33 (I/O)	–		RXDA1	SEG29[a]	P33 (I)	M
P34 (I/O)	TOP01	TOG21	–	SEG8[a]	P34 (I)	M
P35 (I/O)	TOP21	TOG22	TIG22	SEG9[a]	P35 (I)	M
P36 (I/O)	TOP31	TOG23	TIG23	SEG10[a]	P36 (I)	M
P37 (I/O)	TOP11	TOG24	TIG24	SEG11[a]	P37 (I)	M

[a] µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only

[b] In I^2C function mode open drain emulation has to be enabled (PODC3.PODC30 = 1 and PODC3.PODC31 = 1). Thus output function is enabled automatically, although PMnm = 1.

Note

1. For pins that support only one alternative output mode, the PFCnm bit is not available.
2. Alternative *input* functions of I^2C1 (SDA1, SCL1) are provided on two pins each. Thus you can select on which pin the alternative function should appear. If alternative functions SDA1/SCL1 are used at P30/31 make sure to set also PFSR0.PFSR05 = 1.
   Refer to *"Alternative input selection" on page 50*.

Table 2-29 Port group 3: configuration registers

Register	Address	Initial value	Used bits							
PM3	FFFF F426$_H$	FF$_H$	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30
PMC3	FFFF F446$_H$	00$_H$	PMC37	PMC36	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30
PFC3	FFFF F466$_H$	00$_H$	PFC37	PFC36	PFC35	PFC34	X	X	X	PFC30
PLCDC3[a]	FFFF F346$_H$	00$_H$	PLCDC37	PLCDC36	PLCDC35	PLCDC34	PLCDC33	PLCDC32	X	X
P3	FFFF F406$_H$	00$_H$	P37	P36	P35	P34	P33	P32	P31	P30
PPR3	FFFF F3C6$_H$	00$_H$	PPR37	PPR36	PPR35	PPR34	PPR33	PPR32	PPR31	PPR30
PDSC3	FFFF F306$_H$	00$_H$	PDSC37	PDSC36	PDSC35	PDSC34	PDSC33	PDSC32	PDSC31	PDSC30
PODC3	FFFF F366$_H$	00$_H$	PODC37	PODC36	PODC35	PODC34	PODC33	PODC32	PODC31	PODC30

a) µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only

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

2.4.8 Port group 4

Port group 4 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:

- Clocked Serial Interface CSIB0 data/clock line (SIB0, SOB0, SCKB0)
- Clocked Serial Interface CSIB1 data/clock line (SIB1, SOB1, SCKB1)
- LCD controller segment signal output (SEG20 to SEG22) (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only)
- CAN0 transmit/receive data (CTXD0, CRXD0)

Port group 4 includes the following pins:

Table 2-30 Port group 4: pin functions and port types

Pin functions in different modes				Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)		LCD mode (PLCDCnm = 1)[a]		
	Output mode (PMnm = 0)	Input mode (PMnm = 1)			
P40 (I/O)	–	SIB0	–	P40 (I)	M
P41 (I/O)	SOB0	–	–	P41 (I)	M
P42 (I/O)	SCKB0	SCKB0	–	P42 (I)	M
P43 (I/O)	–	SIB1	SEG22[a]	P43 (I)	M
P44 (I/O)	SOB1	–	SEG21[a]	P44 (I)	M
P45 (I/O)	SCKB1	SCKB1	SEG20[a]	P45 (I)	M
P46 (I/O)	–	CRXD0	–	P46 (I)	M
P47 (I/O)	CTXD0	–	–	P47 (I)	M

a) µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only

Note The alternative *input* function of CSIB0 isprovided on two pins. Thus you can select on which pin the alternative function should appear.
Refer to *"Alternative input selection" on page 50*.

Table 2-31 Port group 4: configuration registers

Register	Address	Initial value	Used bits							
PM4	FFFF F428$_H$	FF$_H$	PM47	PM46	PM45	PM44	PM43	PM42	PM41	PM40
PMC4	FFFF F448$_H$	00$_H$	PMC47	PMC46	PMC45	PMC44	PMC43	PMC42	PMC41	PMC40
PLCDC4[a]	FFFF F348$_H$	00$_H$	X	X	PLCDC45	PLCDC44	PLCDC43	X	X	X
P4	FFFF F408$_H$	00$_H$	P47	P46	P45	P44	P43	P42	P41	P40
PPR4	FFFF F3C8$_H$	00$_H$	PPR47	PPR46	PPR45	PPR44	PPR43	PPR42	PPR41	PPR40
PDSC4	FFFF F308$_H$	00$_H$	PDSC47	PDSC46	PDSC45	PDSC44	PDSC43	PDSC42	PDSC41	PDSC40
PICC4	FFFF F388$_H$	00$_H$	PICC47	PICC46	PICC45	PICC44	PICC43	PICC42	PICC41	PICC40
PODC4	FFFF F368$_H$	00$_H$	PODC47	PODC46	PODC45	PODC44	PODC43	PODC42	PODC41	PODC40

a) µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only

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

2.4.9 Port group 5

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

- External interrupt (INTP7) (µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 only)
- Sound Generator outputs (SGO, SGOA)
- Frequency output (FOUT)
- N-Wire interface signals (DDI, DDO, DCK, DMS)
- CAN1 transmit/receive data (CTXD1, CRXD1)
- UARTA1 transmit/receive data (TXDA1, RXDA1)

Port group 5 includes the following pins:

Table 2-32 Port group 5: pin functions and port types

Pin functions in different modes					Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)			On-chip debug mode (OCDM0 = 1)		
	Output mode (PMnm = 0)		Input mode (PMnm = 1)			
	PFCnm = 0 ALT1-OUT	PFCnm = 1 ALT2-OUT				
P50 (I/O)	FOUT	SGOA	INTP7[a]	–	P50 (I)	M
P51 (I/O)	SGO		–	–	P51 (I)	M
P52 (I/O)	–		–	DDI (I)	P52 or DDI (I)[b]	R
P53 (I/O)	–		–	DDO (O)	P53 (I) or DDO (O)[b]	R
P54 (I/O)	–		–	DCK (I)	P54 or DCK (I)[b]	R
P55 (I/O)	–		–	DMS(I)	P55 or DMS(I)[b]	R
P56 (I/O)	–		CRXD1/ RXDA1	–	P56 (I)	M
P57 (I/O)	TXDA1	CTXD1	–	–	P57 (I)	M

[a] µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 only

[b] 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 45* and to the *"On-Chip Debug Unit" on page 877*.

Note

1. For pins that support only one alternative output mode, the PFCnm bit is not available.
2. The setting of bit OCDM.OCDM0 applies only to pins of port type R.
3. The alternative *input* function of UARTA1 is provided on two pins. Thus you can select on which pin the alternative function should appear. Refer to *"Alternative input selection" on page 50*.

Table 2-33 Port group 5: configuration registers

Register	Address	Initial value	Used bits							
PM5	FFFF $F42A_H$	FF_H	PM57	PM56	PM55	PM54	PM53	PM52	PM51	PM50
PMC5	FFFF $F44A_H$	00_H	PMC57	PMC56	X	X	X	X	PMC51	PMC50
PFC5	FFFF $F46A_H$	00_H	PFC57	X	X	X	X	X	X	PFC50
OCDM	FFFF $F9FC_H$	$00_H/01_H$	0	0	0	0	0	0	0	OCDM0
P5	FFFF $F40A_H$	00_H	P57	P56	P55	P54	P53	P52	P51	P50
PPR5	FFFF $F3CA_H$	00_H	PPR57	PPR56	PPR55	PPR54	PPR53	PPR52	PPR51	PPR50
PDSC5	FFFF $F30A_H$	00_H	PDSC57	PDSC56	PDSC55	PDSC54	PDSC53	PDSC52	PDSC51	PDSC50
PICC5	FFFF $F38A_H$	00_H	PICC57	PICC56	PICC55	PICC54	PICC53	PICC52	PICC51	PICC50
PODC5	FFFF $F36A_H$	00_H	PODC57	PODC56	PODC55	PODC54	PODC53	PODC52	PODC51	PODC50

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

2.4.10 Port group 6

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

- Timer TMP0 to TMP3 channels (TIP00 to TIP31, TOP00 to TOP31)
- Timer TMG2 channels (TIG20 to TIG25, TOG21 to TOG24)
- LCD controller segment signal output (SEG12 to SEG19) (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only)
- I^{2}C0 data/clock line (SDA0, SCL0)

Port group 6 includes the following pins:

Table 2-34 Port group 6: pin functions and port types

Pin functions in different modes					Pin function after reset	Port type
Port mode (PMCnm = 0)	**Alternative mode (PMCnm = 1)**			**LCD mode (PLCDCnm = 1)[a]**		
	Output mode (PMnm = 0)		**Input mode (PMnm = 1)**			
	PFCnm = 0 ALT1-OUT	**PFCnm = 1 ALT2-OUT**				
P60 (I/O)	–		TIG20	SEG12[a]	P60 (I)	M
P61 (I/O)	–		TIP01/TIG21	SEG13[a]	P61 (I)	M
P62 (I/O)	TOP10		TIP10/TIG25	SEG14[a]	P62 (I)	M
P63 (I/O)	TOP11		TIP11	SEG15[a]	P63 (I)	M
P64 (I/O)	SCL0[b]		SCL0/TIP20	SEG16[a]	P64 (I)	M
P65 (I/O)	SDA0[b]	TOP30	SDA0/TIP30	SEG17[a]	P65 (I)	M
P66 (I/O)	–		TIP21	SEG18[a]	P66 (I)	M
P67 (I/O)	TOP31		TIP31	SEG19[a]	P67 (I)	M

a) µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only

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

Note

1. For pins that support only one alternative output mode, the PFCnm bit is not available.
2. Alternative *input* functions of I^{2}C0 (SDA0, SCL0) are provided on two pins each. Thus you can select on which pin the alternative function should appear. If alternative functions SDA0/SCL0 are used at P64/65 make sure to set also PFSR0.PFSR04 = 1.
   Refer to *"Alternative input selection" on page 50.*

Table 2-35 Port group 6: configuration registers

Register	Address	Initial value	Used bits							
PM6	FFFF F42C$_H$	FF$_H$	PM67	PM66	PM65	PM64	PM63	PM62	PM61	PM60
PMC6	FFFF F44C$_H$	00$_H$	PMC67	PMC66	PMC65	PMC64	PMC63	PMC62	PMC61	PMC60
PFC6	FFFF F46C$_H$	00$_H$	0[a]	0[a]	PFC65	0[a]	0[a]	0[a]	0[a]	X
PLCDC6[b]	FFFF F34C$_H$	00$_H$	PLCDC67	PLCDC66	PLCDC65	PLCDC64	PLCDC63	PLCDC62	PLCDC61	PLCDC60
P6	FFFF F40C$_H$	00$_H$	P67	P66	P65	P64	P63	P62	P61	P60
PPR6	FFFF F3CC$_H$	00$_H$	PPR67	PPR66	PPR65	PPR64	PPR63	PPR62	PPR61	PPR60
PDSC6	FFFF F30C$_H$	00$_H$	PDSC67	PDSC66	PDSC65	PDSC64	PDSC63	PDSC62	PDSC61	PDSC60
PICC6	FFFF F38C$_H$	00$_H$	PICC67	PICC66	PICC65	PICC64	PICC63	PICC62	PICC61	PICC60
PODC6	FFFF F36C$_H$	00$_H$	PODC67	PODC66	PODC65	PODC64	PODC63	PODC62	PODC61	PODC60

a) The default value “0” of these bits must not be changed!
b) µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only

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

2.4.11 Port group 7

Port group 7 is a 16-bit port group. It includes pins for the A/D Converter input.

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

Port group 7 includes the following pins:

Table 2-36 Port group 7: pin functions and port types

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

a) µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 only

Note All pins of port group 7 always function in alternative input mode.

Table 2-37 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	PMC714	PMC713	PMC712	PMC711	PMC710	PMC79	PMC78
PMC7 (16 bit)	FFFF F44E$_H$	0000$_H$	PMC715 to PMC78 (PMC7H)				PMC77 to PMC70 (PMC7L)			
P7L	FFFF F40E$_H$	00$_H$	P77	P76	P75	P74	P73	P72	P71	P70
P7H	FFFF F40F$_H$	00$_H$	P715[a]	P714[a]	P713[a]	P712[a]	P711	P710	P79	P78
P7 (16 bit)	FFFF F04E$_H$	0000$_H$	P715 to PMC78 (P7H)				P77 to P70 (P7L)			

[a] µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 only

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.

2.4.12 Port group 8

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

- Clocked Serial Interface CSIB2 data/clock line (SIB2, SOB2, SCKB2) (µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 only)
- LCD controller segment signal output (SEG23 to SEG28, SEG30) (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only)
- Frequency output (FOUT)
- UARTA0 transmit/receive data (TXDA0, RXDA0)

Port group 8 includes the following pins:

Table 2-38 Port group 8: pin functions and port types

Pin functions in different modes				Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)		LCD mode (PLCDCnm = 1)[a]		
	Output mode (PMnm = 0)	Input mode (PMnm = 1)			
P80 (I/O)	–	SIB2[b]	SEG26[a]	P80 (I)	M
P81 (I/O)	SOB2[b]	–	SEG25[a]	P81 (I)	M
P82 (I/O)	SCKB2[b]	SCKB2[b]	SEG24[a]	P82 (I)	M
P83 (I/O)	–	–	SEG23[a]	P83 (I)	M
P84 (I/O)	–	–	–	P84 (I)	M
P85 (I/O)	FOUT	–	SEG27[a]	P85 (I)	M
P86 (I/O)	TXDA0	–	SEG30[a]	P86 (I)	M
P87 (I/O)	–	RXDA0	SEG28[a]	P87 (I)	M

a) µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only

b) µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 only

Note

1. For pins that support only one alternative output mode, the PFCnm bit is not available.
2. The alternative *input* function of UART0 is provided on two pins. Thus you can select on which pin the alternative function should appear.
   Refer to *“Alternative input selection“ on page 50*.

Table 2-39 Port group 8: configuration registers

Register	Address	Initial value	Used bits							
PM8	FFFF F430$_{H}$	FF$_{H}$	PM87	PM87	PM86	PM85	PM84	PM83	PM82	PM81
PMC8	FFFF F450$_{H}$	00$_{H}$	PMC87	PMC86	PMC85	PMC84	PMC83	PMC82	PMC81	PMC80
PLCDC8[a]	FFFF F350$_{H}$	00$_{H}$	PLCDC87	PLCDC86	PLCDC85	X	PLCDC83	PLCDC82	PLCDC81	PLCDC80
P8	FFFF F410$_{H}$	00$_{H}$	P87	P86	P85	P84	P83	P82	P81	P80
PPR8	FFFF F3D0$_{H}$	00$_{H}$	PPR87	PPR86	PPR85	PPR84	PPR83	PPR82	PPR81	PPR80
PDSC8	FFFF F310$_{H}$	00$_{H}$	PDSC87	PDSC86	PDSC85	PDSC84	PDSC83	PDSC82	PDSC81	PDSC80
PICC8	FFFF F390$_{H}$	00$_{H}$	PICC87	PICC86	PICC85	PICC84	PICC83	PICC82	PICC81	PICC80
PILC8	FFFF F3B0$_{H}$	00$_{H}$	PILC87	PILC86	PILC85	PILC84	PILC83	PILC82	PILC81	PILC80
PODC8	FFFF F370$_{H}$	00$_{H}$	PODC87	PODC86	PODC85	PODC84	PODC83	PODC82	PODC81	PODC80

a) μPD70(F)3420, μPD70(F)3421, μPD70(F)3422, μPD70F3423 only

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

2.4.13 Port group 9

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

- LCD Bus Interface data lines (DBD0 to DBD7)
- LCD controller segment signal output (SEG36 to SEG39) (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only)
- LCD controller common signal output (COM0 to COM4) (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only)

Port group 9 includes the following pins:

Table 2-40 Port group 9: pin functions and port types

Pin functions in different modes				Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)		LCD mode (PLCDCnm = 1)[a]		
	Output mode (PMnm = 0)	Input mode (PMnm = 1)			
P90 (I/O)	DBD0	–	SEG36[a]	P90 (I)	M
P91 (I/O)	DBD1	–	SEG37[a]	P91 (I)	M
P92 (I/O)	DBD2	–	SEG38[a]	P92 (I)	M
P93 (I/O)	DBD3	–	SEG39[a]	P93 (I)	M
P94 (I/O)	DBD4	–	COM0[a]	P94 (I)	M
P95 (I/O)	DBD5	–	COM1[a]	P95 (I)	M
P96 (I/O)	DBD6	–	COM2[a]	P96 (I)	M
P97 (I/O)	DBD7	–	COM3[a]	P97 (I)	M

a) µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only

Note Though DBD0-7 is a bidirectional bus PMnm must be set to "0", i.e. to output mode, when the port is used as LCD bus I/F bus DBD0-7. The change of the direction is performed automatically, when data is read from the external bus.

Table 2-41 Port group 9: configuration registers

Register	Address	Initial value	Used bits							
PM9	FFFF F432$_H$	FF$_H$	PM97	PM96	PM95	PM94	PM93	PM92	PM91	PM90
PMC9	FFFF F452$_H$	00$_H$	PMC97	PMC96	PMC95	PMC94	PMC93	PMC92	PMC91	PMC90
PLCDC9[a]	FFFF F352$_H$	00$_H$	PLCDC97	PLCDC96	PLCDC95	PLCDC94	PLCDC93	PLCDC92	PLCDC91	PLCDC90
P9	FFFF F412$_H$	00$_H$	P97	P96	P95	P94	P93	P92	P91	P90
PPR9	FFFF F3D2$_H$	00$_H$	PPR97	PPR96	PPR95	PPR94	PPR93	PPR92	PPR91	PPR90
PDSC9	FFFF F312$_H$	00$_H$	PDSC97	PDSC96	PDSC95	PDSC94	PDSC93	PDSC92	PDSC91	PDSC90
PICC9	FFFF F392$_H$	00$_H$	PICC97	PICC96	PICC95	PICC94	PICC93	PICC92	PICC91	PICC90
PODC9	FFFF F372$_H$	00$_H$	PODC97	PODC96	PODC95	PODC94	PODC93	PODC92	PODC91	PODC90

a) µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only

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

2.4.14 Port group 10

Port group 10 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:

- Timer TMP0 and TMP2 (TOP00/01, TOP20/21, TIP00)
- LCD Bus Interface read/write strobe ($\overline{\text{DBRD}}$, $\overline{\text{DBWR}}$)
- LCD controller segment signal output (SEG32 to SEG35) (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only)
- Clocked Serial Interface CSIB0 data/clock line (SIB0, SOB0, SCKB0)

Port group 10 includes the following pins:

Table 2-42 Port group 9: pin functions and port types

Pin functions in different modes				Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)		LCD mode (PLCDCnm = 1)[a]		
	Output mode (PMnm = 0)	Input mode (PMnm = 1)			
P100 (I/O)	TOP00	TIP00	–	P100 (I)	M
P101 (I/O)	TOP01	–	–	P101 (I)	M
P102 (I/O)	TOP20	–	–	P102 (I)	M
P103 (I/O)	TOP21	–	–	P103 (I)	M
P104(I/O)	$\overline{\text{DBRD}}$	–	SEG35[a]	P104(I)	M
P105 (I/O)	$\overline{\text{DBWR}}$	SIB0	SEG34[a]	P105 (I)	M
P106 (I/O)	SOB0	–	SEG33[a]	P106 (I)	M
P107 (I/O)	SCKB0	SCKB0	SEG32[a]	P107 (I)	M

[a] µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only

Note Alternative *input* functions of CSIB0 are provided on two pins each. Thus you can select on which pin the alternative function should appear.
Refer to *“Alternative input selection“ on page 50*.

Table 2-43 Port group 10: configuration registers

Register	Address	Initial value	Used bits							
PM10	FFFF F434$_H$	FF$_H$	PM107	PM106	PM105	PM104	PM103	PM102	PM101	PM100
PMC10	FFFF F454$_H$	00$_H$	PMC107	PMC106	PMC105	PMC104	PMC103	PMC102	PMC101	PMC100
PLCDC10[a]	FFFF F354$_H$	00$_H$	PLCDC107	PLCDC106	PLCDC105	PLCDC104	X	X	X	X
P10	FFFF F414$_H$	00$_H$	P107	P106	P105	P104	P103	P102	P101	P100
PPR10	FFFF F3D4$_H$	00$_H$	PPR107	PPR106	PPR105	PPR104	PPR103	PPR102	PPR101	PPR100
PDSC10	FFFF F314$_H$	00$_H$	PDSC107	PDSC106	PDSC105	PDSC104	PDSC103	PDSC102	PDSC101	PDSC100
PICC10	FFFF F394$_H$	00$_H$	PICC107	PICC106	PICC105	PICC104	PICC103	PICC102	PICC101	PICC100
PODC10	FFFF F374$_H$	00$_H$	PODC107	PODC106	PODC105	PODC104	PODC103	PODC102	PODC101	PODC100

[a] µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only

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

2.4.15 Port group 11

Port group 11 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:

- Stepper Motor Controller/Driver outputs (SM11 to SM14, SM21 to SM24)

Port group 11 includes the following pins:

Table 2-44 Port group 11: pin functions and port types

Pin functions in different modes			Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)			
	Output mode (PMnm = 0)	Input mode (PMnm = 1)		
P110 (I/O)	SM11	–	P110 (I)	M
P111 (I/O)	SM12	–	P111 (I)	M
P112 (I/O)	SM13	–	P112 (I)	M
P113 (I/O)	SM14	–	P113 (I)	M
P114 (I/O)	SM21	–	P114 (I)	M
P115 (I/O)	SM22	–	P115 (I)	M
P116 (I/O)	SM23	–	P116 (I)	M
P117 (I/O)	SM24	–	P117 (I)	M

Note Port group 11 is equipped with high driver buffers for stepper motor control.

Table 2-45 Port group 11: configuration registers

Register	Address	Initial value	Used bits							
PM11	FFFF F436$_H$	FF$_H$	PM117	PM116	PM115	PM114	PM113	PM112	PM111	PM110
PMC11	FFFF F456$_H$	00$_H$	PMC117	PMC116	PMC115	PMC114	PMC113	PMC112	PMC111	PMC110
P11	FFFF F416$_H$	00$_H$	P117	P116	P115	P114	P113	P112	P111	P110
PPR11	FFFF F3D6$_H$	00$_H$	PPR117	PPR116	PPR115	PPR114	PPR113	PPR112	PPR111	PPR110
PICC11	FFFF F396$_H$	00$_H$	PICC117	PICC116	PICC115	PICC114	PICC113	PICC112	PICC111	PICC110
PODC11	FFFF F376$_H$	00$_H$	PODC117	PODC116	PODC115	PODC114	PODC113	PODC112	PODC111	PODC110

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

2.4.16 Port group 12

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

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

Port group 12 includes the following pins:

Table 2-46 Port group 12: pin functions and port types

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

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

Table 2-47 Port group 12: configuration registers

Register	Address	Initial value	Used bits							
PM12	FFFF F438$_H$	FF$_H$	0[a]	0[a]	0[a]	0[a]	0[a]	0[a]	0[a]	0[a]
PMC12	FFFF F458$_H$	00$_H$	PMC127	PMC126	PMC125	PMC124	PMC123	PMC122	PMC121	PMC120
P12	FFFF F418$_H$	00$_H$	P127	P126	P125	P124	P123	P122	P121	P120
PPR12	FFFF F3D8$_H$	00$_H$	PPR127	PPR126	PPR125	PPR124	PPR123	PPR122	PPR121	PPR120
PICC12	FFFF F398$_H$	00$_H$	PICC127	PICC126	PICC125	PICC124	PICC123	PICC122	PICC121	PICC120
PODC12	FFFF F378$_H$	00$_H$	PODC127	PODC126	PODC125	PODC124	PODC123	PODC122	PODC121	PODC120

a) The default value “0” of these pins must not be changed!

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

2.4.17 Port group 13

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

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

Port group 13 includes the following pins:

Table 2-48 Port group 13: pin functions and port types

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

Note

1. Alternative *input* functions of TMG0…TMG1 are provided on two pins each. Thus you can select on which pin the alternative function should appear.
   Refer to *"Alternative input selection" on page 50*.
2. Port group 13 is equipped with high driver buffers for stepper motor control.

Table 2-49 Port group 13: configuration registers

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

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

2.4.18 Port group 14 (µPD70F3427 only)

Port group 14 is a 3-bit port group. In alternative mode, it comprises pins for the following functions:

- External memory interface bus clock BCLK
- External memory interface byte enable signals $\overline{\text{BE2}}$, $\overline{\text{BE3}}$

Port group 14 includes the following pins:

Table 2-50 Port group 14: pin functions and port types

Pin functions in different modes			Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)			
	Output mode (PMnm = 0)	Input mode (PMnm = 1)		
P140 (I/O)	BCLK	–	P140 (I)	M
P141 (I/O)	$\overline{\text{BE2}}$	–	P141 (I)	M
P142 (I/O)	$\overline{\text{BE3}}$	–	P142 (I)	M

Table 2-51 Port group 14: configuration registers

Register	Address	Initial value	Used bits							
PM14	FFFF F43C$_H$	FF$_H$	X	X	X	X	X	PM142	PM141	PM140
PMC14	FFFF F45C$_H$	00$_H$	X	X	X	X	PMC143[a]	PMC142	PMC141	PMC140
P14	FFFF F41C$_H$	00$_H$	X	X	X	X	X	P142	P141	P140
PPR14	FFFF F3DC$_H$	00$_H$	X	X	X	X	X	PPR142	PPR141	PPR140

[a] PMC143 specifies the data bus width of the external memory interface:
- PMC143 = 0: 16-bit data bus D[15:0], D[31:16] pins of port groups 3, 8, 9, 10 operate in port/alternative mode
- PMC143 = 1: 32-bit data bus D[31:0], D[31:16] pins of port groups 3, 8, 9, 10 operate as data bus pins

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

2.5 Noise Elimination

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

2.5.1 Analog filtered inputs

The external interrupts INTP0…INTP7, NMI and the external $\overline{\text{RESET}}$ input are passed through an analog filter to remove noise and glitches. The analog filter suppresses input pulses that are shorter than a specified puls width (refer to the Electrical Target Specification). This assures the hold time for the external interrupt signals.

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

2.5.2 Digitally filtered inputs

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

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

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

Table 2-52 Digitally filtered external signals

Module	Signal	Comment
CSIB0	SIB0, SCKB0	For high clock rates of the Clocked Serial Interface, the digital filter should be disabled. Otherwise, desired input pulses may be removed by the digital filter.
CSIB1	SIB1, SCKB1	
CSIB2	SIB2, SCKB2	
TMP0	TIP00, TIP01	
TMP1	TIP10, TIP11	
TMP2	TIP20, TIP21	
TMP3	TIP30, TIP31	
TMG0	TIG01 to TIG04	
TMG1	TIG11 to TIG14	
TMG2	TIG20 to TIG25	

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

Filter operation The input terminal signal is sampled with the sampling frequency f_S. Spikes shorter than 2 sampling cycles are suppressed and no internal signal is generated. Pulses longer than 3 sampling cycles are recognized as valid pulses and an internal signal is generated. For pulses between 2 and 3 sampling cycles, the behaviour is not defined. The filter operation is illustrated in *Figure 2-6*.

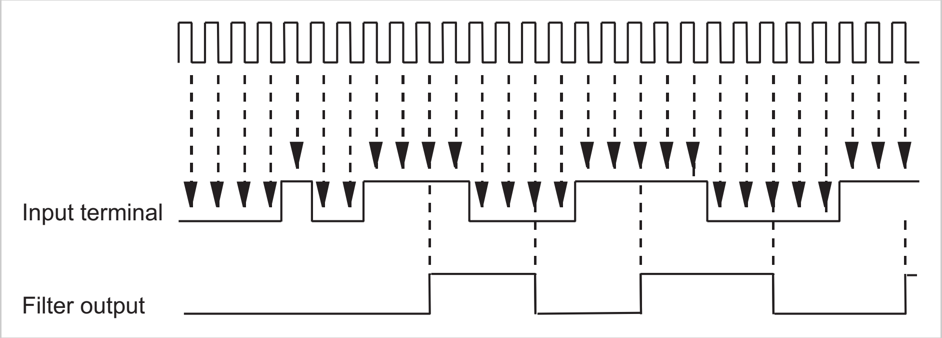


Figure 2-6 Digital noise removal example

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

Table 2-53 Digital noise removal features

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

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

(1) DFEN0 - Digital filter enable register

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

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

Address FFFF F710$_H$

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

15	14	13	12	11	10	9	8
DFENC15	DFENC14	DFENC13	DFENC12	DFENC11	DFENC10	DFENC9	DFENC8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
DFENC7	DFENC6	DFENC5	DFENC4	DFENC3	DFENC2	DFENC1	DFENC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-54 DFEN0 register contents

Bit position	Bit name	Function
15 to 0	DFENC[15:0]	Enables/disables the digital noise elimination filter for the corresponding input signal: 0: Digital filter is disabled. 1: Digital filter is enabled. For an assignment of bit positions to input signals see table *Table 2-55*.

Table 2-55 Assignment of input signals to bit positions for register DFEN0

Bit position	Bit name	Input signal	Description
0	DFENC0	SIB0	CSIB0 data input[a]
1	DFENC1	SIB1	CSIB1 data input[a]
2	DFENC2	SIB2	CSIB2 data input[a]
3	DFENC3	SCKIB0	CSIB0 clock input[a]
4	DFENC4	SCKIB1	CSIB1 clock input[a]
5	DFENC5	SCKIB2	CSIB2 clock input[a]
6	DFENC6	TIP00	Timer TMP0 channel 0 capture input
7	DFENC7	TIP01	Timer TMP0 channel 1 capture input
8	DFENC8	TIP10	Timer TMP1 channel 0 capture input
9	DFENC9	TIP11	Timer TMP1 channel 1 capture input
10	DFENC10	TIP20	Timer TMP2 channel 0 capture input
11	DFENC11	TIP21	Timer TMP2 channel 1 capture input
12	DFENC12	TIP30	Timer TMP3 channel 0 capture input
13	DFENC13	TIP31	Timer TMP3 channel 1 capture input
14	DFENC14	TIG01	Timer TMG0 channel 1 capture input
15	DFENC15	TIG02	Timer TMG0 channel 2 capture input

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

(2) DFEN1 - Digital filter enable register

The 16-bit DFEN1 register enables/disables the digital filter for TMG0 to TMG2 and TMP0 to TMP1 input channels.

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

Address FFFF F712$_H$

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

15	14	13	12	11	10	9	8
X	X	X	X	DFENC27	DFENC26	DFENC25	DFENC24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
DFENC23	DFENC22	DFENC21	DFENC20	DFENC19	DFENC18	DFENC17	DFENC16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-56 DFEN1 register contents

Bit position	Bit name	Function
11 to 0	DFENC[27:16]	Enables/disables the digital noise elimination filter for the corresponding input signal: 0: Digital filter is disabled. 1: Digital filter is enabled. For an assignment of bit positions to input signals see table *Table 2-57*.

Table 2-57 Assignment of input signals to bit positions for register DFEN1

Bit position	Bit name	Input signal	Description
0	DFENC16	TIG03	Timer TMG0 channel 3 capture input
1	DFENC17	TIG04	Timer TMG0 channel 4 capture input
2	DFENC18	TIG11	Timer TMG1 channel 1 capture input
3	DFENC19	TIG12	Timer TMG1 channel 2 capture input
4	DFENC20	TIG13	Timer TMG1 channel 3 capture input
5	DFENC21	TIG14	Timer TMG1 channel 4 capture input
6	DFENC22	TIP00/TIG20	shared input: Timer TMP0 channel 0 capture input / Timer TMG2 channel 0 capture input
7	DFENC23	TIG21	Timer TMG2 channel 1 capture input
8	DFENC24	TIG22	Timer TMG2 channel 2 capture input
9	DFENC25	TIG23	Timer TMG2 channel 3 capture input
10	DFENC26	TIG24	Timer TMG2 channel 4 capture input
11	DFENC27	TIP10/TTIG25	shared input: Timer TMP1 channel 0 capture input/ Timer TMG2 channel 5 capture input

2.6 Pin Functions in Reset and Power Save Modes

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

The reset source makes a difference concerning the N-Wire debugger interface pins DRST, DDI, DDO, DCK and DMS after reset release. An external 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 2-58 Pin functions and reset / power save modes

Operating status		Pin status
Power-On-Clear	during	• P05/DRST: P05 port input with internal pull-down resistor • all other pins: Hi-Z (3-state)
	after	input port mode
external RESET	during	• P05/DRST: P05 port input with internal pull-down resistor • all other pins: Hi-Z (3-state)
	after	• P05/DRST: 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
all other reset sources	during	• P05/DRST: P05 port input with internal pull-down resistor • all other pins: Hi-Z (3-state)
	after	• P05/DRST, P52/DDI, P54/DCK, P55/DMS, P53/DDO: no change. same function as before reset • all other pins: input port mode
HALT mode	during	same as before HALT mode
	after	
IDLE, WATCH, Sub-WATCH, STOP mode	during	same as before power save mode: • Output signals are valid and output levels are remained. • Input signals with wake-up capability[a] are valid. • Input signals without wake-up capability are ignored.
	after	same as before power save mode

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

If flash programming mode is enabled by FLMD0 = 1 P07 is used as FLMD1 pin in input port mode during and after reset.

2.7 Recommended connection of unused pins

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

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

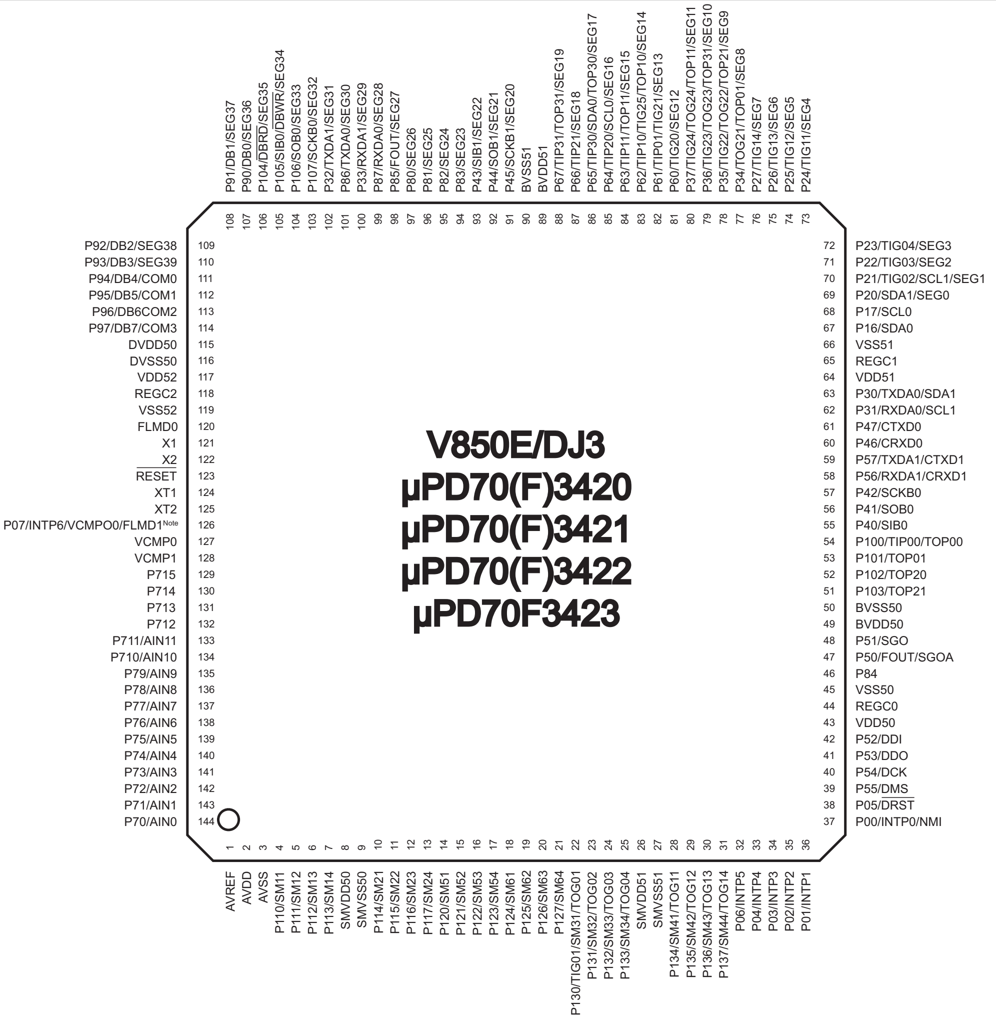
Sub oscillator connection If no sub oscillator crystal is connected, connect XT1 to V_{ss} and leave XT2 open.

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

2.8 Package Pins Assignment

The following sections show the location of pins in top view. Every pin is labelled with its pin number and all possible pin names. Additionally, a recommendation for the connection of unused pins is given at the end of the chapter.

2.8.1 µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 — 144 pin package



Note: FLMD1 for flash memory devices only

Figure 2-7 Pin overview of devices µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423

2.8.2 µPD70F3424, µPD70F3425, µPD70F3426 — 144 pin package

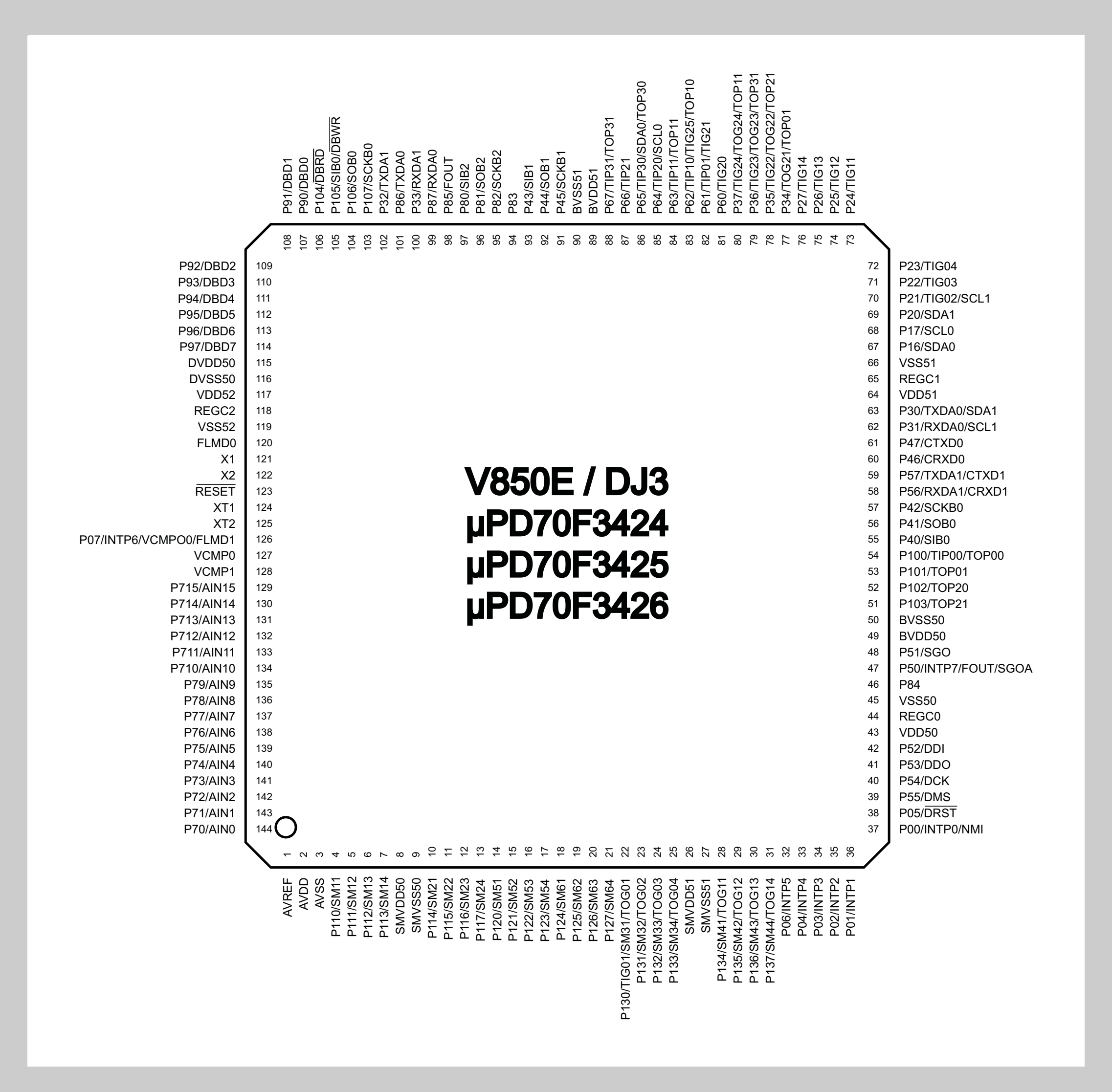


Figure 2-8 Pin overview of device µPD70F3424, µPD70F3425, µPD70F3426

2.8.3 µPD70F3427 — 208 pin package

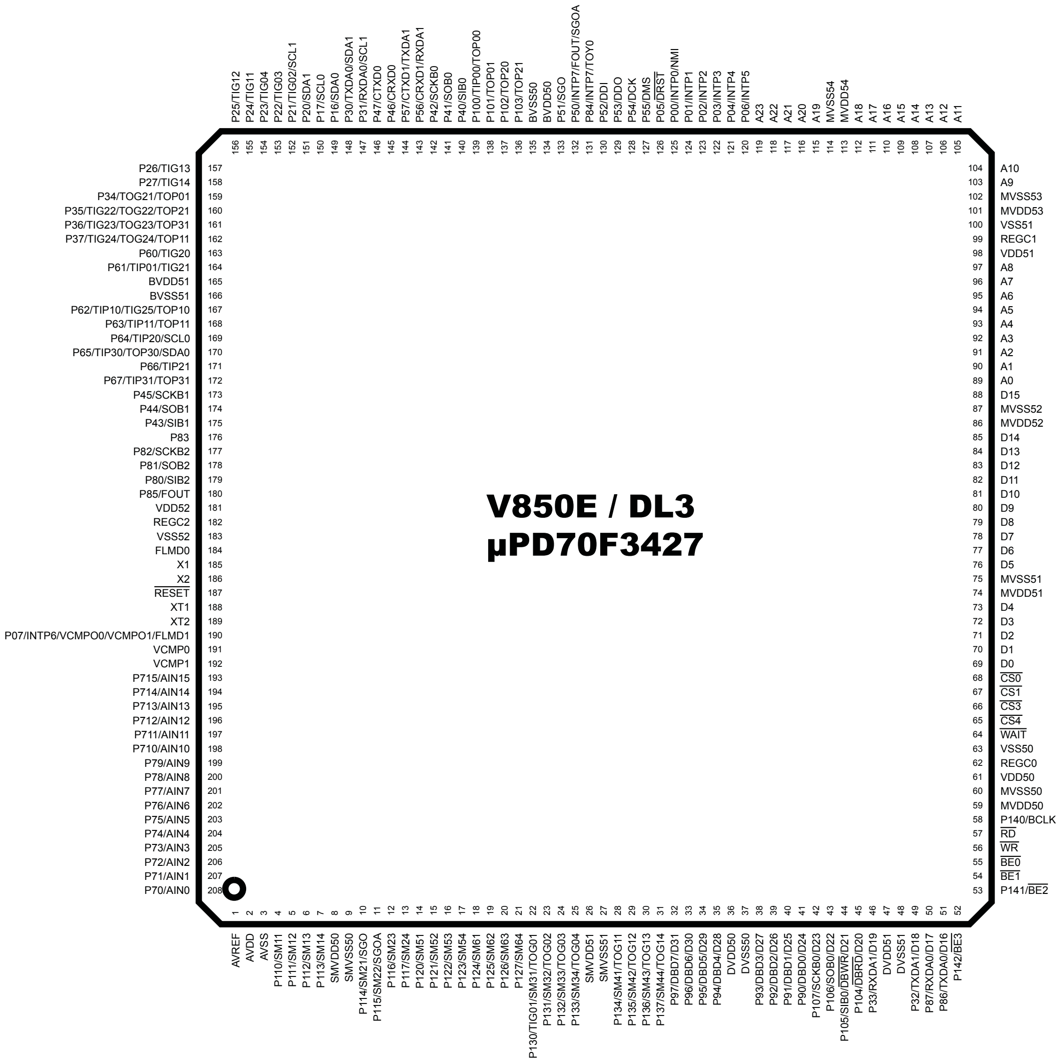


Figure 2-9 Pin overview of µPD70F3427

Chapter 3 CPU System Functions

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

3.1 Overview

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

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

Features summary The CPU has the following special features:

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

3.1.1 Description

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

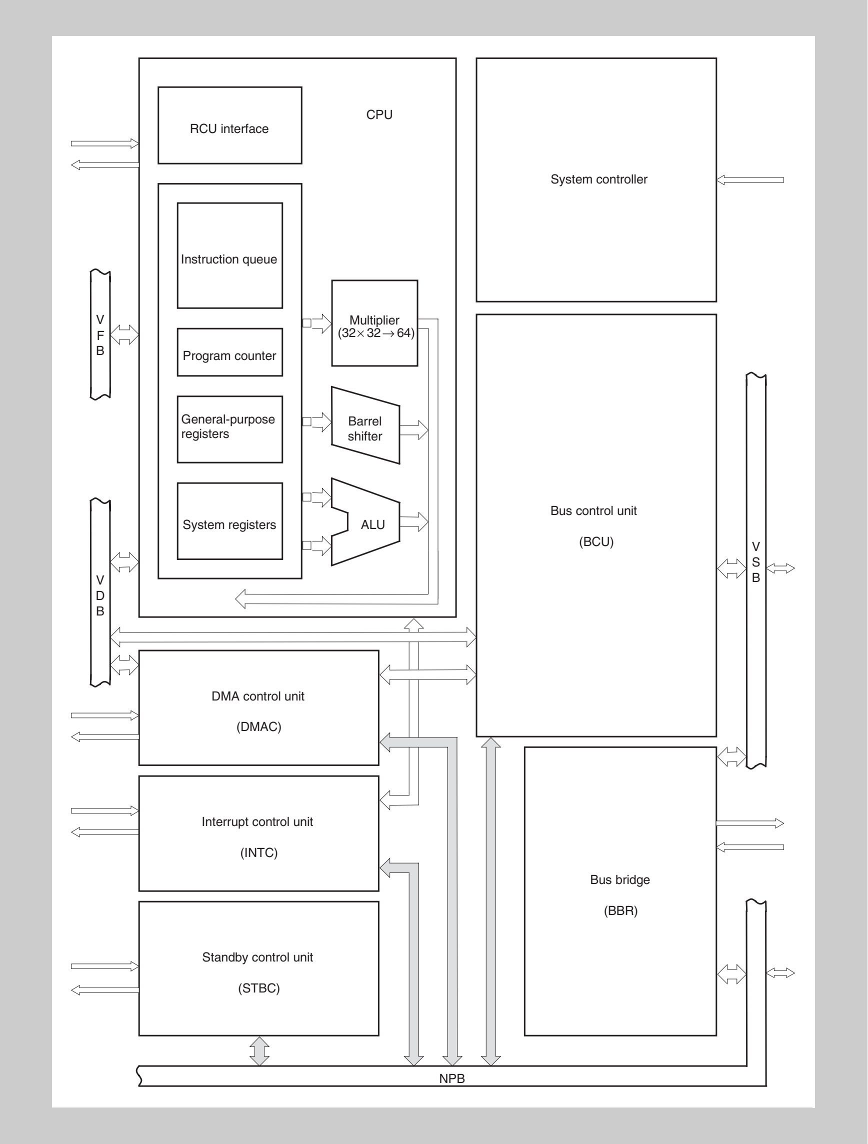


Figure 3-1 CPU system

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

Table 3-1 Bus types

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

3.2 CPU Register Set

There are two categories of registers:

- General purpose registers
- System registers

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

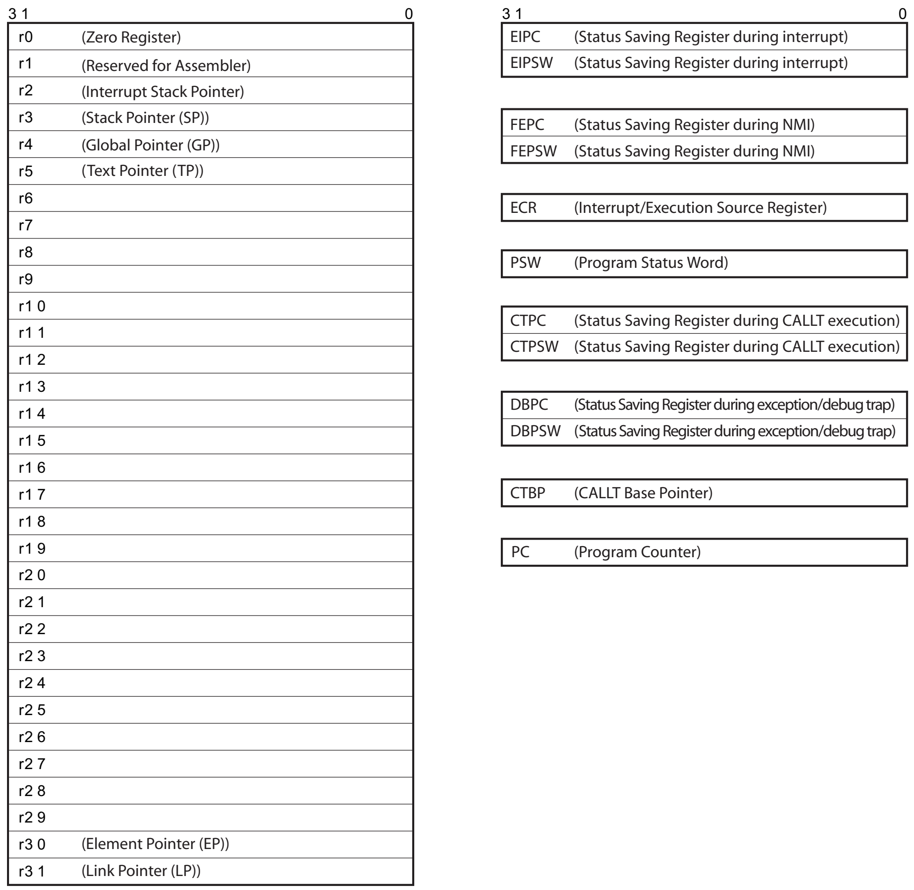


Figure 3-2 CPU register set

Some registers are write protected. That means, writing to those registers is protected by a special sequence of instructions. Refer to *"Write Protected Registers" on page 124* for more details.

3.2.1 General purpose registers (r0 to r31)

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

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

Table 3-2 General purpose registers

Register name	Usage	Operation
r0	Zero register	Always holds 0. It is used for operations using 0 and offset 0 addressing.[a]
r1	Assembler-reserved register	Used for 32-bit direct addressing.[b]
r2	User address/data variable register	
r3	Stack pointer	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 and ROM Correction detections (refer to *“Interrupt Controller (INTC)“ on page 187* and *“ROM Correction Function (ROMC)“ on page 331*).

(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 112* and to the *“Interrupt Controller (INTC)“ on page 187*.

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

Table 3-4 PC saving registers

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

[a] Reading from this register is only enabled between a DBTRAP exception (exception handler address 0000 0060_H) and the exception handler terminating DBRET instruction. DBTRAP exceptions are generated upon ILGOP and ROM Correction detections (refer to *“Interrupt Controller (INTC)“ on page 187* and *“ROM Correction Function (ROMC)“ on page 331*).

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

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

(3) PSW - Program status word

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

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

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

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

Table 3-5 PSW register contents

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

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

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

Table 3-6 Saturation-processed operation result

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

[a] Retains the value before operation.

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

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

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

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

Table 3-7 PSW saving registers

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

[a] Reading from this register is only enabled between a DBTRAP exception (exception handler address 0000 0060_H) and the exception handler terminating DBRET instruction. DBTRAP exceptions are generated upon ILGOP and ROM Correction detections (refer to *“Interrupt Controller (INTC)“ on page 187* and *“ROM Correction Function (ROMC)“ on page 331*).

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 112*.

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 187*	Interrupt	refer to *“Interrupt Controller (INTC)“ on page 187*	• 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 acknowlegded 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.

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

- Normal operation mode
- Flash programming mode

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

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

Pins		Operation Mode
FLMD0	**FLMD1 (P07)**	
0	X	Normal operation mode (fetch from flash)
1	0	Flash programming mode
	1	Setting prohibited

Note The FLMD1 pin function is shared with the P07 pin.

ROM mask devices Since mask ROM devices do not have any programmable flash memory on-chip the only operation mode is the normal operation mode.

Caution The FLMD0 pin of ROM mask devices must be set to low level during and after reset.

3.3.1 Normal operation mode

Flash memory devices

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

After reset release, the firmware acquires the user's reset vector from the 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.

ROM mask devices

After reset release the user’s program is always started at address $0000\ 0000_H$.

3.3.2 Flash programming mode (flash memory devices only)

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

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

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

For more information see section *“Flash Memory“ on page 229*.

3.4 Address Space

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

3.4.1 CPU address space and physical address space

The CPU supports the following address space:

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

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

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

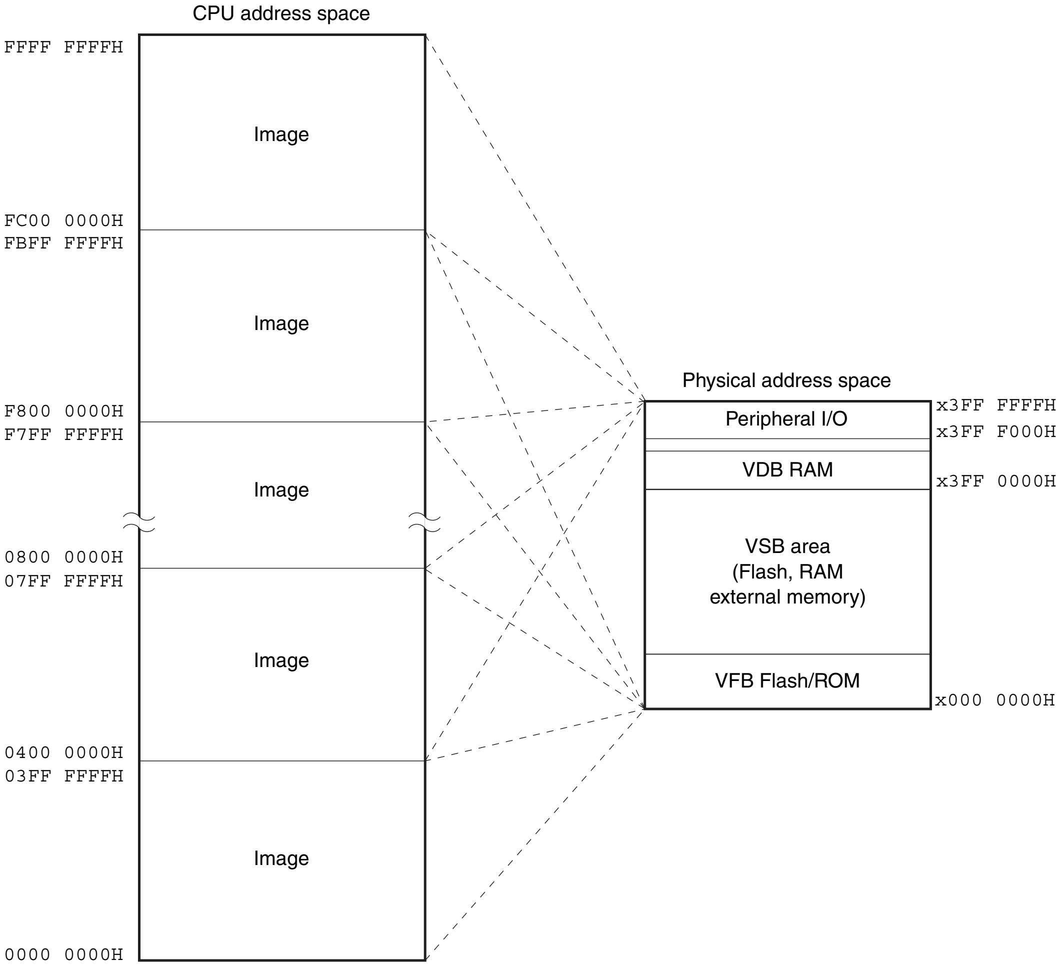


Figure 3-3 Images in the CPU address space

3.4.2 Program and data space

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

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

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

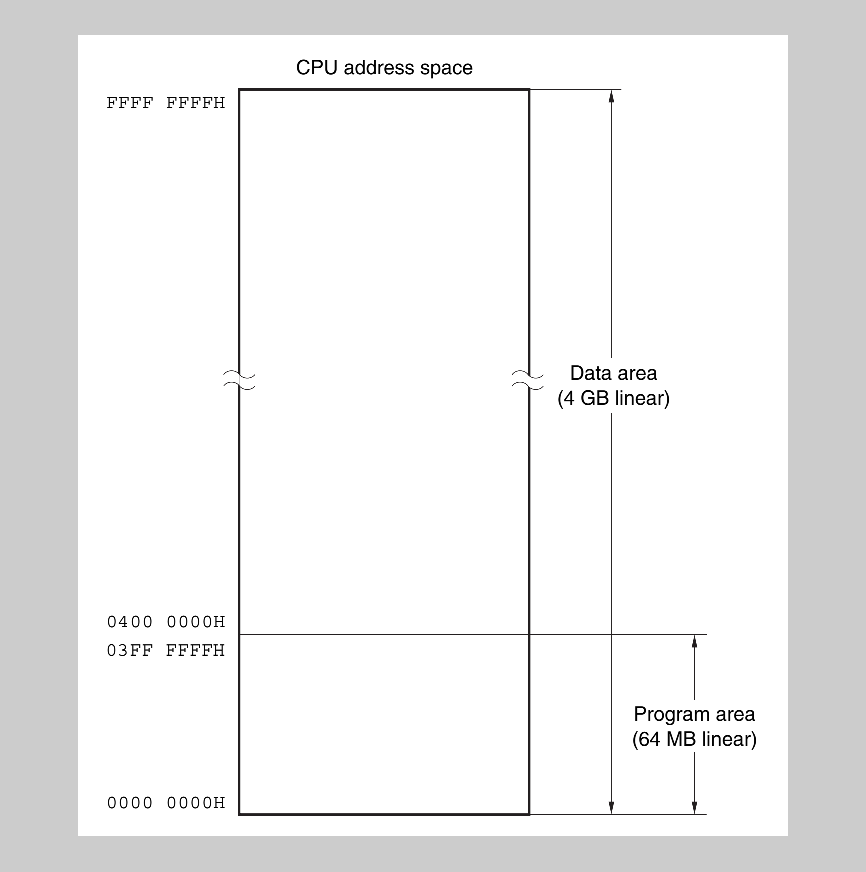


Figure 3-4 CPU address space

(1) Wrap-around of data space

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

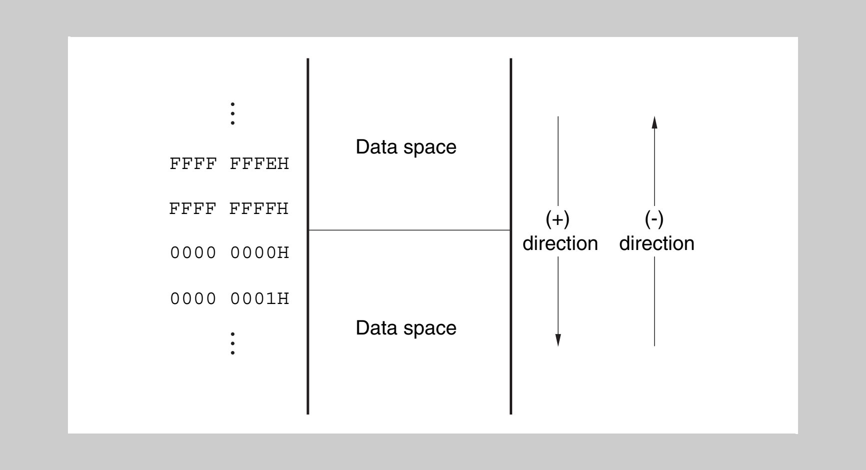


Figure 3-5 Wrap-around of data space

(2) Wrap-around of program space

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

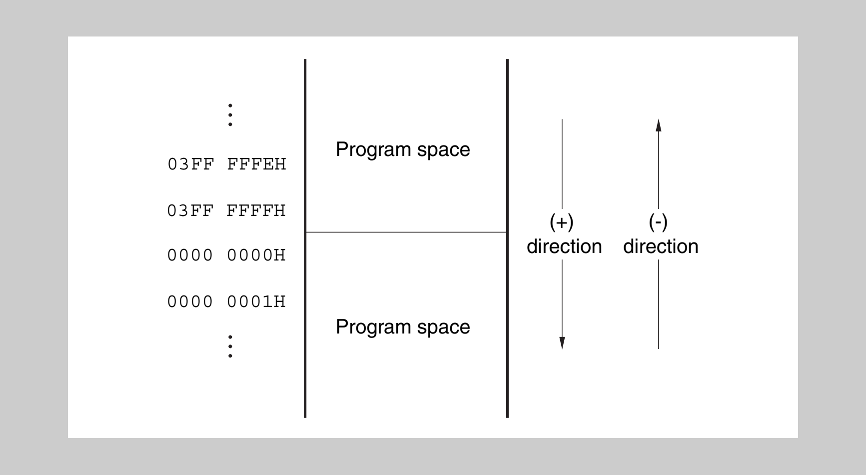


Figure 3-6 Wrap-around of program space

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

3.5 Memory

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

3.5.1 Memory areas

The internal memory of the CPU provides several areas:

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

The areas are briefly described below.

(1) Internal VFB flash area

Table 3-11 summarizes the size and addresses of the ROM and flash memories, which are accessible via the VFB (V850 Fetch Bus).

Table 3-11 VFB ROM and flash memory

Device	ROM	Flash	Address range
µPD703420	128 KB	–	0000 0000_H to 0001 $FFFF_H$
µPD70F3420	–	128 KB	0000 0000_H to 0001 $FFFF_H$
µPD703421	256 KB	–	0000 0000_H to 0003 $FFFF_H$
µPD70F3421	–	256 KB	0000 0000_H to 0003 $FFFF_H$
µPD703422	384 KB	–	0000 0000_H to 0005 $FFFF_H$
µPD70F3422	–	384 KB	0000 0000_H to 0005 $FFFF_H$
µPD70F3423	–	512 KB	0000 0000_H to 0007 $FFFF_H$
µPD70F3424	–	512 KB	0000 0000_H to 0007 $FFFF_H$
µPD70F3425	–	1 MB	0000 0000_H to 000F $FFFF_H$
µPD70F3426	–	1 MB	0000 0000_H to 000F $FFFF_H$
µPD70F3427	–	1 MB	0000 0000_H to 000F $FFFF_H$

(2) Internal VDB RAM area

After reset

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

Table 3-12 summarizes the VDB (V850 Data Bus) RAM blocks compilation and their address assignment.

Table 3-12 Internal VDB RAM areas

Device	RAM size	Block		
		Number	Size	Address
µPD70(F)3420	6 KB	0	4 KB	03FF 0000$_H$ – 03FF 0FFF$_H$
		1	2 KB	03FF 1000$_H$ – 03FF 17FF$_H$
µPD703421	12 KB	0	4 KB	03FF 0000$_H$ – 03FF 0FFF$_H$
		1	4 KB	03FF 1000$_H$ – 03FF 1FFF$_H$
		2	4 KB	03FF 2000$_H$ – 03FF 2FFF$_H$
µPD70F3421	12 KB	0	4 KB	03FF 0000$_H$ – 03FF 0FFF$_H$
		1	8 KB	03FF 1000$_H$ – 03FF 2FFF$_H$
µPD70(F)3422	16 KB	0	8 KB	03FF 0000$_H$ – 03FF 1FFF$_H$
		1	8 KB	03FF 2000$_H$ – 03FF 3FFF$_H$
µPD70F3423	20 KB	0	8 KB	03FF 0000$_H$ – 03FF 1FFF$_H$
		1	8 KB	03FF 2000$_H$ – 03FF 3FFF$_H$
		2	4 KB	03FF 4000$_H$ – 03FF 4FFF$_H$
µPD70F3424	24 KB	0	8 KB	03FF 0000$_H$ – 03FF 1FFF$_H$
		1	8 KB	03FF 2000$_H$ – 03FF 3FFF$_H$
		2	8 KB	03FF 4000$_H$ – 03FF 5FFF$_H$
µPD70F3425[a]	32 KB	0	16 KB	03FF 0000$_H$ – 03FF 3FFF$_H$
		1	16 KB	03FF 4000$_H$ – 03FF 7FFF$_H$
µPD70F3426 µPD70F3427	60 KB	0	16 KB	03FF 0000$_H$ – 03FF 3FFF$_H$
		1	16 KB	03FF 4000$_H$ – 03FF 7FFF$_H$
		2	16 KB	03FF 8000$_H$ – 03FF BFFF$_H$
		3	12 KB	03FF C000$_H$ – 03FF EFFF$_H$

[a] The µPD70F3425's 32 KB RAM area 03FF 0000$_H$ to 03FF 7FFF$_H$ is mirrored to the subsequent area 03FF 8000$_H$ to 03FF FFFF$_H$. Since the upper 4 KB 03FF F000$_H$ to 03FF FFFF$_H$ is used to access the fixed peripheral I/O area, the RAM mirror must not be used to access the RAM.

Note that the internal firmware, which is processed after reset, uses some RAM (refer to *"General reset performance" on page 861*).

(3) Internal VSB flash area (µPD70F3426 only)

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

Table 3-13 Internal VSB flash memory

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

(4) Internal VSB RAM area (µPD70F3426 only)

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

Table 3-14 Internal VSB RAM

Device	RAM size	Block		
		Number	Size	Address
µPD70F3426	24 KB	0	12 KB	0060 0000_H – 0060 $2FFF_H$
		1	12 KB	0060 3000_H – 0060 $5FFF_H$

(5) Fixed peripheral I/O area

The 4 KB area between addresses 03FF $F000_H$ and 03FF $FFFF_H$ is provided as the fixed peripheral I/O area. Accesses to these addresses are passed over to the NPB bus (internal bus).

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

- All registers of peripheral functions
- Registers of timers
- Configuration registers of interrupt, 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 891*.

Note

1. Because the physical address space covers 64 MB, the address bits A[31:26] are not considered. Thus, this address space can also be addressed via the area FFFF 0000_H to FFFF $FFFF_H$. This has the advantage that the area can be indirectly addressed by an offset and the zero base r0.
   Therefore, in this manual, all addresses of peripheral I/O registers in the 4 KB peripheral I/O area are given in the range FFFF $F000_H$ to FFFF $FFFF_H$ instead of 03FF $F000_H$ to 03FF $FFFF_H$.
2. The *fixed* peripheral I/O area is mirrored to the upper 4 KB of the *programmable* peripheral I/O area PPA - regardless of the base address of the PPA. If data is written to one area, it appears also in the other area.
3. Program fetches cannot be executed from any peripheral I/O area.
4. Word registers, that means 32-bit registers, are accessed in two half word accesses. The lower two address bits are ignored.

5. For registers in which byte access is possible, if half word access is executed:
   - During read operation: The higher 8 bits become undefined.
   - During write operation: The lower 8 bits of data are written to the register.

Caution

1. Addresses that are not defined as registers are reserved for future expansion. If these addresses are accessed, the operation is undefined and not guaranteed.
2. For DMA transfer, the fixed peripheral I/O area 03FF $F000_H$ to 03FF $FFFF_H$ cannot be specified as the source/destination address. Be sure to use the RAM area 0FFF $F000_H$ to 0FFF $FFFF_H$ for source/destination address of DMA transfer.

(6) Programmable peripheral I/O area

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

See *“Bus and Memory Control (BCU, MEMC)“ on page 249* for details.

(7) External memory area (µPD70F3427 only)

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

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

For access to external memory, see *“Bus and Memory Control (BCU, MEMC)“ on page 249*.

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

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

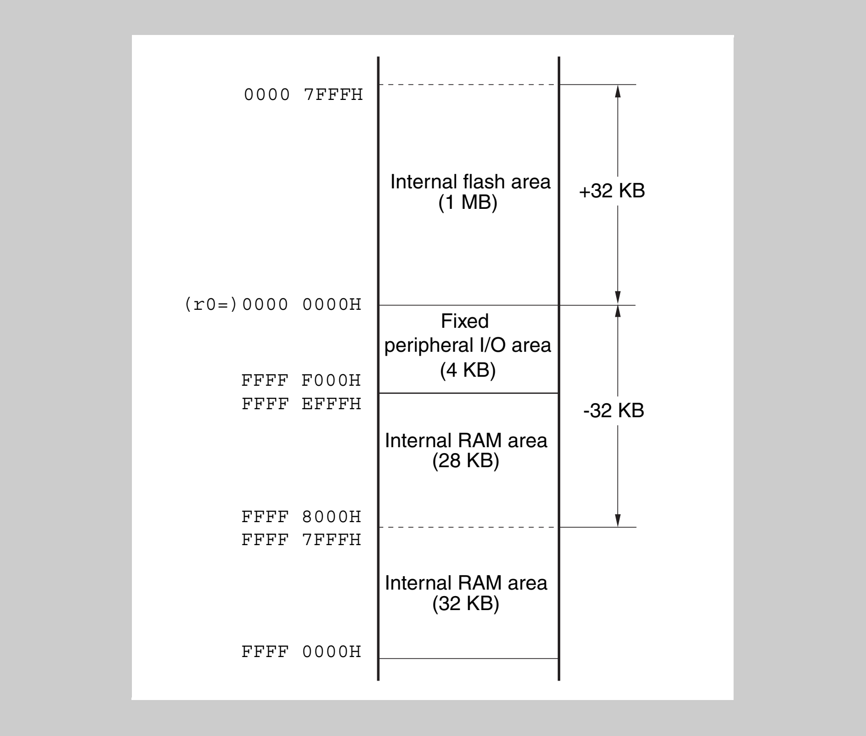


Figure 3-7 Example application of wrap-around

3.6 Write Protected Registers

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

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

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

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

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

Table 3-15 Overview of write protected registers

Write protected register	**Shortcut**	**Corresponding write enable register**	**Shortcut**	**Status register**	**For details see**
Clock control register	CKC	Peripheral command register	PHCMD	PHS	*“Clock Generator“ on page 129*
Watchdog timer clock control register	WCC				
Processor clock control register	PCC				
Watch Timer clock control register	TCC				
SPCLK control register	SCC				
FOUTCLK control register	FCC				
I^{2}C clock control register	ICC				
Main oscillator clock monitor mode register	CLMM	Main oscillator clock monitor command protection register	PRCMDCMM	—	*“Clock Generator“ on page 129*
Sub oscillator clock monitor mode register	CLMS	Sub oscillator clock monitor command protection register	PRCMDCMS		
Power save control register	PSC	Command register	PRCMD	–	*“Clock Generator“ on page 129*
Self-programming enable control register[a]	SELFEN	Sequence protect register	SELFENP	–	*“Flash Memory“ on page 229*
Watchdog timer frequency select register	WDCS	Watchdog timer security register	WCMD	WPHS	*“Watchdog Timer (WDT)“ on page 497*
Watchdog timer mode register	WDTM				
N-Wire security disable control register	RSUDISC	RSUDISC write protection register	RSUDISCP	–	*“On-Chip Debug Unit“ on page 877*

a) Flash memory devices only

Example Start the Watchdog Timer

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

```
do {
     _WPRERR = 0;

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

} while (_WPRERR != 0)
```

Note

1. Make sure that the compiler generates two consecutive assembler “store” instructions to WCMD and WDTM from the associated C statements.
2. Special care must be taken when writing to registers PCS and PRCMD. Please refer to *“Clock Generator“ on page 129* for details.

Since any action between writing to a write enable register and writing to a protected register destroys this sequence, the effects of interrupts 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.

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

3.7 Instructions and Data Access Times

The below *Table 3-16* and *Table 3-17* list the instruction execution and data access cycles, required when accessing instructions or data in VFB flash/ROM, and VDB RAM and VSB flash/RAM.

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

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

The memory access time in a real application is deterministic, but can hardly be predicted, as this heavily depends on the status of the microcontroller and its components, the program flow and concurrent processes, like DMA transfers, interrupts, accesses to peripheral registers via the NPB, etc.
Thus the figures in the below tables assume
- all busses (VFB, VDB, VSB, NPB) are not occupied, i.e. collision with other bus traffic is excluded
- 32-bit instruction/data accesses to word-aligned - that means 32-bit aligned - addresses
- data is not accessed via the same bus as the instruction is fetched from

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

Instruction execution

The given numbers of cycles in *Table 3-16* describe the time required to execute a single-cycle instruction, fetched from the respective memory:
- Consecutive access
  describes the number of cycles required to fetch instructions from the memory on consecutive addresses.
- Random access
  describes the number of cycles required to access the memory in case instructions are accessed on random, i.e. non-consecutive, addresses. In case of instruction flow branches a CPU’s pipeline break occurs and an additional cycle is required to refill the pipeline. The table figures include this cycle.

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

Table 3-16 Single-cycle instructions execution times in CPU clock cycles

Memory	Access type	µPD70F3427	µPD70F3424 µPD70F3425	µPD70F3426	µPD70F3421 µPD70F3422 µPD70F3423	µPD703420 µPD703421 µPD703422
VFB flash	Consecutive	1	1	1	1	–
	Random	3[a]	3[a]	3[a]	1[a]	–
VFB ROM	Consecutive	–	–	–	–	1
	Random	–	–	–	–	1[a]
VDB RAM	Consecutive	1	1	1	1	1
	Random	1[a]	1[a]	1[a]	1[a]	1[a]
VSB flash	Consecutive	–	–	2	–	–
	Random	–	–	5[a]	–	–
VSB RAM	Consecutive	–	–	2	–	–
	Random	–	–	3[a]	–	–

a) These values include the additional clock cycle, cause by the CPU’s pipeline break

Data access The given numbers of cycles in *Table 3-17* describe the time additionally required when an instruction accesses data in the respective memory.

Note that data accesses are always random accesses.

Table 3-17 Additional time for data accesses in CPU clock cycles

Data access memory	Instruction code fetch bus	µPD70F3427	µPD70F3424 µPD70F3425	µPD70F3426	µPD70F3421 µPD70F3422 µPD70F3423	µPD703420 µPD703421 µPD703422
VFB flash	VFB	4	4	4	1	–
VFB ROM	VFB	–	–	–	–	1
VDB RAM	VFB/VSB	0	0	0	0	0
VSB flash	VFB	–	–	4	–	–
VSB RAM	VFB	–	–	• 1 (single access) • 3 (multiple access)	–	–

Chapter 4 Clock Generator

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

4.1 Overview

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

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

Features summary Special features of the clock generator are:

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

4.1.1 Description

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

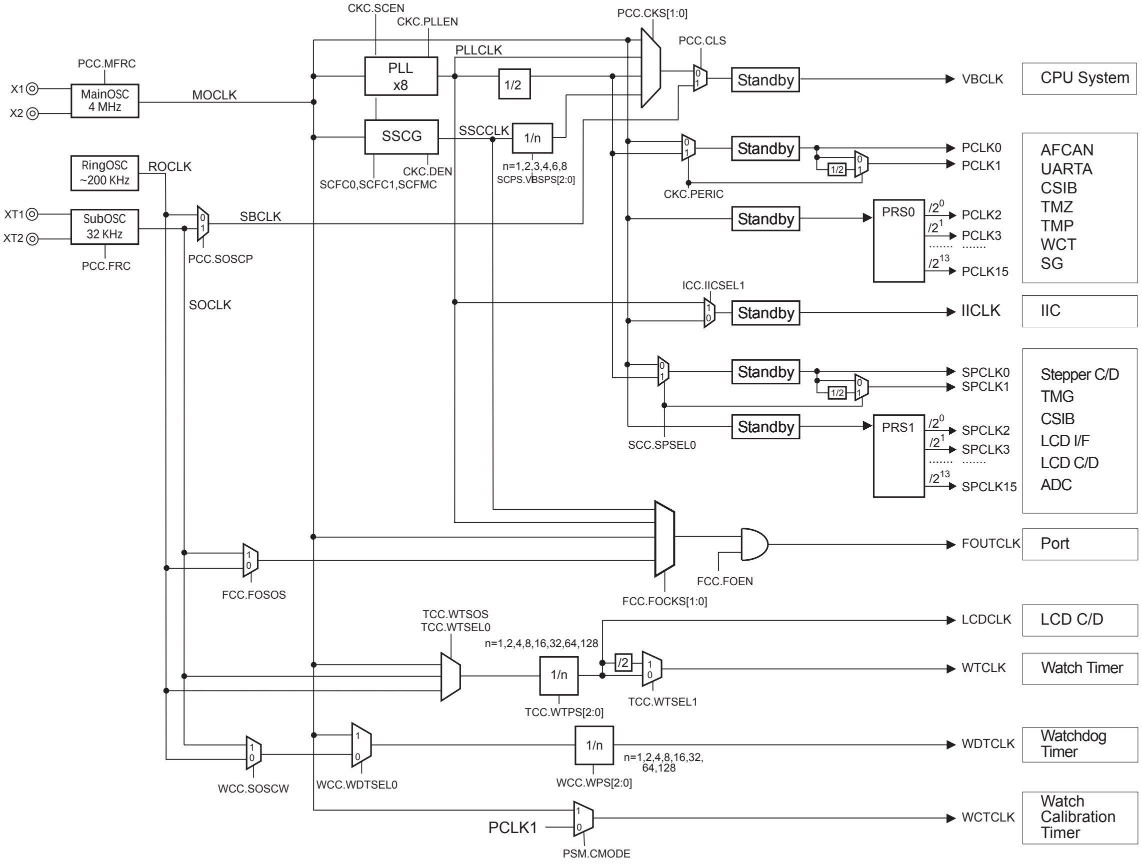


Figure 4-1 Block diagram of the Clock Generator

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

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

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

The SSCG can supply the CPU system.

(1) CPU clocks

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

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

Table 4-1 Clock sources and frequencies for the CPU

Clock source	Frequency	Description
Ring osc	~240 KHz	Default clock source after reset release. Selectable as clock source for Sub-WATCH mode release.
Sub osc	32 KHz	Selectable as clock source for Sub-WATCH mode release.
Main osc	4 MHz	Always selected after power save mode release except on Sub-WATCH mode release or default clock setting.[a] On Sub-WATCH mode release or default clock setting, main or sub oscillator can be selected.
PLL	16 MHz	$f_{main} \times 4$ can be selected for CPU clock supply.
	32 MHz	$f_{main} \times 8$ can be selected for CPU clock supply.
SSCG	8 MHz	$f_{main} \times 12/6$[b] can be selected for CPU clock supply.
	16 MHz	$f_{main} \times 16/4$[b] can be selected for CPU clock supply.
	24 MHz	$f_{main} \times 12/2$[b] can be selected for CPU clock supply.
	32 MHz	$f_{main} \times 16/2$[b] can be selected for CPU clock supply.
	48 MHz	$f_{main} \times 12/1$[b] can be selected for CPU clock supply.

a) See also *“CPU operation after power save mode release” on page 181*

b) Multiplication is performed by the SSCG, the division by the SSCG post scaler.

(2) Peripheral clocks

The right-hand side of *Figure 4-1 on page 130* shows how the clocks for the peripheral modules are generated and distributed.

PCLK clocks The PCLK clocks supply following peripherals: the CAN Controllers CAN, the UARTs, the Timers Z, the Watch Calibration Timer, and the Clocked Serial Interfaces CSIB.

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

SPCLK clocks The SPCLK clocks supply following peripherals: Stepper Motor Controller/Driver, the Timer G, the Sound Generator, the Clocked Serial Interfaces CSIB, the LCD Bus I/F and Controller/Driver, and A/D Converter ADC.

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

IICLK clock The clock IICLK for the I^2C interface is supplied by the main oscillator or the PLL.

(3) Special clocks

The figure shows also some special clock signals. These are dedicated clocks for the LCD Controller/Driver, Watch Timer, Watchdog Timer, and Watch

Calibration Timer. These clocks are directly derived from the oscillators and bypass the PLLs.

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

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

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

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

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

WDTCLK This is the clock for the Watchdog Timer that is used for recovering from a system deadlock. WDTCLK is available (and hence the Watchdog Timer running) as long as the chosen clock source is active.

(4) Stand-by control

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

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

CPU system clock, PCLK, SPCLK, IICLK.

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

FOUTCLK, LCDCLK, WTCLK, WCTCLK.

4.1.2 Clock monitors

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

The monitors require that the built-in ring oscillator is active. For details see *“Operation of the Clock Monitors” on page 185*.

4.1.3 Power save modes overview

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

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

For details, please refer to *“Power save modes description” on page 167* and the register descriptions.

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

This mode is entered by executing the HALT instruction.

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

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

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

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

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

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

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

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

STOP mode In this mode, the entire system stops.

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

4.1.4 Start conditions

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

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

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

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

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

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

Table 4-2 Clock Generator status after reset release

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

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

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

4.1.5 Start-up guideline

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

After that, the user's software will typically:

1. Ensure that the main oscillator has stabilized (check CGSTAT.OSCSTAT).
2. Switch the source of LCDCLK/WTCLK and WDTCLK to main oscillator (if desired).
3. Start the PLL (set CKC.PLLEN) and wait until the PLL has stabilized (refer to the Electrical Target Specification).
4. If the SSCG is going to be used:
   Write SSCG registers to set up the SSCG. This is only possible when the SSCG is switched off.
   Start the SSCG (set CKC.SCEN) and wait until the SSCG has stabilized (refer to the Electrical Target Specification).
   Set up the SSCG post clock divider by SCPS.VBSPS[2:0].
5. Write the PCC register to specify the SSCG as the clock source for the CPU.
6. Set up the clock sources for the peripherals according to application requirements.
7. The default value of following registers must be changed after reset:
   - WCC.bit1 = 1 (refer to *"Control registers for peripheral clocks" on page 150*)
   - ADA0M1.bit1 = 1 (refer to *"ADC Registers" on page 773*)

4.2 Clock Generator Registers

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

Table 4-3 Clock Generator register overview

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

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

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

The subsequent register descriptions are grouped as follows:

- **General Clock Generator Registers:**
  - *"CKC - Clock Generator control register" on page 138*
  - *"CGSTAT - Clock Generator status register" on page 139*
  - *"PHCMD - Command protection register" on page 140*
  - *"PHS - Peripheral status register" on page 141*
  - *"PCC - Processor clock control register" on page 142*
- **SSCG Control Registers:**
  - *"SCFC0 - SSCG frequency control register 0" on page 145*
  - *"SCFC1 - SSCG frequency control register 1" on page 146*
  - *"SCFMC - SSCG frequency modulation control register" on page 147*
  - *"SCPS - SSCG post scaler control register" on page 149*
- **Control Registers for Peripheral Clocks:**
  - *"WCC - Watchdog Timer clock control register" on page 150*
  - *"TCC - Watch Timer clock control register" on page 152*
  - *"SCC - SPCLK control register" on page 154*
  - *"FCC - FOUTCLK control register" on page 155*
  - *"ICC - IIC clock control register" on page 156*
- **Control Registers for Power Save Modes:**
  - *"PSM - Power save mode register" on page 157*
  - *"PSC - Power save control register" on page 159*
  - *"PRCMD - PSC write protection register" on page 160*
  - *"STBCTL- Stand-by control register" on page 161*
  - *"STBCTLP - Stand-by control protection register" on page 162*
- **Clock Monitor Registers:**
  - *"CLMM - Main oscillator clock monitor mode register" on page 163*
  - *"PRCMDCMM - CLMM write protection register" on page 164*
  - *"CLMS - Sub oscillator clock monitor register" on page 165*
  - *"PRCMDCMS - CLMS write protection register" on page 165*
  - *"CLMCS - Sub oscillator clock monitor control register" on page 166*

4.2.1 General clock generator registers

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

(1) CKC - Clock Generator control register

The 8-bit CKC register controls the clock management.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *“PHCMD - Command protection register” on page 140* for details.

Address FFFF $F822_H$.

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

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

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

Table 4-4 **CKC register contents**

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

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

(2) CGSTAT - Clock Generator status register

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

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

Address FFFF F824$_H$.

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

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

Table 4-5 CGSTAT register contents

Bit position	Bit name	Function
7	CMPLPSM	Completed power save mode entry: 0: Power save mode configuration not completed. 1: Power save mode configuration completed. This bit is cleared when the clock generator has accepted a power save mode request. However if CGSTAT.CMLPSM was already 0 before a power save mode request it can not be used as an indicator that the clock generator has accepted this power save mode request. This bit is set when the clock generator has completely set up it's power save mode configuration, i.e. all registers are set up, PLL and SSCG are switched off. However if CGSTAT.CMLPSM was already 1 before a power save mode request it can not be used as the only indicator that the clock generator has completed power save mode configuration. If the clock generator has not accepted a power save mode request this bit remains unchanged. Refer also to "*"CPU operation after power save mode release" on page 181*".
1	OSCSTAT	Main oscillator status indicator (determined by counter): 0: Main oscillator has not settled. 1: Main oscillator has stabilized. The OSCSTAT flag is cleared whenever the main oscillator is switched to stand-by mode due to entering the Sub-WATCH or STOP mode. After the main oscillator is restarted, the oscillation stabilization counter will count up from 0 to 40.960 (approx. 10ms @ 4 MHz) to assure stable oscillator operation. When the oscillation stabilization counter reaches 40.960, the counter is stopped, and the OSCSTAT flag is set.

(3) PHCMD - Command protection register

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

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

Address FFFF $F800_H$.

Initial Value The contents of this register is undefined.

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

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

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

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

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

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

The following registers are protected by PHCMD:

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

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

(4) PHS - Peripheral status register

The 8-bit PHS register indicates the status of a write attempt to a register protected by PHCMD (see also *"PHCMD - Command protection register" on page 140*).

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

Address FFFF F802$_H$.

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

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

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

Table 4-6 PHS register contents

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

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

(5) PCC - Processor clock control register

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

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

Writing to this register is protected by a special sequence of instructions. Please refer to *"PHCMD - Command protection register" on page 140* for details.

Address FFFF $F828_H$.

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

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

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

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

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

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

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

Note

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

Write protection

Write protection of this register is achieved in two ways:

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

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

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

4.2.2 SSCG control registers

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

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

Reconfiguration of SSCG registers

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

- the SSCG is disabled by CKC.SCEN = 0
- the SSCG is safely switched off after a power save mode wake-up. Refer to *"CPU operation after power save mode release" on page 181* for a procedure to ensure that the SSCG is switched off after wake-up.

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

(1) SCFC0 - SSCG frequency control register 0

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

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

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

Address FFFF $F82C_H$.

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

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

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

Table 4-8 SCFC0 register contents

Bit position	Bit name	Function
6 to 5	SCFC0[6:5]	Must be set to 01_B
4 to 3	SCFC0[4:3]	Must be set according to *Table 4-9*
2 to 0	SCFC0[2:0]	Determines the divisor m

Note

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

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

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

where:

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

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

Table 4-9 Supported settings of N (n) and M (m)

$f_{SSCGmax}$	M (m)	N (n)	SCFC0	SCFC1
48 MHz	4 (3)	96 (95)	$2B_H$	DF_H
64 MHz[a]	3 (2)	96 (95)	32_H	DF_H

[a] In this mode the 64 MHz SSCG output frequency has to be devided by the SSCG post scaler. Thus set SCPS = 21_H for 32 MHz or SCPS = 23_H for 16 MHz operation.

(2) SCFC1 - SSCG frequency control register 1

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

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

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

Address FFFF $F82E_H$.

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

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

Table 4-10 SCFC1 register contents

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

Note

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

(3) SCFMC - SSCG frequency modulation control register

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

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

Address FFFF $F82A_H$.

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

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

Table 4-11 SCFMC register contents

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

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

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

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

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

Example If:

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

Then:

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

(4) SCPS - SSCG post scaler control register

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

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

Address FFFF F830$_H$.

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

7	6	5	4	3	2	1	0
0	0[a]	1[a]	0[a]	0	VBSPS2	VBSPS1	VBSPS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

a) These bits must not be altered.

Table 4-12 SCPS register contents

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

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

4.2.3 Control registers for peripheral clocks

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

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

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

(1) WCC - Watchdog Timer clock control register

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

Writing to this register is protected by a special sequence of instructions. Please refer to *"PHCMD - Command protection register" on page 140* for details.

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

Address FFFF $F826_H$.

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

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

Caution Bit 1 of WCC must be set to "1" after reset and must not be altered afterwards.

Table 4-13 WCC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>SOSTP</td><td>Sub oscillator STOP mode control
1: Sub oscillator will stop when STOP mode is entered.
0: Sub oscillator will not stop when STOP mode is entered.</td></tr>
<tr><td>6 to 4</td><td>WPS[2:0]</td><td>WDT clock divider selection:
<table>
<tr><th>WPS2</th><th>WPS1</th><th>WPS0</th><th>Clock divider setting</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1 / 2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1 / 4</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1 / 8</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1 / 16</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1 / 32</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1 / 64</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1 / 128</td></tr>
</table></td></tr>
<tr><td>3</td><td>ROSTP</td><td>Ring oscillator stop control:
1: Ring oscillator stops if WATCH, Sub-WATCH or STOP mode is entered
0: Ring oscillator always operates</td></tr>
<tr><td>2, 0</td><td>SOSCW,
WDTSEL0</td><td>Watchdog Timer clock source selection:
<table>
<tr><th>SOSCW</th><th>WDTSEL0</th><th>WDT clock source</th></tr>
<tr><td>0</td><td>0</td><td>Ring oscillator</td></tr>
<tr><td>1</td><td>0</td><td>Sub oscillator</td></tr>
<tr><td>0</td><td>1</td><td>Main oscillator</td></tr>
<tr><td>1</td><td>1</td><td>Setting prohibited</td></tr>
</table>
By default, the sub oscillator is disabled in STOP mode (see bit SOSTP). If SOSTP is 1, choose main or ring oscillator before entering STOP mode.

Caution: Do not specify the sub oscillator, if the sub oscillator is not enabled or not connected.</td></tr>
</table>

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

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

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

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

(2) TCC - Watch Timer clock control register

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

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

Writing to this register is protected by a special sequence of instructions. Please refer to *“PHCMD - Command protection register” on page 140* for details.

Address FFFF $F836_H$.

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

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

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

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

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>6 to 4</td><td>WTPS[2:0]</td><td>LCDCLK clock divider selection:
<table>
<tr><th>WTPS2</th><th>WTPS1</th><th>WTPS0</th><th>Clock divider setting</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1 / 2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1 / 4</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1 / 8</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1 / 16</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1 / 32</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1 / 64</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1 / 128</td></tr>
</table></td></tr>
<tr><td>1</td><td>WTSEL1</td><td>WTCLK (Watch Timer clock) divider setting:
0: WTCLK = LCDCLK.
1: WTCLK = LCDCLK / 2.</td></tr>
</table>

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

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>2, 0</td><td>WTSOS,
WTSEL0</td><td>Clock source for Watch Timer and LCD controller:
<table>
<tr><th>WTSOS</th><th>WTSEL0</th><th>Clock source</th></tr>
<tr><td>0</td><td>0</td><td>Ring oscillator</td></tr>
<tr><td>1</td><td>0</td><td>Sub oscillator</td></tr>
<tr><td>0</td><td>1</td><td>Main oscillator</td></tr>
<tr><td>1</td><td>1</td><td>Setting prohibited</td></tr>
</table>
By default, the sub oscillator is disabled in STOP mode (see bit WCC.SOSTP). If WCC.SOSTP is 1, choose main or ring oscillator before entering STOP mode.
Caution: Do not specify the sub oscillator, if the sub oscillator is not enabled or not connected.</td></tr>
</table>

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

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

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

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

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

(3) SCC - SPCLK control register

The 8-bit SCC register selects the SPCLK sources.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *“PHCMD - Command protection register” on page 140* for details.

Address FFFF F832$_H$.

Initial Value 00$_H$. The register is initialized by entering WATCH, Sub-WATCH, or STOP mode

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0[a]	SPSEL0
R	R	R	R	R	R	R/W	R/W

a) This bit must not be altered.

Table 4-15 SCC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>0</td><td>SPSEL0</td><td rowspan="6">Source selection for generating the SPCLK clocks:<table><tr><th rowspan="2">SPSEL0</th><th colspan="3">Clock sources</th></tr><tr><th>SPCLK0</th><th>SPCLK1</th><th>SPCLK2</th></tr><tr><td>0</td><td>Main osc</td><td>Main osc</td><td>Main osc</td></tr><tr><td>1</td><td>PLL / 2</td><td>PLL / 4</td><td>Main osc</td></tr></table></td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
</table>

Note

1. “Main osc” is the clock provided by the main oscillator.
2. “PLL” is the clock provided by the PLL.

(4) FCC - FOUTCLK control register

The 8-bit FCC register configures the output clock FOUTCLK that can be used for external devices.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *“PHCMD - Command protection register” on page 140* for details.

Address FFFF $F834_H$.

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

7	6	5	4	3	2	1	0
FOEN	0[a]	0[a]	0[a]	0	FOSOS	FOCKS1	FOCKS0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W

[a] These bits must not be altered.

Table 4-16 FCC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>FOEN</td><td>Output clock FOUTCLK enable:
0: FOUTCLK is disabled.
1: FOUTCLK is enabled.</td></tr>
<tr><td>2 to 0</td><td>FOSOS,
FOCKS[1:0]</td><td>Clock source selection for FOUTCLK:
<table>
<tr><th>FOSOS</th><th>FOCKS1</th><th>FOCKS0</th><th>Clock source</th></tr>
<tr><td>X</td><td>0</td><td>0</td><td>Main oscillator</td></tr>
<tr><td>X</td><td>0</td><td>1</td><td>SSCG</td></tr>
<tr><td>X</td><td>1</td><td>0</td><td>PLL</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Ring oscillator</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Sub oscillator</td></tr>
</table>
Caution: Do not specify the sub oscillator, if the sub oscillator is not enabled or not connected.</td></tr>
</table>

Note

1. FOUTCLK is not influenced by stand-by modes of the microcontroller. It runs as long as it is enabled and the selected clock source operates. Application software must stop FOUTCLK by clearing the FOEN bit to minimize power consumption in stand-by modes.
2. There is an upper frequency limit for the output buffer of the FOUTCLK function. Do not select a frequency higher than the maximum output buffer frequency. Please refer to the Electrical Target Specification for the frequency limit.

(5) ICC - IIC clock control register

The 8-bit ICC register determines the I^2C clock source for IICLK.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *"PHCMD - Command protection register" on page 140* for details.

Address FFFF $F838_H$.

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

7	6	5	4	3	2	1	0
0	0[a]	0[a]	0[a]	0	0	IICSEL1	0[a]
R[b]	R/W	R/W	R/W	R[b]	R[b]	R/W	R/W

[a] These bits must not be altered.
[b] These bits may be written, but write is ignored.

Table 4-17 ICC register contents

Bit position	Bit name	Function
1	IICSEL1	Clock source for IICLK: IICSEL1 | Clock source 0 | Main oscillator 1 | PLL

IICSEL1	Clock source
0	Main oscillator
1	PLL

Note On release of WATCH, Sub-WATCH and STOP mode, IICSEL1 is cleared—the main oscillator is selected as the I^2C clock source.

Pay attention if PSM.OSCDIS = 1 before entering any of the above power save modes, because the main oscillator will be disabled. Therefore the I^2C interface will have no clock supply after power save mode release.

4.2.4 Control registers for power save modes

The registers described in this section control the begin and end of the power save modes IDLE, WATCH, Sub-WATCH, and STOP.

Please refer to *“Power save mode activation” on page 179* for instructions and an example on how to enter a power save mode.

(1) PSM - Power save mode register

The 8-bit PSM register specifies the power save mode and controls the clock generation after reset and Sub-WATCH mode release. In addition, it specifies the source of the Watch Calibration Timer clock WCTCLK.

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

Address FFFF F820$_H$.

Initial Value 08_H. The register is initialized by any reset.
Since the main oscillator is started by the internal firmware are reset, PSM enters the user’s program with the setting 00_H.

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

Table 4-18 PSM register contents

Bit position	Bit name	Function
6	CMODE	Watch Calibration Timer clock selection: 0: PCLK1. 1: Main oscillator.
3	OSCDIS	Main oscillator disable/enable control during and after power save mode: 0: Main oscillator enabled. 1: Main oscillator disabled. **Caution:** If OSCDIS is set to 1, the main oscillator clock supply for the Watch Timer and the LCD Controller/Driver are stopped immediately. Thus these function stop their operation immediately as well, when the main oscillator is used as the clock source. OSCDIS determines also the behaviour of the main oscillator during and after power save mode. The effect of this bit differs, depending on the power save mode. • Sub-WATCH mode During Sub-WATCH mode the main oscillator is always stopped. OSCDIS determines whether the main oscillator shall be started and chosen as CPU clock source or should remain stopped after Sub-WATCH mode release. 0: Main oscillator enable. The main oscillator is started after Sub-WATCH mode release and the CPU is supplied with the main oscillator clock, after the oscillation stabilization time has elapsed. 1: Main oscillator disable. The main oscillator remains stopped after Sub-WATCH release. The CPU is supplied with the selected sub clock—either sub oscillator or ring oscillator (see bit PCC.SOSCP). Since the reset value of OSCDIS is 1 and PCC.SOSCP is 0 the CPU starts always with the ring oscillator clock after reset release. In both cases, the application software must start the main oscillator by clearing the OSCDIS bit. After the oscillator stabilization time has elapsed (see bit CGSTAT.OSCSTAT), the main oscillator can be used as system clock source by setting the PCC register accordingly. • WATCH mode This bit determines whether the main oscillator shall be stopped or remain in operation during WATCH mode. In either case after WATCH mode release the CPU is operating on the main oscillator. 0: Main oscillator enable. The main oscillator is operating during WATCH mode. After WATCH mode release the CPU is supplied with the main oscillator clock. 1: Main oscillator disable. The main oscillator is stopped during WATCH mode. After WATCH mode release the main oscillator is started and the CPU is supplied with the main oscillator clock, after the oscillation stabilization time has elapsed.
1 to 0	PSM[1:0]	Power save mode selection: PSM1 / PSM0 / Power save mode 0 / 0 / IDLE 0 / 1 / STOP 1 / 0 / WATCH 1 / 1 / Sub-WATCH mode (main oscillator shut down) It is not possible to switch to IDLE or WATCH mode when the CPU is operated by a sub clock. If IDLE or WATCH mode is selected during sub clock operation, the Sub-WATCH mode will be entered.

(2) PSC - Power save control register

The 8-bit PSC register is used to enter or leave the power save mode specified in register PSM.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *“PRCMD - PSC write protection register” on page 160* for details.

Address FFFF $F1FE_H$.

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

7	6	5	4	3	2	1	0
0	NMIWDT	NMI0	INTM	0	0	STP	0
R	R/W	R/W	R/W	R	R	R/W	R

Table 4-19 PSC register contents

Bit position	Bit name	Function
6	NMIWDT	Mask for non-maskable interrupt request from WDT: 0: Permit NMIWDT request during power save mode. 1: Prohibit NMIWDT request during power save mode.
5	NMI0	Mask for non-maskable interrupt request 0: 0: Permit external NMI0 request during power save mode 1: Prohibit external NMI0 request during power save mode.
4	INTM	Mask for maskable interrupt request: 0: Permit maskable interrupt requests during power save mode.[a] 1: Prohibit maskable interrupt requests during power save mode.
1	STP	Enter/release power save mode: 0: Power save mode is released. 1: Power save mode is entered.

a) Only dedicated maskable interrupts have wake-up capability, refer to *“Power save modes description” on page 167*.

Note
1. If bits 7, 3, 2, and 0 are not set to 0, proper operation of the controller can not be guaranteed.
2. PSC.STP is automatically cleared when the controller is awakened from power save mode.
3. Entering a power save mode requires some attention, refer to *“Power save mode activation” on page 179*.

(3) PRCMD - PSC write protection register

The 8-bit PRCMD register protects the register PSC from inadvertent write access, so that the system does not stop in case of a program hang-up.

After data has been written to the PRCMD register, the first write access to register PSC is valid. All subsequent write accesses are ignored. Thus, the value of PSC can only be rewritten in a specified sequence, and illegal write access is inhibited.

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

Address FFFF $F1FC_H$

Initial Value The contents of this register is undefined.

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

Caution Before writing to PRCMD, make sure that all DMA channels are disabled. Otherwise, a direct memory access could occur between the write access to PRCMD and the write access to PSC. If that happens, the power save mode may not be entered.

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

(4) STBCTL- Stand-by control register

The 8-bit STBCTL register is used to control the stand-by function of the voltage regulators.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *"STBCTLP - Stand-by control protection register" on page 162* for details.

Address FFFF FCA2$_H$.

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

7	6	5	4	3	2	1	0
0	0	0	0	0	0	STYCD	STBYMD
R	R	R	R	R	R	R/W	R/W

Table 4-20 STBCTL register contents

Bit position	Bit name	Function
1	STBYCD	Enable stand-by function of VDD50 and VDD51 voltage regulators: 0: Stand-by function disabled 1: Stand-by function enabled
2	STBYMD	Enable stand-by function of VDD52 voltage regulator: 0: Stand-by function disabled 1: Stand-by function enabled

In order to reduce the power consumption during power save modes the stand-by function of the voltage regulators should be enabled during the initialization.

If a dedicated microcontroller does not include any of the voltage regulators dedicated to the controls bit STBCTL.STBYCD andSTBCTL.STBYMD, the status of the control bit has no function. Thus the initialization for enabling the stand-by functions by STBCTL = 03$_H$ can be retained. For further details concerning voltage regulators refer to *"Power Supply Scheme" on page 855*.

(5) STBCTLP - Stand-by control protection register

The 8-bit STBCTLP register protects the register STBCTL from inadvertent write access.

After data has been written to the STBCTLP register, the first write access to register STBCTL is valid. All subsequent write accesses are ignored. Thus, the value of STBCTL can only be rewritten in a specified sequence, and illegal write access is inhibited.

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

Address FFFF $FCAA_H$

Initial Value The contents of this register is undefined.

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

4.2.5 Clock monitor registers

The following registers are used to control the monitor circuits of the main oscillator clock and the sub oscillator clock.

Please refer to *"Operation of the Clock Monitors" on page 185* for supplementary information.

(1) CLMM - Main oscillator clock monitor mode register

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

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

Writing to this register is protected by a special sequence of instructions. Please refer to *"PRCMDCMM - CLMM write protection register" on page 164* for details.

Address FFFF F870$_{H}$.

Initial Value 00$_{H}$. This register is cleared by any reset.

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

Table 4-21 CLMM register contents

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

Note CLMM.CLMEM can be set at any time. However, the clock monitor is only activated after the main oscillator has stabilized, indicated by CGSTAT.OSCSTAT = 1.

(2) PRCMDCMM - CLMM write protection register

The 8-bit PRCMDCMM register protects the register CLMM from inadvertent write access, so that the system does not stop in case of a program hang-up.

After data has been written to the PRCMDCMM register, the first write access to register CLMM is valid. All subsequent write accesses are ignored. Thus, the value of CLMM can only be rewritten in a specified sequence, and illegal write access is inhibited.

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

Address FFFF FCB0$_H$

Initial Value The contents of this register is undefined.

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

After writing to the PRCMDCMM register, you are permitted to write once to CLMM. The write access to CLMM must happen with the immediately following instruction.

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

(3) CLMS - Sub oscillator clock monitor register

The 8-bit CLMS register is used to enable the monitor for the sub oscillator clock.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *“PRCMDCMS - CLMS write protection register” on page 165* for details.

Address FFFF F878$_H$.

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

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

Table 4-22 CLMS register contents

Bit position	Bit name	Function
0	CLMES	Clock monitor enable: 0: Clock monitor for sub oscillator disabled. 1: Clock monitor for sub oscillator enabled. This bit can only be cleared by reset.

Note Setting CLMS.CLMES to 1 does not start the sub oscillator clock monitor. To start the clock monitor CLMCS.CMRT has to be set to 1 afterwards.

CLMCS.CMRT must not be set before the sub oscillator has stabilized.

(4) PRCMDCMS - CLMS write protection register

The 8-bit PRCMDCMS register protects the register CLMS from inadvertent write access, so that the system does not stop in case of a program hang-up.

After data has been written to the PRCMDCMS register, the first write access to register CLMS is valid. All subsequent write accesses are ignored. Thus, the value of CLMS can only be rewritten in a specified sequence, and illegal write access is inhibited.

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

Address FFFF FCB2$_H$

Initial Value The contents of this register is undefined.

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

After writing to the PRCMDCMS register, you are permitted to write once to CLMS. The write access to CLMS must happen with the immediately following instruction.

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

(5) CLMCS - Sub oscillator clock monitor control register

The 8-bit CLMCS register is used to start the monitor of the sub oscillator clock.

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

Address FFFF F71A$_H$.

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

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

Table 4-23 CLMCS register contents

Bit position	Bit name	Function
0	CMRT	Sub oscillator clock monitor start: 0: Clock monitor for sub oscillator off. 1: Clock monitor for sub oscillator on.

Setting CLMCS.CMRT to 1 generates a trigger to activate the sub oscillator clock monitor.

Note

1. The sub oscillator clock monitor can only be started, if it has been enabled by setting CLMS.CLMES to 1.
2. Make sure that the sub oscillator stabilization time has elapsed before starting the clock monitor.

Caution Starting the sub oscillator clock monitor requires a special procedure. Refer to *“Operation of the Clock Monitors” on page 185*.

4.3 Power Save Modes

This chapter describes the various power save modes and how they are operated. For details see:

- *"Power save modes description" on page 167*
- *"Power save mode activation" on page 179*
- *"CPU operation after power save mode release" on page 181*

4.3.1 Power save modes description

This section explains the various power save modes in detail.

During power save mode

During all power save modes, the pins behave as follows:

- All output pins retain their function. That means all outputs are active, provided the required clock source is available.
- All input pins remain as input pins.
- All input pins with stand-by wake-up capability remain active, the function of all others is disabled.

During all power save modes, the main and sub oscillator clock monitors remain active, provided that the monitored oscillator is operating. If the oscillator is switched off during stand-by, the associated clock monitor enters stand-by as well.

Wake-up signals

The following signals can awake the controller from power save modes IDLE, WATCH, Sub-WATCH, STOP:

- Reset signals
  - external $\overline{\text{RESET}}$
  - Power-On-Clear reset RESPOC
  - Watchdog Timer reset RESWDT
    The Watchdog Timer must be configured to generate the reset WDTRES in case of overflow (WDTM.WDTMODE = 1) and it's input clock WDTCLK must be active during stand-by.
  - Clock monitors resets RESCMM, RESCMS
    The main oscillator respectively sub oscillator must be active during stand-by.
- Non maskable interrupts
  - NMI0
    The appropriate port must be configured correctly.
  - NMIWDT
    The Watchdog Timer must be configured to generate the in case of overflow (WDTM.WDTMODE = 0) and it's input clock WDTCLK must be active during stand-by.
- Maskable interrupts
  - external interrupts INTPn
    The appropriate port must be configured correctly.
  - CAN wake up interrupts INTCnWUP
    The appropriate port and the CAN (CnCTRL.PSMODE[1:0] = 01_B) must be configured correctly.

- Watch Timer interrupts INTWTnUV
  The Watch Timer clock WTCLK must be active and the Watch Timer must be enabled.
- Watch Calibration Timer interrupt INTTM01
  The Watch Calibration Timer clock WCTCLK must be active and the Watch Calibration Timer must be enabled.
- Voltage Comparators interrupts INTVCn
  The Voltage Comparators must be enabled.
- CSIB receive interrupts INTCBnR
  The CSIB must be operated in slave reception mode and the appropriate ports must be configured correctly.

Note that not all these signals are available in all power save modes.

The following signals can awake the controller from the power save mode HALT, provided the appropriate ports and modules are correctly configured and the required clocks are active:

- all reset signals
- the non-maskable interrupts NMI0, NMIWDT
- all maskable interrupts

To grant wake-up capability to maskable interrupts these interrupts have to be unmasked by setting the dedicated mask flags xxMK to 0 (refer to *"Interrupt Controller (INTC)" on page 187*).

A general disable of maskable interrupts acknowledgement ("DI", i.e. PSW.ID = 1) does not affect their wake-up capability.

After power save mode After power save mode release, the clock source for CPU operation should be checked. If the user application issues a wake-up request immediately after power save mode request, the power save mode may not be entered and the clock sources remain as programmed before the stand-by request.

After power save mode release, the same procedure as for system reset is required to set up the clock supply for the application.

Note In the following tables the clock status "operates" does not necessarily mean that the functions that use this clock source are operating as well.

(1) HALT mode

The HALT mode can be entered from normal run mode. In HALT mode, all clock settings remain unchanged. Only the CPU clock is suspended and hence program execution.

Table 4-24 Clock Generator status in HALT mode

Item	Status	Remarks
Main oscillator	unchanged	
Sub oscillator	operates	
Ring oscillator	operates	
SSCG	unchanged	
PLL	unchanged	
VBCLK (CPU system)	suspended	Clock setup is unchanged
IICLK	unchanged	
PCLK0, PCLK1	unchanged	
PCLK2…PCLK15	unchanged	
SPCLK0, SPCLK1	unchanged	
SPCLK2…SPCLK15	unchanged	
FOUTCLK	unchanged	
WTCLK / LCDCLK	unchanged	
WDTCLK	unchanged	
WCTCLK	unchanged	

The HALT mode can be released by any unmasked maskable interrupt, NMI or system reset.

On HALT mode release, all clock settings remain unchanged. The CPU clock resumes operation.

(2) IDLE mode

The IDLE mode can be entered from any run mode. The main oscillator must be operating. IDLE mode can not be entered if the CPU is clocked by the sub or ring oscillator.

In IDLE mode, the clock distribution is stopped (refer to the “Standby” switches in *Figure 4-1, “Block diagram of the Clock Generator,” on page 130*).

The states of all clock sources, that means, sub and ring oscillator as well as SSCG and PLL, remain unchanged. If a clock source was operating before entering IDLE mode, it continues operating.

Table 4-25 Clock Generator status in IDLE mode

Item	Status	Remarks
Main oscillator	unchanged	
Sub oscillator	operates	
Ring oscillator	operates	
SSCG	unchanged	
PLL	unchanged	
VBCLK (CPU system)	stopped	
IICLK	stopped	
PCLK0, PCLK1	stopped	
PCLK2…PCLK15	stopped	
SPCLK0, SPCLK1	stopped	
SPCLK2…SPCLK15	stopped	
FOUTCLK	unchanged	
WTCLK / LCDCLK	unchanged	
WDTCLK	unchanged	
WCTCLK	unchanged/stopped	Depends on clock selector PSM.CMODE

The IDLE mode can be released by

- the unmasked maskable interrupts INTPn, INTCnWUP, INTWTnUV, INTTM01, INTVCn, INTCBnR
- NMI0, NMIWDT
- $\overline{\text{RESET}}$, RESPOC, RESWDT, RESCMM, RESCMS

On IDLE mode release, the CPU clock and peripheral clocks are supplied by the main oscillator.

(3) WATCH mode

In WATCH mode, the clock supply for the CPU system and the majority of peripherals is stopped.

The main oscillator continues operation. PLL and SSCG are stopped. By default, ring oscillator and sub oscillator operation is not affected. For exceptions see *"Ring and sub oscillator operation" on page 184*.

Table 4-26 Clock Generator status in WATCH mode

Item	Status	Remarks
Main oscillator	unchanged/stopped	Stopped if PSM.OSCDIS = 1
Sub oscillator	operates	
Ring oscillator	operates/stopped	Stopped if WCC.ROSTP = 1
SSCG	stopped	
PLL	stopped	
VBCLK (CPU system)	stopped	
IICLK	stopped	
PCLK0, PCLK1	stopped	
PCLK2…PCLK15	stopped	
SPCLK0, SPCLK1	stopped	
SPCLK2…SPCLK15	stopped	
FOUTCLK	unchanged/stopped	Stopped, if the selected clock source stops
WTCLK / LCDCLK	unchanged/stopped	Stopped, if the selected clock source stops
WDTCLK	unchanged/stopped	Stopped, if the selected clock source stops
WCTCLK	unchanged/stopped	Depends on clock selector PSM.CMODE

The WATCH mode can be released by

- the unmasked maskable interrupts INTPn, INTCnWUP, INTWTnUV, INTTM01, INTVCn, INTCBnR
- NMI0, NMIWDT
- $\overline{\text{RESET}}$, RESPOC, RESWDT, RESCMM, RESCMS

On WATCH mode release, the CPU starts operation using the following clocks:

- if PSM.OSCDIS = 1: sub clock source selected before WATCH mode was entered, that means, either ring oscillator or sub oscillator (defined by PCC.SOSCP)
- if PSM.OSCDIS = 0: main oscillator

If the ring oscillator was stopped before entering the WATCH mode, the oscillation stabilization time for the ring oscillator is ensured by hardware after WATCH mode release.

PLL and SSCG remain stopped after WATCH release.

Peripheral clock supply is switched to main oscillator supply, if PSM.OSCDIS = 0, otherwise the ring oscillator is used for peripheral clocks.

(4) Sub-WATCH mode

In Sub-WATCH mode, the clock supply for the CPU and the majority of peripherals is stopped. Main oscillator, PLL, and SSCG are stopped. By default, ring oscillator and sub oscillator operation is not influenced. For exceptions see *“Ring and sub oscillator operation” on page 184*.

Table 4-27 Clock Generator status in Sub-WATCH mode

Item	Status	Remarks
Main oscillator	unchanged/stopped	Stopped if PSM.OSCDIS = 1
Sub oscillator	operates	
Ring oscillator	operates/stopped	Stopped if WCC.ROSTP = 1
SSCG	stopped	
PLL	stopped	
VBCLK (CPU system)	stopped	
IICLK	stopped	
PCLK0, PCLK1	stopped	
PCLK2…PCLK15	stopped	
SPCLK0, SPCLK1	stopped	
SPCLK2…SPCLK15	stopped	
FOUTCLK	unchanged	Stopped, if the selected clock source stops
WTCLK / LCDCLK	unchanged	Stopped, if the selected clock source stops
WDTCLK	unchanged/stopped	Stopped, if the selected clock source stops
WCTCLK	stopped	

The Sub-WATCH mode can be released by

- the unmasked maskable interrupts INTPn, INTCnWUP, INTWTnUV, INTVCn, INTCBnR
- NMI0, NMIWDT
- $\overline{\text{RESET}}$, RESPOC, RESWDT, RESCMM, RESCMS

On Sub-WATCH mode release, the CPU starts operation using the following clocks:

- if PSM.OSCDIS = 1: sub clock source selected before Sub-WATCH mode was entered, that means, either ring oscillator or sub oscillator (defined by PCC.SOSCP)
- if PSM.OSCDIS = 0: main oscillator

If the ring oscillator was stopped before entering the Sub-WATCH mode, the oscillation stabilization time for the ring oscillator is ensured by hardware after Sub-WATCH release.

PLL and SSCG remain stopped after Sub-WATCH release.

Peripheral clock supply is switched to main oscillator supply, if PSM.OSCDIS = 0, otherwise the ring oscillator is used for peripheral clocks.

(5) STOP mode

In STOP mode, all clock sources are stopped, except sub and ring oscillator. These can be configured in register WCC to stop as well. No clock is available, and no internal self-timed processes operates.

Table 4-28 Clock Generator status in STOP mode

Item	Status	Remark
Main oscillator	stopped	
Sub oscillator	operates/stopped	Stopped if WCC.SOSTP = 1
Ring oscillator	operates/stopped	Stopped if WCC.ROSTP = 1
SSCG	stopped	
PLL	stopped	
VBCLK (CPU system)	stopped	
IICLK	stopped	
PCLK0, PCLK1	stopped	
PCLK2…PCLK15	stopped	
SPCLK0, SPCLK1	stopped	
SPCLK2…SPCLK15	stopped	
FOUTCLK	stopped	
WTCLK / LCDCLK	stopped	
WDTCLK	unchanged/stopped	Stopped, if the selected clock source stops
WCTCLK	stopped	

The STOP mode can be released by

- the unmasked maskable interrupts INTPn, INTCnWUP, INTVCn, INTCBnR
- NMI0, NMIWDT
- $\overline{\text{RESET}}$, RESPOC, RESWDT, RESCMM, RESCMS

On STOP mode release, the CPU clock and peripheral clocks are supplied by the main oscillator.

(6) Clock status summary

Table 4-29 on page 174 summarizes the status of all clocks delivered by the Clock Generator in the different states.

“Normal” describes all status except reset and power save modes.

The HALT mode is not listed in the table. It does not change any of the table items, but stoppes only the CPU core operation.

Below the table you find the explanation of the terms used in the table.

Table 4-29 Status of oscillators and Clock Generator output clocks (1/3)

Macro	Clock signal	Condition	Reset	Reset release	Normal	IDLE	IDLE release	STOP	STOP release	WATCH	WATCH release	Sub-WATCH	Sub-WATCH release
Oscillators													
Main-osc	–	OSCDIS=0	n.a.		on	on		stop	on	on	on	stop	on
		OSCDIS=1	stop		stop	n.a.				stop	n.a.		stop
Sub-osc	–	SOSTP=1	n.a.		on	on		stop	on	on		on	
		SOSTP=0	on					on					
Ring-osc	–	ROSTP=1	n.a.		on	on		stop	on	stop	on	stop	on
		ROSTP=0	on					on		on		on	
SSCG/PLL													
SSCG	–	–	stby		scen	scen		stby		stby		stby	
PLL	–	–			pllen	pllen							
Clock Generator output clocks													
CPU system clock	VBCLK	CLS/CKS = 000_B	n.a.		MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
		CLS/CKS = 001_B			SSCCLK		n.a.		n.a.		n.a.		n.a.
		CLS/CKS = $01x_B$			PLLCLK		n.a.		n.a.		n.a.		n.a.
		CLS/CKS = $1xx_B$	off	ROCLK	SBCLK		n.a.		n.a.		n.a.		SBCLK

Table 4-29 Status of oscillators and Clock Generator output clocks (2/3)

Macro	Clock signal	Condition		Reset	Reset release	Normal	IDLE	IDLE release	STOP	STOP release	WATCH	WATCH release	Sub-WATCH	Sub-WATCH release
Peripheral clocks	PCLK0 PCLK1	PERIC=0		off	MOCLK[a]	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
		PERIC=1		n.a.		PLLCLK		PLLCLK		n.a.		n.a.		n.a.
	PCLK2 - PCLK15	–		off	MOCLK[a]	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
	SPCLK0 SPCLK1	SPSEL[1:0]=00_B		off	MOCLK[a]	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
		SPSEL[1:0]=01_B		n.a.		PLLCLK		PLLCLK		n.a.		n.a.		n.a.
		SPSEL[1:0]=11_B				SSCCLK		SSCCLK		n.a.		n.a.		n.a.
	SPCLK2 - SPCLK15	SPSEL1=0		off	MOCLK[a]	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
		SPSEL1=1		n.a.		PLLCLK		PLLCLK		n.a.		n.a.		n.a.
Watch Calibration Timer	WCTCLK	CMODE=0		off	MOCLK[a]	PCLK1	off	PCLK1	off	PCLK1	off	PCLK1	off	PCLK1
		CMODE=1		n.a.		MOCLK	MOCLK	MOCLK		MOCLK	MOCLK	MOCLK		MOCLK
Watchdog Timer	WDTCLK	SOSC=0 WDTSEL0=0	WDTSEL1=0	off	ROSCK	ROSCK	off	ROSCK	off	ROSCK	off	ROSCK	off	ROSCK
			WDTSEL1=1	n.a.			ROSCK		ROSCK (off[b])		ROSCK (off[b])		ROSCK (off[b])	
		SOSC=1 WDTSEL0=0	WDTSEL1=0	n.a.		SOCLK	off	SOCLK	off	SOCLK	off	SOCLK	off	SOCLK
			WDTSEL1=1				SOCLK		SOCLK (off[b])		SOCLK		SOCLK	
		SOSC=x WDTSEL0=1	WDTSEL1=0	n.a.		MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
			WDTSEL1=1				MOCLK				MOCLK			
I^{2}C	IICLK	IICSEL[1:0]=00_B		off	MOCLK[a]	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
		IICSEL[1:0]=01_B		n.a.		SSCSCK		SSCSCK		n.a.		n.a.		n.a.
		IICSEL[1:0]=$1x_B$				PLLSCK		PLLSCK		n.a.		n.a.		n.a.

Table 4-29 Status of oscillators and Clock Generator output clocks (3/3)

Macro	Clock signal	Condition	Reset	Reset release	Normal	IDLE	IDLE release	STOP	STOP release	WATCH	WATCH release	Sub-WATCH	Sub-WATCH release
Watch Timer LCD-C/D	WTCLK LCDCLK	WTSOS/WTSEL0=00$_B$	ROSCK		ROSCK	ROSCK		ROSCK (off^b)	ROSCK	ROSCK (off^b)	ROSCK	ROSCK (off^b)	ROSCK
		WTSOS/WTSEL0=10$_B$	n.a.		SOCLK	SOCLK		SOCLK (off^b)	SOCLK	SOCLK	SOCLK	SOCLK	SOCLK
		WTSOS/WTSEL0=x1$_B$			MOCLK	MOCLK		off	MOCLK	MOCLK (off^c)	MOCLK	MOCLK (off^c)	MOCLK
Port	FOUTCLK	FOSOS/FOCKS1/0=x00$_B$	off		MOCLK	MOCLK		MOCLKLCD-C/D		MOCLK		MOCLK	
		FOSOS/FOCKS1/0=x01$_B$	n.a.		SSCCLK	SSCCLK		SSCCLK		SSCCLK		SSCCLK	
		FOSOS/FOCKS1/0=x10$_B$			PLLCLK	PLLCLK		PLLCLK		PLLCLK		PLLCLK	
		FOSOS/FOCKS1/0=011$_B$			ROCLK	ROCLK		ROCLK		ROCLK		ROCLK	
		FOSOS/FOCKS1/0=111$_B$			SOCLK	SOCLK		SOCLK		SOCLK		SOCLK	

a) After reset release these clocks are supplied with the ring ROCLK. When the main oscillator is stable, these clocks are automatically changed to MOCLK.

b) ROCLK (SOCLK) remains clock source, but ring oscillator (sub oscillator) may be stopped in the respective power save mode by WCC.ROSTP = 1 (WCC.SOSTP = 1).

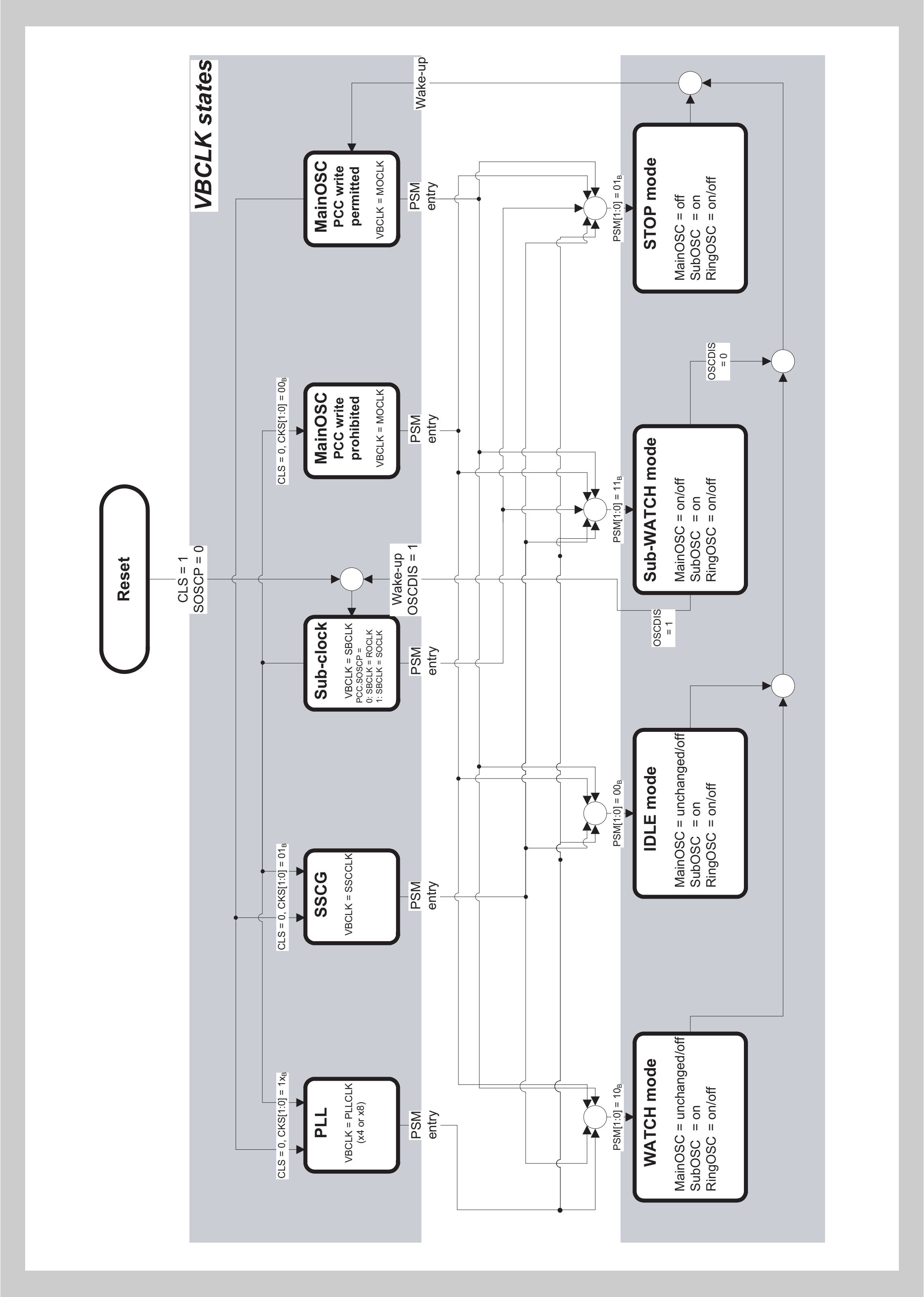
c) MOSCLK remains clock source, but main oscillator may be stopped in the respective power save mode by PSM.OSCDIS = 1.

In the table following terms are used:

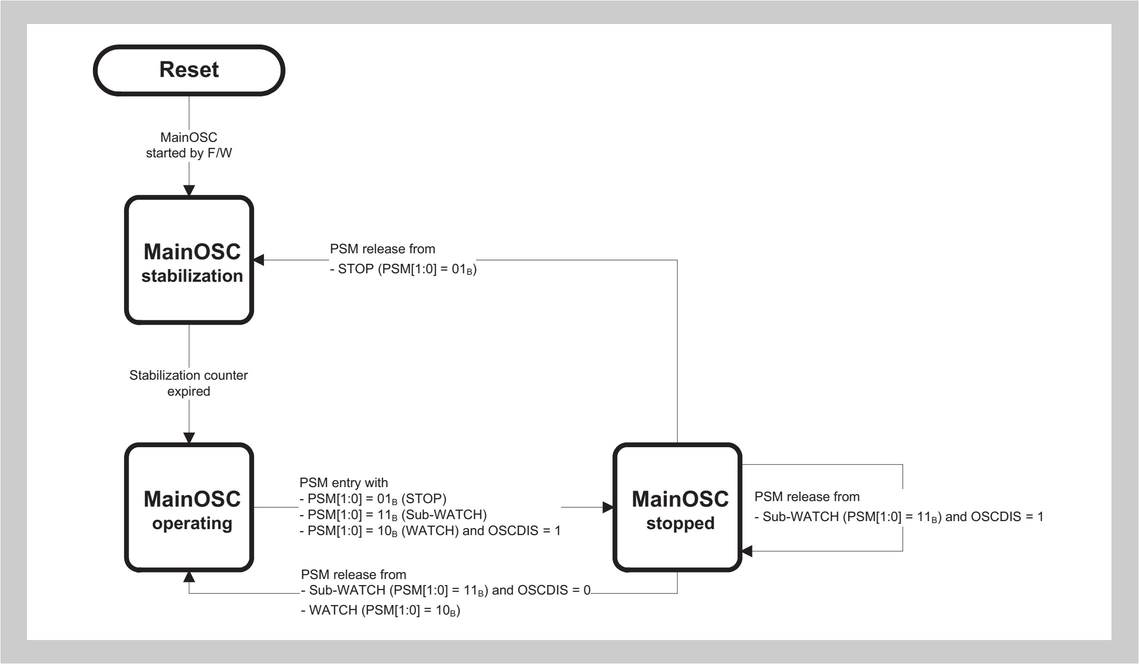
stop:	Oscillator stopped	MOCLK:	Main oscillator clock
on:	Oscillator operating	ROCLK:	Ring oscillator clock
stby:	PLL/SSCG in standby, no clock output	SOCLK:	Sub oscillator clock
pllen/scen:	PLL/SSCG genrates clock output	SBCLK:	Sub clock – PCC.SOSCP = 0: ROCLK – PCC.SOSCP = 1: SOCLK
off:	Clock inactive	PLLCLK:	PLL output clock
n.a.	not applicable (control bits are determined by hardware)	SSCGCLK:	SSCG output clock

4.3.2 Clock Generator state transistions

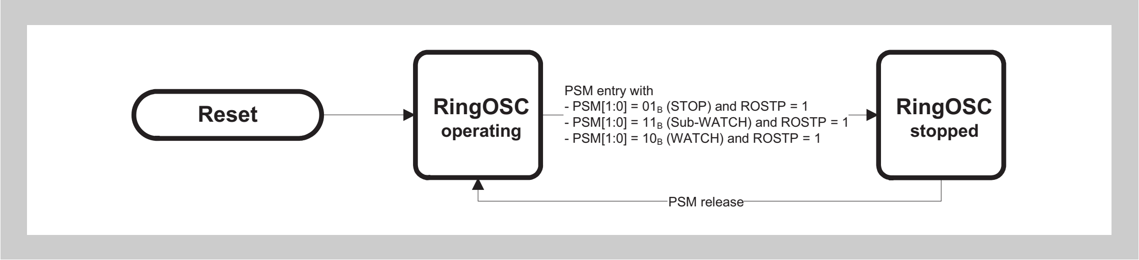
(1) VBCLK state transitions



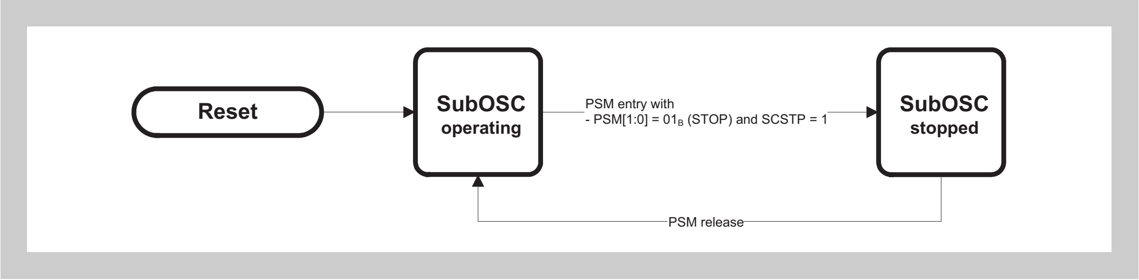
(2) Main oscillator state transitions



(3) Ring oscillator states



(4) Sub oscillator states



4.3.3 Power save mode activation

In the following procedures for securely entering a power save mode are described.

Stepper-C/D shut down In order to minimize power consumption during power save modes the Stepper Motor Controller/Driver needs to be shut down in a special sequence. Refer to *“MCNTCn0, MCNTCn1 - Timer mode control registers” on page 799*.

(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) WATCH, Sub-WATCH, STOP and IDLE mode

For entering these power save mode proceed as follows:

1. In case maskable interrupts shall be used for wake-up unmask these interrupts by IMRm.xxMK = 0 (refer to *“IMR0 to IMR5 - Interrupt mask registers” on page 214*).
2. Mask all other interrupts, i.e.
   - none wake-up capable interrupts
   - wake-up capable interrupts which shall not be used for wake-up

   by IMRm.xxMK = 1. This prevents the power save mode entry procedure from being interrupted by these interrupts.
3. It is recommended to disable interrupt acknowledgement by the “di” instruction.
4. Specify the desired power save mode in PSM.PSM[1:0].
5. Enable writing to the write-protected register PSC by writing to PRCMD.
6. Write to PSC for specifying permitted wake-up events and activate the power save mode by setting PSC.STP to 1.

Example The following example shows how to initialize and enter a WATCH, Sub-WATCH, STOP or IDLE power save mode.

First the desired power save mode is specified (WATCH mode in this example, that means PSM.PSM[1:0] = 10_B).

The PSC register is a write-protected register, and the PRCMD register is the corresponding write-enable register. PRCMD has to be written immediately before writing to PSC.

In this example, maskable interrupts are permitted to leave the power save mode.

```
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,PSM[r0]             // PSM.PSM[1:0] = 10B: WATCH 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. The data written to the PRCMD register must be the same data that shall be written to the write-protected register afterwards.
   The above example ensures this method, since the contents of r10 is first written to PRCMD and then immediately to PSC.

5. Make sure that all DMA channels are disabled. Otherwise a DMA could happen between steps 7 and 8, and the power down mode may not be entered at all.
   Further on do not perform write operations to PRCMD and write-protected registers by DMA transfers.
6. No special sequence is required for reading the PSC register.

Caution If a wake-up event occurs within the 5 “nop” instructions after a power save mode request (PSC.STP = 1) the microcontroller is immediately returning from power save mode, but may have not at all or only partly entered the power save mode. Following three situations can occur:

1. power save mode request not accepted
   wake-up configuration not established, PLL/SSCG are operating
2. power save mode request accepted, but not completed
   wake-up configuration established, but PLL/SSCG operating
3. power save mode request accepted and completed
   wake-up configuration established, PLL/SSCG stopped

4.3.4 CPU operation after power save mode release

Clock Generator re-configuration

The clock for the CPU system can be switched only once after reset or power save mode release.

The clocks for the Watchdog Timer, Watch Timer, and LCD Controller/Driver can be switched only once after system reset.

Access to peripherals that have no clock supply in Sub-WATCH mode may cause system deadlock. This can happen if the main oscillator remains disabled.

Wake-up configuration

Wake-up configuration established means that all registers and clock paths are set to their wake-up state.

The software should check after wake-up whether the expected wake-up configuration has been completely established. This can be achieved by observing

- following clock generator registers, which are modified by power save mode entry and wake-up
  - after WATCH, Sub-WATCH, STOP wake-up following bits are cleared: CKC.PLLEN, CKC.SCEN, CKC.PERIC, SCC.SEL, ICC.SEL
  - after IDLE or STOP wake-up following bits are cleared: PCC.CLS, PCC.CKS
  - after Sub-WATCH or WATCH wake-up
    PCC.CLS/PCC.CKS = 000_B, if PSM.OSCDIS = 0
    PCC.CLS/PCC.CKS = 100_B, if PSM.OSCDIS = 1
- the “completed power save mode” bit CGSTAT.CMPLPSM
  - CGSTAT.CMPLPSM = 0 if a power save mode request has been accepted but not completed, wake-up configuration established, but PLL/

SSCG operating (provided that CGSTAT.CMPLPSM = 1 before power save mode request)

- CGSTAT.CMPLPSM = 1 if a power save mode has been completely entered, wake-up configuration established, PLL/SSCG stopped (provided that a power save mode request has been accepted before, i.e. CGSTAT.CMPLPSM = 1 → 0)

Note that CGSTAT.CMPLPSM is set to 0 if a power save mode request is accepted. If it was 0 before it does not change it’s state.

Table 4-30 summarizes the different configurations.

Table 4-30 Power save mode wake-up configurations

CGSTAT.CMPLPSM		Registers and clock paths[a]	Configuration after wake-up
before PSM-RQ[b]	after wake-up		
0	0	not changed	PSM-RQ not accepted
		changed	PSM-RQ accepted configuration done, but PLL/ SSCG operating
	1	not changed	not possible
		changed	PSM-RQ accepted configuration done, PLL/SSCG off
1	0	not changed	not possible
		changed	PSM-RQ accepted configuration done, but PLL/ SSCG operating
	1	not changed	PSM-RQ not accepted
		changed	PSM-RQ accepted configuration done, PLL/SSCG off

a) A change of a register’s contents can only be taken as an indicator if it is before power save mode request different to the wake-up configuration.

b) PSM-RQ: power save mode request (PSC.STP = 1)

If the power save mode request was accepted the entire clock generator can be reconfigured after wake-up. Afterwards set CKC.PLLEN = 1 and CKC.SCEN = 1 and wait the stabilization times before using the PLL and SSCG as clock sources.

After IDLE and STOP

On return from IDLE or STOP mode, the bits PCC.CLS, PCC.CKS1, and PCC.CKS2 are cleared. After IDLE mode, the main oscillator is still running; on return from STOP mode, it is automatically started.

As a result, the main oscillator is chosen and enabled as the source for the CPU system clock VBCLK.

After WATCH

In WATCH mode the main oscillator operation depends on PSM.OSCDIS:

- If PSM.OSCDIS was 0 before entering WATCH mode the main oscillator remains active. After WATCH mode release the main oscillator is chosen as the CPU system clock.
- If PSM.OSCDIS was 1 before entering WATCH mode the main oscillator is stopped during WATCH mode. After WATCH mode release the main

oscillator is automatically started, the oscillator stabilization time is waited and the main oscillator is chosen as the CPU system clock.

After Sub-WATCH

In Sub-WATCH mode the main oscillator is stopped. On return from Sub-WATCH, PCC.CLS is set to the status of PSM.OSCDIS.

- If PSM.OSCDIS was 0 before entering Sub-WATCH, the main oscillator is started and chosen as the source for the CPU system clock (PCC.CLS = 0, PCC.CKS[1:0] = 00_B).
- If PSM.OSCDIS was 1 before entering Sub-WATCH, the main oscillator remains stopped, and the CPU is clocked by a sub clock (PCC.CLS = 1, PCC.CKS[1:0] = xx_B).

“Sub clock” means the clocks supplied by either the 32 KHz sub oscillator or the 200 KHz ring oscillator. The selection must be made in the PCC register *before* entering the Sub-WATCH or WATCH mode:

- PCC.SOSCP = 0: Ring oscillator
- PCC.SOSCP = 1: Sub oscillator

Software can switch from sub clock CPU operation to normal run mode (by enabling the main oscillator by PSM.OSCDIS = 0) or re-enter Sub-WATCH respectively WATCH mode.

After HALT

On return from HALT mode the CPU resumes operation with the same clock settings as before HALT was entered.

4.4 Clock Generator Operation

4.4.1 Ring and sub oscillator operation

By default, sub and ring oscillator operate during all power save modes.

However, it can be specified in the WCC register that the sub oscillator stops in STOP mode (WCC.SOSTP).

It can also be specified that the ring oscillator stops in WATCH, Sub-WATCH, and STOP mode (WCC.ROSTP).

These bits can be written once after system reset, independent of the reset source.

4.4.2 Watch Timer and Watch Calibration Timer clocks

The Watch Timer input clock WTCLK can be derived directly from the main, sub, or ring oscillator. Therefore, the WT can be operating in all power save modes.

Because PCLK1 is stopped during power save modes, the Watch Calibration Timer input clock WCTCLK can be directly connected to the main oscillator output.

Note WCTCLK is not available in Sub-WATCH and STOP mode where the main oscillator is stopped. These modes must be released before the WCT can operate.

4.4.3 Clock output FOUTCLK

The Clock Generator output signal FOUTCLK supplies a clock for external components. It can be derived from any internal clock source, that means ring oscillator, sub oscillator, main oscillator, PLL, or SSCG.

FOUTCLK must be enabled by register setting (FCC.FOEN = 1). It is not influenced by the power save modes. But FOUTCLK stops, if the selected clock source stops.

After reset release, FOUTCLK is disabled (register FCC is cleared), and the pin FOUT put in input mode.

Note

1. If you change the configuration of FOUTCLK or enable/disable the selected clock source while FOUTCLK is active, glitches or irregular clock periods may appear at the output pin.
2. The clock signal FOUTCLK cannot be used to synchronize external circuitry to other output signals of the microcontroller—it has no specified phase relation to other output signals.
3. There is an upper frequency limit for the output buffer of the FOUTCLK function. Do not select a frequency higher than the maximum output buffer frequency. Please refer to the Electrical Target Specification for the frequency limit.

4.4.4 Operation of the Clock Monitors

The microcontroller provides two separate clock monitors to watch the activity of the main oscillator and the sub oscillator.

(1) Description

The functional block diagram is shown below.

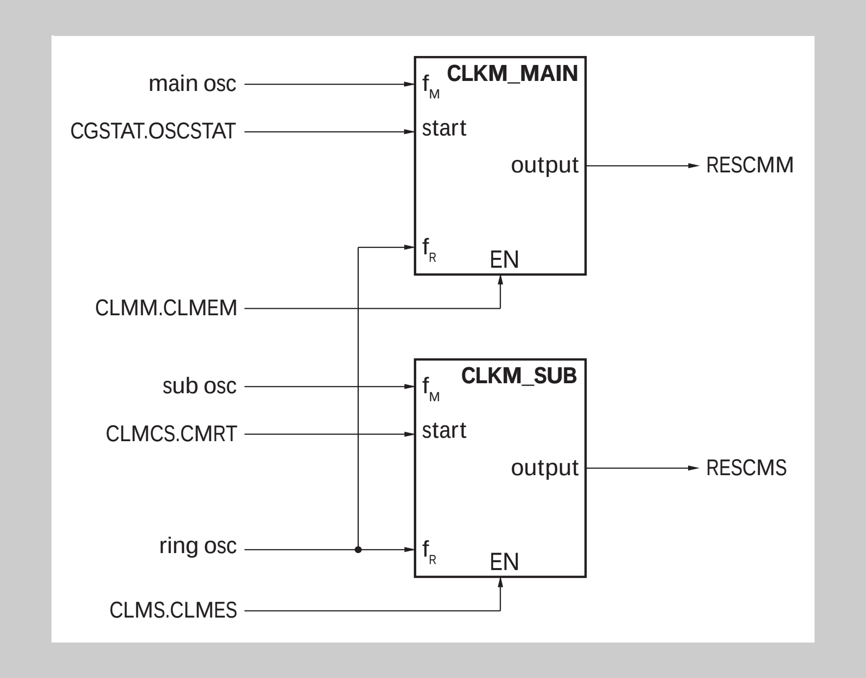


Figure 4-2 Clock Monitors Block Diagram

The clock monitors use the ring oscillator (f_R) for monitoring the main and sub oscillators (f_M).

If the main oscillator clock monitor detects a malfunction of the main oscillator (no pulse), it generates the reset request RESCMM. If the sub oscillator clock monitor detects a malfunction of the sub oscillator, it generates the reset request RESCMS.

(2) Start and stop

Before the clock monitors can be started, they have to be enabled by setting CLMM.CLMEM and CLMS.CLMES to 1.

Main oscillator monitor start

After enabling CLMM.CLMEM = 1 the main oscillator monitor is automatically started as soon as the main oscillator is stable, indicated by CGSTAT.OSCSTAT = 1.

Main oscillator monitor start

After enabling CLMM.CLMES = 1 the sub oscillator monitor must be started by software by setting CLMCS.CMRT to 1.

After starting the sub oscillator clock monitor by CLMCS.CMRT = 1 clear CLMCS.CMRT by software.

Since CLMCS.CMRT = 1 is synchronized with the ring oscillator any change of this bit has to be maintained for at least 65 ring oscillator periods $T_{ROSC} = 1/f_{ROSC}$ to become effective. Therefore a wait period has to be assured before this bit is changed again.

Proceed as follows to start the sub oscillator clock monitor:

1. After reset enable sub oscillator clock monitor:
   PRCMDCM = FF_Hpermit write to CLMS
   CLMS.CLMES = 1enable sub oscillator clock monitor
2. Make sure the sub oscillator is stable.
3. Start sub oscillator clock monitor after reset and after power save mode wake-up:
   CLMCS.CMRT = 1
4. Wait for 65 ring oscillator periods T_{ROSC} before resetting CMRT:
   wait (65 x max(T_{ROSC}))
5. Clear CLMCS.CMRT:
   CLMCS.CMRT = 0
6. Before CMRT should be set to 1 again, wait for 65 ring oscillator periods T_{ROSC}:
   wait (65 x max(T_{ROSC}))

Note that the minimum ring oscillator frequency min(f_{ROSC}) (max(T_{ROSC})) has to be taken into account for the wait time in steps (3) and (5).

Caution The sub oscillator clock monitor is sometimes already started by setting CLMS.CLMES = 1, i.e. without CLMCS.CMRT = 1.
In these cases it would not be required to start the sub oscillator by setting CLMCS.CMRT = 1 additionally.

Since it is unpredictable whether the clock monitor has already started after CLMS.CLMES = 1 the procedure described above should be followed in any case.

---

(3) Operation during and after power save modes

Main oscillator stopped If the main oscillator is stopped, its clock monitor changes to stand-by. When the main oscillator is restarted after power save mode release, the main oscillator clock monitor restarts automatically.

Sub oscillator stopped If the sub oscillator is stopped, its clock monitor stops.

When the sub oscillator is restarted after power save mode release, the sub oscillator clock monitor does not start automatically.

Software must ensure that the sub oscillator stabilization time has elapsed and then start the monitor by setting CLMCS.CMRT to 1.

Ring oscillator stopped If the ring oscillator is stopped, both clock monitors’ operation is suspended. Their operation is automatically resumed as soon as the ring 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:

Interrupt source	**µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPDF3420**	**µPD70F3424, µPD70F3425**
Internal peripherals	75	82
External	7	8
Software	2	2

  - 8 levels of programmable priorities (maskable interrupts)
  - Multiple interrupt control according to priority
  - Masks can be specified for each maskable interrupt request
  - Noise elimination, edge detection and valid edge specification, level detection for external interrupt request signals
  - Wake-up capable
    (analogue noise elimination for external interrupt request signals)
  - NMI and INTP0 share the same pin
- Exceptions
  - Software exceptions: 2 channels with each 16 sources
  - Exception traps: 2 sources (illegal opcode exception and debug trap)

Table 5-1 µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 interrupt/ exception source list (1/4)

Type	Classification	Interrupt/Exception Source			Default Priority	Exception Code	Handler Address	Restored PC
		Name	Generating Source	Generating Unit				
Reset	Interrupt	RESET	$\overline{\text{RESET}}$ input	Pin	–	0000H	00000000H	undef.
Non-maskable	Interrupt	NMI0	NMI Input	PORT	–	0010H	00000010H	nextPC
		NMIWDT	Watchdog Timer	WDT	–	0020H	00000020H	nextPC
		NMI2	Unused	–	–	0030H	00000030H	nextPC
Software exception	Exception	TRAP0n (n = 0 to F_H)	TRAP instruction	–	–	004nH (n = 0 to F_H)	00000040H	nextPC
	Exception	TRAP1n (n = 0 to F_H)	TRAP instruction	–	–	005nH (n = 0 to F_H)	00000050H	nextPC
Exception trap	Exception	ILGOP/ DBTRAP	Illegal opcode/ DBTRAP instruction	–	–	0060H	00000060H	nextPC
Maskable	Interrupt	INTVC0	Voltage Comparator 0	AC0	0	0080H	00000080H	next PC
	Interrupt	INTVC1	Voltage Comparator 1	AC1	1	0090H	00000090H	next PC
	Interrupt	INTWT0UV	WT0 underflow	WT0	2	00A0H	000000A0H	next PC
	Interrupt	INTWT1UV	WT1 underflow	WT1	3	00B0H	000000B0H	next PC
	Interrupt	INTTM00	Watch calibration timer capture compare	WCT	4	00C0H	000000C0H	next PC
	Interrupt	Reserved	Reserved	–	5	00D0H	000000D0H	next PC
	Interrupt	INTP0	External interrupt 0	PORT	6	00E0H	000000E0H	next PC
	Interrupt	INTP1	External interrupt 1	PORT	7	00F0H	000000F0H	next PC
	Interrupt	INTP2	External interrupt 2	PORT	8	0100H	00000100H	next PC
	Interrupt	INTP3	External interrupt 3	PORT	9	0110H	00000110H	next PC
	Interrupt	INTP4	External interrupt 4	PORT	10	0120H	00000120H	next PC
	Interrupt	INTP5	External interrupt 5	PORT	11	0130H	00000130H	next PC
	Interrupt	INTP6	External interrupt 6	PORT	12	0140H	00000140H	next PC
	Interrupt	INTTZ0UV	TMZ0 underflow	TMZ0	13	0150H	00000150H	next PC
	Interrupt	INTTZ1UV	TMZ1 underflow	TMZ1	14	0160H	00000160H	next PC
	Interrupt	INTTZ2UV	TMZ2 underflow	TMZ2	15	0170H	00000170H	next PC
	Interrupt	INTTZ3UV	TMZ3 underflow	TMZ3	16	0180H	00000180H	next PC
	Interrupt	INTTZ4UV	TMZ4 underflow	TMZ4	17	0190H	00000190H	next PC
	Interrupt	INTTZ5UV	TMZ5 underflow	TMZ5	18	01A0H	000001A0H	next PC
	Interrupt	INTTP0OV	TMP0 overflow	TMP0	19	01B0H	000001B0H	next PC
	Interrupt	INTTP0CC0	TMP0 capture compare channel 0	TMP0	20	01C0H	000001C0H	next PC
	Interrupt	INTTP0CC1	TMP0 capture compare channel 1	TMP0	21	01D0H	000001D0H	next PC
	Interrupt	INTTP1OV	TMP1 overflow	TMP1	22	01E0H	000001E0H	next PC
	Interrupt	INTTP1CC0	TMP1 capture compare channel 0	TMP1	23	01F0H	000001F0H	next PC
	Interrupt	INTTP1CC1	TMP1 capture compare channel 1	TMP1	24	0200H	00000200H	next PC
	Interrupt	INTTP2OV	TMP2 overflow	TMP2	25	0210H	00000210H	next PC

Table 5-1 µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 interrupt/exception source list (2/4)

Type	Classification	Interrupt/Exception Source			Default Priority	Exception Code	Handler Address	Restored PC
		Name	Generating Source	Generating Unit				
Maskable	Interrupt	INTTP2CC0	TMP2 capture compare channel 0	TMP2	26	0220H	00000220H	next PC
	Interrupt	INTTP2CC1	TMP2 capture compare channel 1	TMP2	27	0230H	00000230H	next PC
	Interrupt	INTTP3OV	TMP3 overflow	TMP3	28	0240H	00000240H	next PC
	Interrupt	INTTP3CC0	TMP3 capture compare channel 0	TMP3	29	0250H	00000250H	next PC
	Interrupt	INTTP3CC1	TMP3 capture compare channel 1	TMP3	30	0260H	00000260H	next PC
	Interrupt	INTTG0OV0	TMG0 overflow interrupt 0	TMG0	31	0270H	00000270H	next PC
	Interrupt	INTTG0OV1	TMG0 overflow interrupt 1	TMG0	32	0280H	00000280H	next PC
	Interrupt	INTTG0CC0	TMG0 capture compare channel 0	TMG0	33	0290H	00000290H	next PC
	Interrupt	INTTG0CC1	TMG0 capture compare channel 1	TMG0	34	02A0H	000002A0H	next PC
	Interrupt	INTTG0CC2	TMG0 capture compare channel 2	TMG0	35	02B0H	000002B0H	next PC
	Interrupt	INTTG0CC3	TMG0 capture compare channel 3	TMG0	36	02C0H	000002C0H	next PC
	Interrupt	INTTG0CC4	TMG0 capture compare channel 4	TMG0	37	02D0H	000002D0H	next PC
	Interrupt	INTTG0CC5	TMG0 capture compare channel 5	TMG0	38	02E0H	000002E0H	next PC
	Interrupt	INTTG1OV0	TMG1 overflow interrupt 0	TMG1	39	02F0H	000002F0H	next PC
	Interrupt	INTTG1OV1	TMG1 overflow interrupt 1	TMG1	40	0300H	00000300H	next PC
	Interrupt	INTTG1CC0	TMG1 capture compare channel 0	TMG1	41	0310H	00000310H	next PC
	Interrupt	INTTG1CC1	TMG1 capture compare channel 1	TMG1	42	0320H	00000320H	next PC
	Interrupt	INTTG1CC2	TMG1 capture compare channel 2	TMG1	43	0330H	00000330H	next PC
	Interrupt	INTTG1CC3	TMG1 capture compare channel 3	TMG1	44	0340H	00000340H	next PC
	Interrupt	INTTG1CC4	TMG1 capture compare channel 4	TMG1	45	0350H	00000350H	next PC
	Interrupt	INTTG1CC5	TMG1 capture compare channel 5	TMG1	46	0360H	00000360H	next PC
	Interrupt	Reserved	Reserved	—	47	0370H	00000370H	next PC
	Interrupt	Reserved	Reserved	—	48	0380H	00000380H	next PC
	Interrupt	INTAD	ADC end of conversion	ADC	49	0390H	00000390H	next PC
	Interrupt	INTC0ERR	CAN0 error interrupt	CAN0	50	03A0H	000003A0H	next PC
	Interrupt	INTC0WUP	CAN0 wake up interrupt	CAN0	51	03B0H	000003B0H	next PC
	Interrupt	INTC0REC	CAN0 receive interrupt	CAN0	52	03C0H	000003C0H	next PC
	Interrupt	INTC0TRX	CAN0 transmit interrupt	CAN0	53	03D0H	000003D0H	next PC

Table 5-1 µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 interrupt/ exception source list (3/4)

Type	Classification	Interrupt/Exception Source			Default Priority	Exception Code	Handler Address	Restored PC
		Name	Generating Source	Generating Unit				
Maskable	Interrupt	INTCB0RE	CSIB0 receive error interrupt	CSIB0	54	03E0H	000003E0H	next PC
	Interrupt	INTCB0R	CSIB0 receive complete interrupt	CSIB0	55	03F0H	000003F0H	next PC
	Interrupt	INTCB0T	CSIB0 transmit interrupt	CSIB0	56	0400H	00000400H	next PC
	Interrupt	INTUA0RE	UARTA0 receive error interrupt	UARTA0	57	0410H	00000410H	next PC
	Interrupt	INTUA0R	UARTA0 receive complete interrupt	UARTA0	58	0420H	00000420H	next PC
	Interrupt	INTUA0T	UARTA0 transmit interrupt	UARTA0	59	0430H	00000430H	next PC
	Interrupt	INTUA1RE	UARTA1 receive error interrupt	UARTA1	60	0440H	00000440H	next PC
	Interrupt	INTUA1R	UARTA1 receive complete interrupt	UARTA1	61	0450H	00000450H	next PC
	Interrupt	INTUA1T	UARTA1 transmit interrupt	UARTA1	62	0460H	00000460H	next PC
	Interrupt	INTIIC0	IIC0 interrupt	IIC0	63	0470H	00000470H	next PC
	Interrupt	INTIIC1	IIC1 interrupt	IIC1	64	0480H	00000480H	next PC
	Interrupt	Reserved	Reserved	—	65	0490H	00000490H	next PC
	Interrupt	INTDMA0	DMA0 transmission end	DMA0	66	04A0H	000004A0H	next PC
	Interrupt	INTDMA1	DMA1 transmission end	DMA1	67	04B0H	000004B0H	next PC
	Interrupt	INTDMA2	DMA2 transmission end	DMA2	68	04C0H	000004C0H	next PC
	Interrupt	INTDMA3	DMA3 transmission end	DMA3	69	04D0H	000004D0H	next PC
	Interrupt	INT70	not generated by hardware[a]	—	70	04E0H	000004E0H	next PC
	Interrupt	INT71	not generated by hardware[a]	—	71	04F0H	000004F0H	next PC
	Interrupt	Reserved	Reserved	—	72	0500H	00000500H	next PC
	Interrupt	INTC1ERR	CAN1 error interrupt	CAN1	73	0510H	00000510H	next PC
	Interrupt	INTC1WUP	CAN1 wake up interrupt	CAN1	74	0520H	00000520H	next PC
	Interrupt	INTC1REC	CAN1 receive interrupt	CAN1	75	0530H	00000530H	next PC
	Interrupt	INTC1TRX	CAN1 transmit interrupt	CAN1	76	0540H	00000540H	next PC
	Interrupt	Reserved	Reserved	—	77... 80	0xx0H	00000xx0H	next PC

Table 5-1 µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 interrupt/ exception source list (4/4)

Type	Classification	Interrupt/Exception Source			Default Priority	Exception Code	Handler Address	Restored PC
		Name	Generating Source	Generating Unit				
Maskable	Interrupt	INTTG2OV0	TMG2 overflow interrupt 0	TMG2	81	0590H	00000590H	next PC
	Interrupt	INTTG2OV1	TMG2 overflow interrupt 1	TMG2	82	05A0H	000005A0H	next PC
	Interrupt	INTTG2CC0	TMG2 capture compare channel 0	TMG2	83	05B0H	000005B0H	next PC
	Interrupt	INTTG2CC1	TMG2 capture compare channel 1	TMG2	84	05C0H	000005C0H	next PC
	Interrupt	INTTG2CC2	TMG2 capture compare channel 2	TMG2	85	05D0H	000005D0H	next PC
	Interrupt	INTTG2CC3	TMG2 capture compare channel 3	TMG2	86	05E0H	000005E0H	next PC
	Interrupt	INTTG2CC4	TMG2 capture compare channel 4	TMG2	87	05F0H	000005F0H	next PC
	Interrupt	INTTG2CC5	TMG2 capture compare channel 5	TMG2	88	0600H	00000600H	next PC
	Interrupt	INTCB1RE	CSIB1 receive error interrupt	CSIB1	89	0610H	00000610H	next PC
	Interrupt	INTCB1R	CSIB1 receive complete interrupt	CSIB1	90	0620H	00000620H	next PC
	Interrupt	INTCB1T	CSIB1 transmit interrupt	CSIB1	91	0630H	00000630H	next PC
	Interrupt	Reserved	Reserved	—	92...94	0xx0H	00000xx0H	next PC
	Interrupt	INTLCD	LCDBUSIF transmit interrupt	LCDBUSIF	95	0670H	00000670H	next PC

a) These interrupts can be used as software triggered interrupts.

Table 5-2 µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 interrupt/exception source list (1/4)

Type	Classification	Interrupt/Exception Source			Default Priority	Exception Code	Handler Address	Restored PC
		Name	Generating Source	Generating Unit				
Reset	Interrupt	RESET	$\overline{\text{RESET}}$ input	Pin	–	0000H	00000000H	undef.
Non-maskable	Interrupt	NMI0	NMI Input	PORT	–	0010H	00000010H	nextPC
		NMIWDT	Watchdog Timer	WDT	–	0020H	00000020H	nextPC
		NMI2	Unused	–	–	0030H	00000030H	nextPC
Software exception	Exception	TRAP0n (n = 0 to F_H)	TRAP instruction	–	–	004nH (n = 0 to F_H)	00000040H	nextPC
	Exception	TRAP1n (n = 0 to F_H)	TRAP instruction	–	–	005nH (n = 0 to F_H)	00000050H	nextPC
Exception trap	Exception	ILGOP/ DBTRAP	Illegal opcode/ DBTRAP instruction	–	–	0060H	00000060H	nextPC
Maskable	Interrupt	INTVC0	Voltage Comparator 0	AC0	0	0080H	00000080H	next PC
	Interrupt	INTVC1	Voltage Comparator 1	AC1	1	0090H	00000090H	next PC
	Interrupt	INTWT0UV	WT0 underflow	WT0	2	00A0H	000000A0H	next PC
	Interrupt	INTWT1UV	WT1 underflow	WT1	3	00B0H	000000B0H	next PC
	Interrupt	INTTM00	Watch calibration timer capture compare	WCT	4	00C0H	000000C0H	next PC
	Interrupt	Reserved	Reserved	—	5	00D0H	000000D0H	next PC
	Interrupt	INTP0	External interrupt 0	PORT	6	00E0H	000000E0H	next PC
	Interrupt	INTP1	External interrupt 1	PORT	7	00F0H	000000F0H	next PC
	Interrupt	INTP2	External interrupt 2	PORT	8	0100H	00000100H	next PC
	Interrupt	INTP3	External interrupt 3	PORT	9	0110H	00000110H	next PC
	Interrupt	INTP4	External interrupt 4	PORT	10	0120H	00000120H	next PC
	Interrupt	INTP5	External interrupt 5	PORT	11	0130H	00000130H	next PC
	Interrupt	INTP6	External interrupt 6	PORT	12	0140H	00000140H	next PC
	Interrupt	INTTZ0UV	TMZ0 underflow	TMZ0	13	0150H	00000150H	next PC
	Interrupt	INTTZ1UV	TMZ1 underflow	TMZ1	14	0160H	00000160H	next PC
	Interrupt	INTTZ2UV	TMZ2 underflow	TMZ2	15	0170H	00000170H	next PC
	Interrupt	INTTZ3UV	TMZ3 underflow	TMZ3	16	0180H	00000180H	next PC
	Interrupt	INTTZ4UV	TMZ4 underflow	TMZ4	17	0190H	00000190H	next PC
	Interrupt	INTTZ5UV	TMZ5 underflow	TMZ5	18	01A0H	000001A0H	next PC
	Interrupt	INTTP0OV	TMP0 overflow	TMP0	19	01B0H	000001B0H	next PC
	Interrupt	INTTP0CC0	TMP0 capture compare channel 0	TMP0	20	01C0H	000001C0H	next PC
	Interrupt	INTTP0CC1	TMP0 capture compare channel 1	TMP0	21	01D0H	000001D0H	next PC
	Interrupt	INTTP1OV	TMP1 overflow	TMP1	22	01E0H	000001E0H	next PC
	Interrupt	INTTP1CC0	TMP1 capture compare channel 0	TMP1	23	01F0H	000001F0H	next PC
	Interrupt	INTTP1CC1	TMP1 capture compare channel 1	TMP1	24	0200H	00000200H	next PC
	Interrupt	INTTP2OV	TMP2 overflow	TMP2	25	0210H	00000210H	next PC

Table 5-2 µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 interrupt/exception source list (2/4)

Type	Classification	Interrupt/Exception Source			Default Priority	Exception Code	Handler Address	Restored PC
		Name	Generating Source	Generating Unit				
Maskable	Interrupt	INTTP2CC0	TMP2 capture compare channel 0	TMP2	26	0220H	00000220H	next PC
	Interrupt	INTTP2CC1	TMP2 capture compare channel 1	TMP2	27	0230H	00000230H	next PC
	Interrupt	INTTP3OV	TMP3 overflow	TMP3	28	0240H	00000240H	next PC
	Interrupt	INTTP3CC0	TMP3 capture compare channel 0	TMP3	29	0250H	00000250H	next PC
	Interrupt	INTTP3CC1	TMP3 capture compare channel 1	TMP3	30	0260H	00000260H	next PC
	Interrupt	INTTG0OV0	TMG0 overflow interrupt 0	TMG0	31	0270H	00000270H	next PC
	Interrupt	INTTG0OV1	TMG0 overflow interrupt 1	TMG0	32	0280H	00000280H	next PC
	Interrupt	INTTG0CC0	TMG0 capture compare channel 0	TMG0	33	0290H	00000290H	next PC
	Interrupt	INTTG0CC1	TMG0 capture compare channel 1	TMG0	34	02A0H	000002A0H	next PC
	Interrupt	INTTG0CC2	TMG0 capture compare channel 2	TMG0	35	02B0H	000002B0H	next PC
	Interrupt	INTTG0CC3	TMG0 capture compare channel 3	TMG0	36	02C0H	000002C0H	next PC
	Interrupt	INTTG0CC4	TMG0 capture compare channel 4	TMG0	37	02D0H	000002D0H	next PC
	Interrupt	INTTG0CC5	TMG0 capture compare channel 5	TMG0	38	02E0H	000002E0H	next PC
	Interrupt	INTTG1OV0	TMG1 overflow interrupt 0	TMG1	39	02F0H	000002F0H	next PC
	Interrupt	INTTG1OV1	TMG1 overflow interrupt 1	TMG1	40	0300H	00000300H	next PC
	Interrupt	INTTG1CC0	TMG1 capture compare channel 0	TMG1	41	0310H	00000310H	next PC
	Interrupt	INTTG1CC1	TMG1 capture compare channel 1	TMG1	42	0320H	00000320H	next PC
	Interrupt	INTTG1CC2	TMG1 capture compare channel 2	TMG1	43	0330H	00000330H	next PC
	Interrupt	INTTG1CC3	TMG1 capture compare channel 3	TMG1	44	0340H	00000340H	next PC
	Interrupt	INTTG1CC4	TMG1 capture compare channel 4	TMG1	45	0350H	00000350H	next PC
	Interrupt	INTTG1CC5	TMG1 capture compare channel 5	TMG1	46	0360H	00000360H	next PC
	Interrupt	Reserved	Reserved	—	47	0370H	00000370H	next PC
	Interrupt	Reserved	Reserved	—	48	0380H	00000380H	next PC
	Interrupt	INTAD	ADC end of conversion	ADC	49	0390H	00000390H	next PC
	Interrupt	INTC0ERR	CAN0 error interrupt	CAN0	50	03A0H	000003A0H	next PC
	Interrupt	INTC0WUP	CAN0 wake up interrupt	CAN0	51	03B0H	000003B0H	next PC
	Interrupt	INTC0REC	CAN0 receive interrupt	CAN0	52	03C0H	000003C0H	next PC
	Interrupt	INTC0TRX	CAN0 transmit interrupt	CAN0	53	03D0H	000003D0H	next PC

Table 5-2 µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 interrupt/exception source list (3/4)

Type	Classification	Interrupt/Exception Source			Default Priority	Exception Code	Handler Address	Restored PC
		Name	Generating Source	Generating Unit				
Maskable	Interrupt	INTCB0RE	CSIB0 receive error interrupt	CSIB0	54	03E0H	000003E0H	next PC
	Interrupt	INTCB0R	CSIB0 receive complete interrupt	CSIB0	55	03F0H	000003F0H	next PC
	Interrupt	INTCB0T	CSIB0 transmit interrupt	CSIB0	56	0400H	00000400H	next PC
	Interrupt	INTUA0RE	UARTA0 receive error interrupt	UARTA0	57	0410H	00000410H	next PC
	Interrupt	INTUA0R	UARTA0 receive complete interrupt	UARTA0	58	0420H	00000420H	next PC
	Interrupt	INTUA0T	UARTA0 transmit interrupt	UARTA0	59	0430H	00000430H	next PC
	Interrupt	INTUA1RE	UARTA1 receive error interrupt	UARTA1	60	0440H	00000440H	next PC
	Interrupt	INTUA1R	UARTA1 receive complete interrupt	UARTA1	61	0450H	00000450H	next PC
	Interrupt	INTUA1T	UARTA1 transmit interrupt	UARTA1	62	0460H	00000460H	next PC
	Interrupt	INTIIC0	IIC0 interrupt	IIC0	63	0470H	00000470H	next PC
	Interrupt	INTIIC1	IIC1 interrupt	IIC1	64	0480H	00000480H	next PC
	Interrupt	Reserved	Reserved	—	65	0490H	00000490H	next PC
	Interrupt	INTDMA0	DMA0 transmission end	DMA0	66	04A0H	000004A0H	next PC
	Interrupt	INTDMA1	DMA1 transmission end	DMA1	67	04B0H	000004B0H	next PC
	Interrupt	INTDMA2	DMA2 transmission end	DMA2	68	04C0H	000004C0H	next PC
	Interrupt	INTDMA3	DMA3 transmission end	DMA3	69	04D0H	000004D0H	next PC
	Interrupt	INT70	not generated by hardware[a]	—	70	04E0H	000004E0H	next PC
	Interrupt	INT71	not generated by hardware[a]	—	71	04F0H	000004F0H	next PC
	Interrupt	INTP7	External interrupt 7	PORT	72	0500H	00000500H	next PC
	Interrupt	INTC1ERR	CAN1 error interrupt	CAN1	73	0510H	00000510H	next PC
	Interrupt	INTC1WUP	CAN1 wake up interrupt	CAN1	74	0520H	00000520H	next PC
	Interrupt	INTC1REC	CAN1 receive interrupt	CAN1	75	0530H	00000530H	next PC
	Interrupt	INTC1TRX	CAN1 transmit interrupt	CAN1	76	0540H	00000540H	next PC
	Interrupt	INTTZ6UV	TMZ6 underflow	TMZ6	77	0550H	00000550H	next PC
	Interrupt	INTTZ7CUV	TMZ7 underflow	TMZ7	78	0560H	00000560H	next PC
	Interrupt	INTTZ8UV	TMZ8 underflow	TMZ8	79	0570H	00000570H	next PC
	Interrupt	INTTZ9UV	TMZ9 underflow	TMZ9	80	0580H	00000580H	next PC

Table 5-2 µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 interrupt/ exception source list (4/4)

Type	Classification	Interrupt/Exception Source			Default Priority	Exception Code	Handler Address	Restored PC
		Name	Generating Source	Generating Unit				
Maskable	Interrupt	INTTG2OV0	TMG2 overflow interrupt 0	TMG2	81	0590H	00000590H	next PC
	Interrupt	INTTG2OV1	TMG2 overflow interrupt 1	TMG2	82	05A0H	000005A0H	next PC
	Interrupt	INTTG2CC0	TMG2 capture compare channel 0	TMG2	83	05B0H	000005B0H	next PC
	Interrupt	INTTG2CC1	TMG2 capture compare channel 1	TMG2	84	05C0H	000005C0H	next PC
	Interrupt	INTTG2CC2	TMG2 capture compare channel 2	TMG2	85	05D0H	000005D0H	next PC
	Interrupt	INTTG2CC3	TMG2 capture compare channel 3	TMG2	86	05E0H	000005E0H	next PC
	Interrupt	INTTG2CC4	TMG2 capture compare channel 4	TMG2	87	05F0H	000005F0H	next PC
	Interrupt	INTTG2CC5	TMG2 capture compare channel 5	TMG2	88	0600H	00000600H	next PC
	Interrupt	INTCB1RE	CSIB1 receive error interrupt	CSIB1	89	0610H	00000610H	next PC
	Interrupt	INTCB1R	CSIB1 receive complete interrupt	CSIB1	90	0620H	00000620H	next PC
	Interrupt	INTCB1T	CSIB1 transmit interrupt	CSIB1	91	0630H	00000630H	next PC
	Interrupt	INTCB2RE	CSIB2 receive error interrupt	CSIB2	92	0640H	00000640H	next PC
	Interrupt	INTCB2R	CSIB2 receive complete interrupt	CSIB2	93	0650H	00000650H	next PC
	Interrupt	INTCB2T	CSIB2 transmit interrupt	CSIB2	94	0660H	00000660H	next PC
	Interrupt	INTLCD	LCDBUSIF transmit interrupt	LCDBUSIF	95	0670H	00000670H	next PC

a) These interrupts can be used as software triggered interrupts.

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 two requests:

- NMI0: NMI pin input
- NMIWDT: Non-maskable Watchdog Timer interrupt request

When the valid edge specified by the ESEL0, ESEL00 and ESEL01 bits of the Interrupt mode register 0(INTM0) is detected on the NMI pin, the interrupt occurs.

The Watchdog Timer interrupt request is only effective as non-maskable interrupt if the WDTMODE bit of the Watchdog Timer mode register (WDTM) is set 0.

If multiple non-maskable interrupts are generated at the same time, the highest priority servicing is executed according to the following priority order (the lower priority interrupt is ignored):

NMIWDT > NMI0

Note that if a NMI from port pin or NMIWDT request is generated while NMI from port pin is being serviced, the service is executed as follows.

(1) If a NMI0 is generated while NMI0 is being serviced

The new NMI0 request is held pending regardless of the value of the PSW.NP bit. The pending NMIVC request is acknowledged after servicing of the current NMI0 request has finished (after execution of the RETI instruction).

(2) If a NMIWDT request is generated while NMI0 is being serviced

If the PSW.NP bit remains set (1) while NMI0 is being serviced, the new NMIWDT request is held pending. The pending NMIWDT request is acknowledge after servicing of the current NMI0 request has finished (after execution of the RETI instruction).

If the PSW.NP bit is cleared (0) while NMI0 is being serviced, the newly generated NMIWDT request is executed (NMI0 servicing is halted).

Caution

1. Although the values of the PC and PSW are saved to an NMI status save register (FEPC, FEPSW) when a non-maskable interrupt request is generated, only the NMI0 can be restored by the RETI instruction at this time. Because NMIWDT cannot be restored by the RETI instruction, the system must be reset after servicing this interrupt.
2. If PSW.NP is cleared to 0 by the LDSR instruction during non-maskable interrupt servicing, a NMI0 interrupt afterwards cannot be acknowledged correctly.

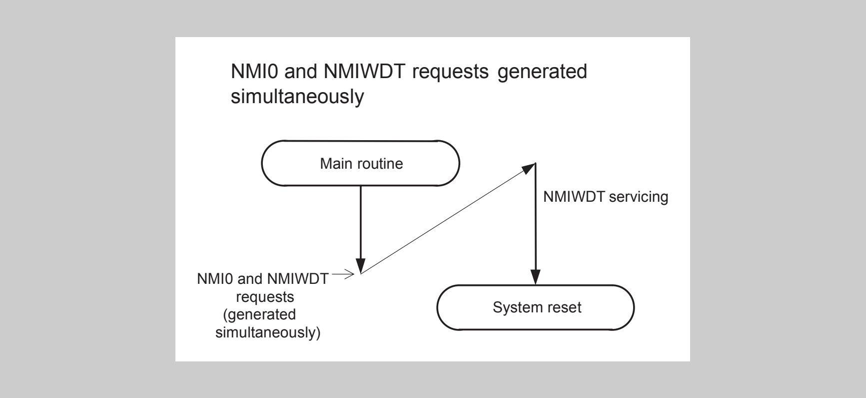


Figure 5-1 Example of non-maskable interrupt request acknowledgement operation: multiple NMI requests generated at the same time

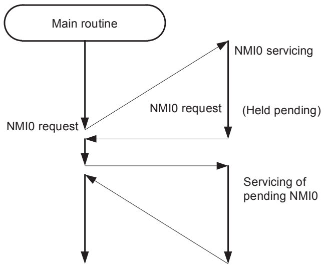
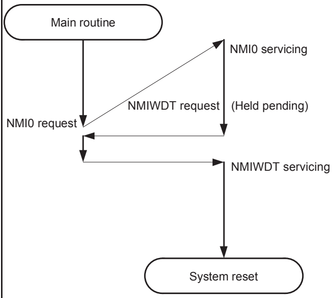
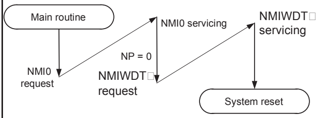
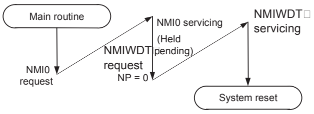
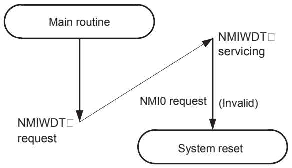
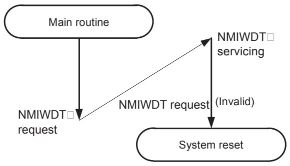
NMI being serviced	NMI request generated during NMI servicing	
	NMI0	NMIWDT
NMI0	NMI0 request generated during NMI0 servicing  	NMIWDT request generated during NMI0 servicing (NP = 1 retained before NMI1 request)  NMIWDT request generated during NMI0 servicing (NP=0 set before NMIWDT request)  NMIWDT request generated during NMI0 servicing (NP=0 set after NMIWDT request) 
NMIWDT	NMI0 request generated during NMIWDT servicing  	NMI1 request generated during NMIWDT servicing 

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 0010H 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*.

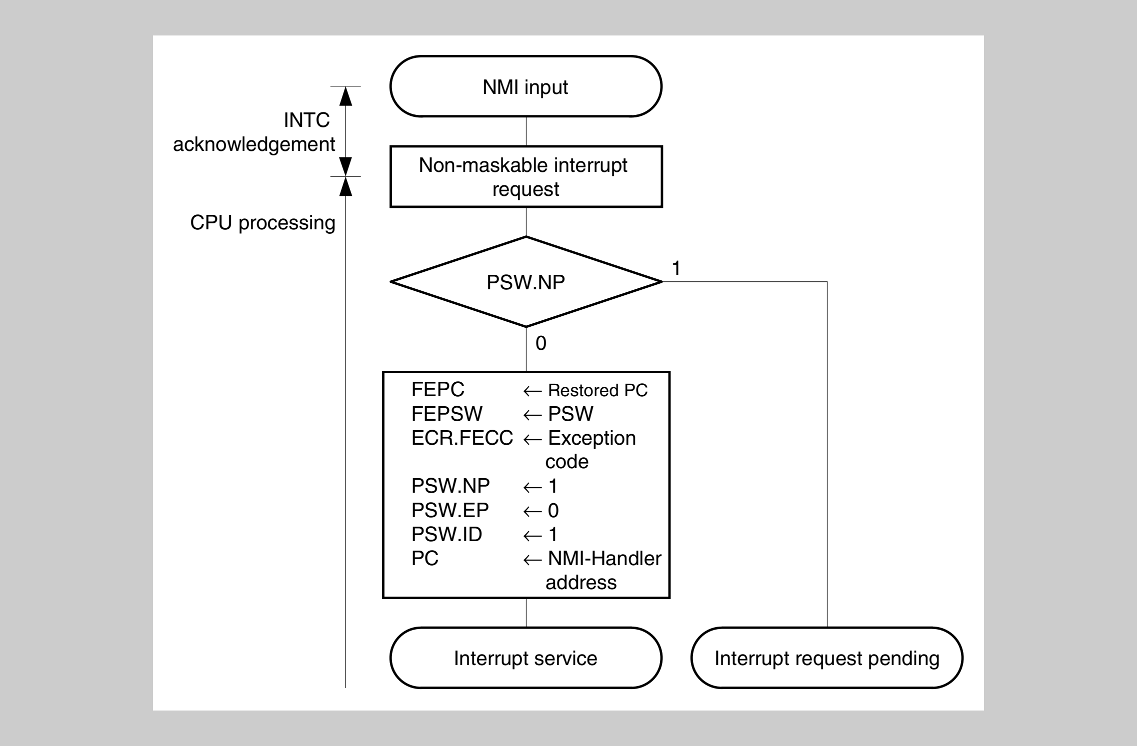


Figure 5-3 Processing configuration of non-maskable interrupt

5.2.2 Restore

(1) NMI0

Execution is restored from the non-maskable interrupt (NMI0) processing by the RETI instruction.

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

<1> Restores the values of the PC and the PSW from FEPC and FEPSW, respectively, because the EP bit of the PSW is 0 and the NP bit of the PSW is 1.

<2> Transfers control back to the address of the restored PC and PSW.

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

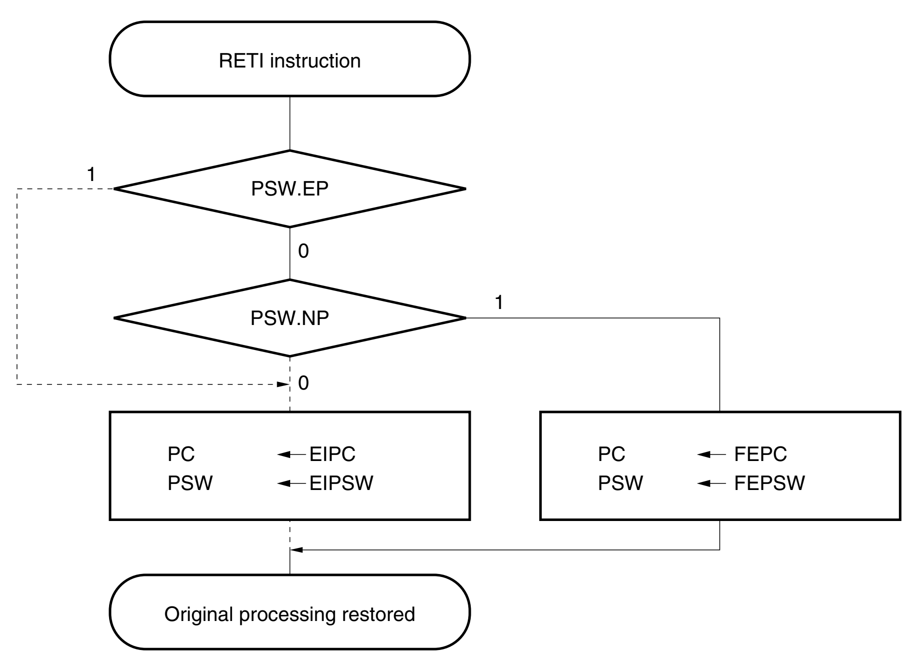


Figure 5-4 RETI instruction processing

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) NMIWDT

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.

	31																							8	7	6	5	4	3	2	1	0	Initial value
PSW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NP	EP	ID	SAT	CY	OV	S	Z	00000020H

Bit position	Bit name	Function
7	NP	Indicates whether NMI interrupt processing is in progress. 0: No NMI interrupt processing 1: NMI interrupt currently being processed

5.2.4 NMI0 control

The NMI0 can be configured to generate an NMI upon a rising, falling or both edges at the NMI pin. To enable respectively disable the NMI0 and to configure the edge refer to *“Edge and Level Detection Configuration“ on page 218*.

5.3 Maskable Interrupts

Maskable interrupt requests can be masked by interrupt control registers.

If two or more maskable interrupt requests are generated at the same time, they are acknowledged according to the default priority. In addition to the default priority, eight levels of priorities can be specified by using the interrupt control registers (programmable priority control).

When an interrupt request has been acknowledged, the acknowledgement of other maskable interrupt requests is disabled and the interrupt disabled (DI) status is set.

When the EI instruction is executed in an interrupt processing routine, the interrupt enabled (EI) status is set, which enables servicing of interrupts having a higher priority than the interrupt request in progress (specified by the interrupt control register). Note that only interrupts with a higher priority will have this capability; interrupts with the same priority level cannot be nested.

However, if multiple interrupts are executed, the following processing is necessary.

(1) Save EIPC and EIPSW in memory or a general-purpose register before executing the EI instruction.

(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 by INT input, the CPU performs the following processing, and transfers control to a handler routine:

(1) Saves the restored PC to EIPC.

(2) Saves the current PSW to EIPSW.

(3) Writes an exception code to the lower halfword of ECR (EICC).

(4) Sets the ID bit of the PSW and clears the EP bit.

(5) Sets the handler address corresponding to each interrupt to the PC, and transfers control.

The processing configuration of a maskable interrupt is shown in *Figure 5-5*.

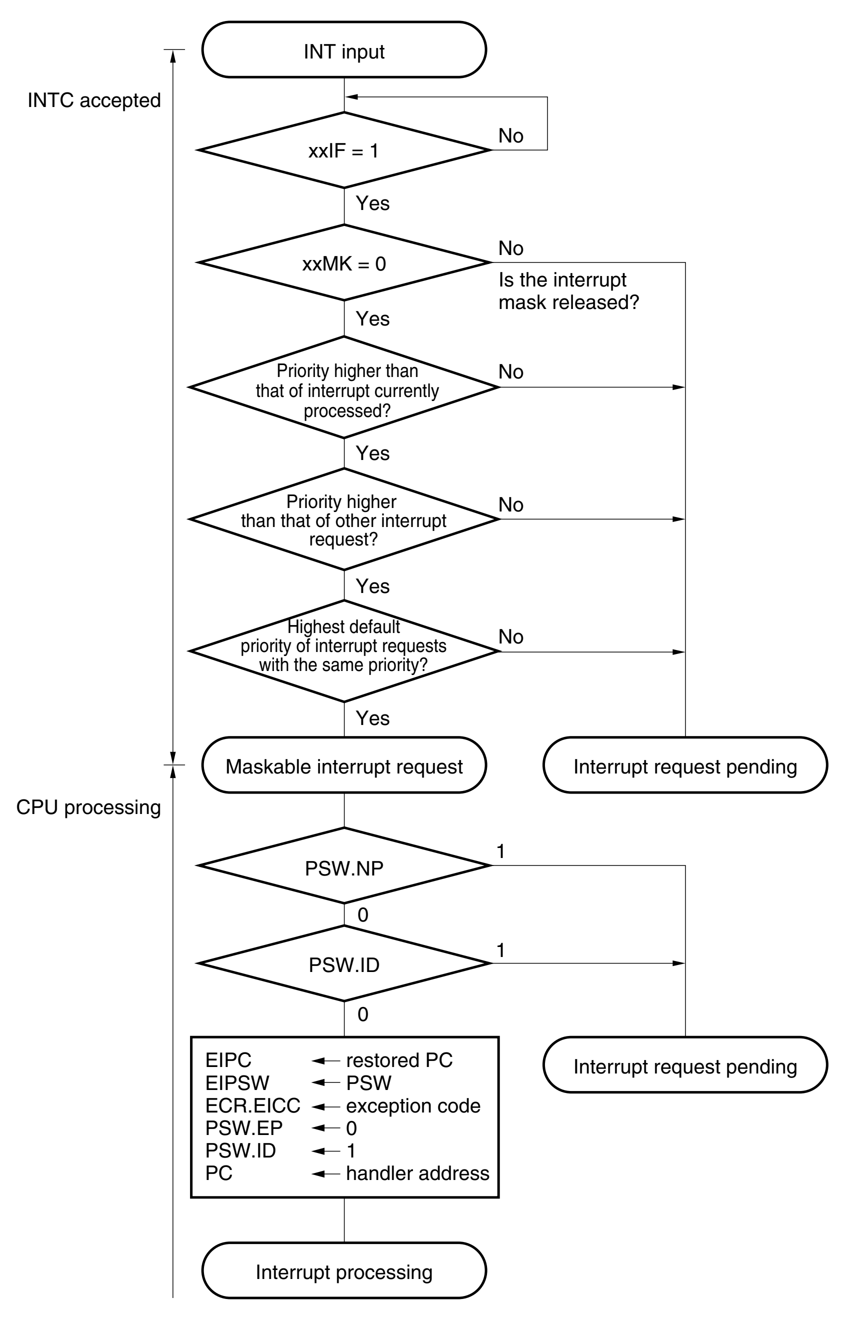


Figure 5-5 **Maskable interrupt processing**

Note For the ISPR register, see *“ISPR - In-service priority register“ on page 216*.

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.

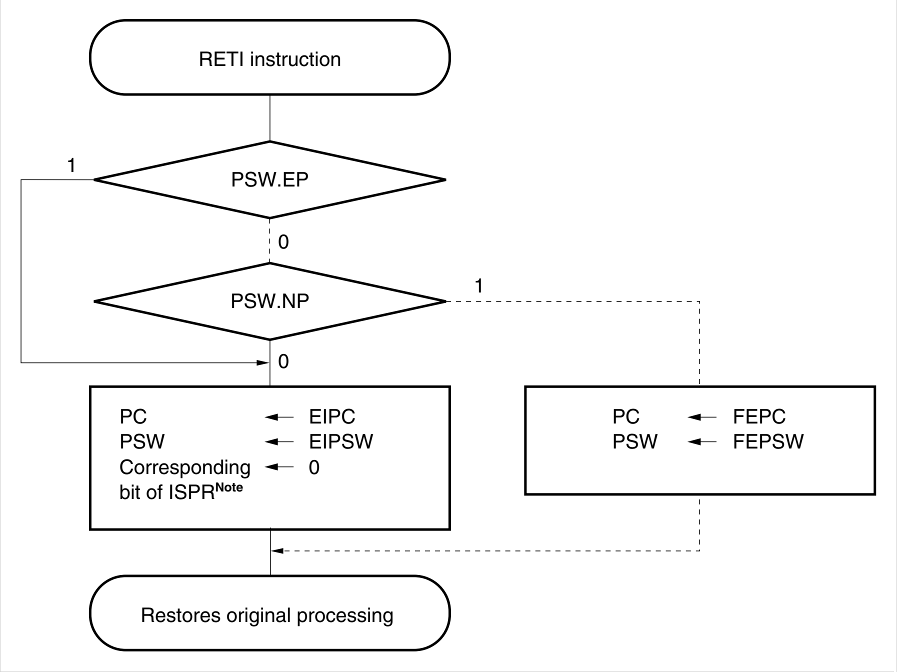


Figure 5-6 **RETI instruction processing**

Note 1. For the ISPR register, see *"ISPR - In-service priority register" on page 216*.

2. The solid lines show the CPU processing flow.

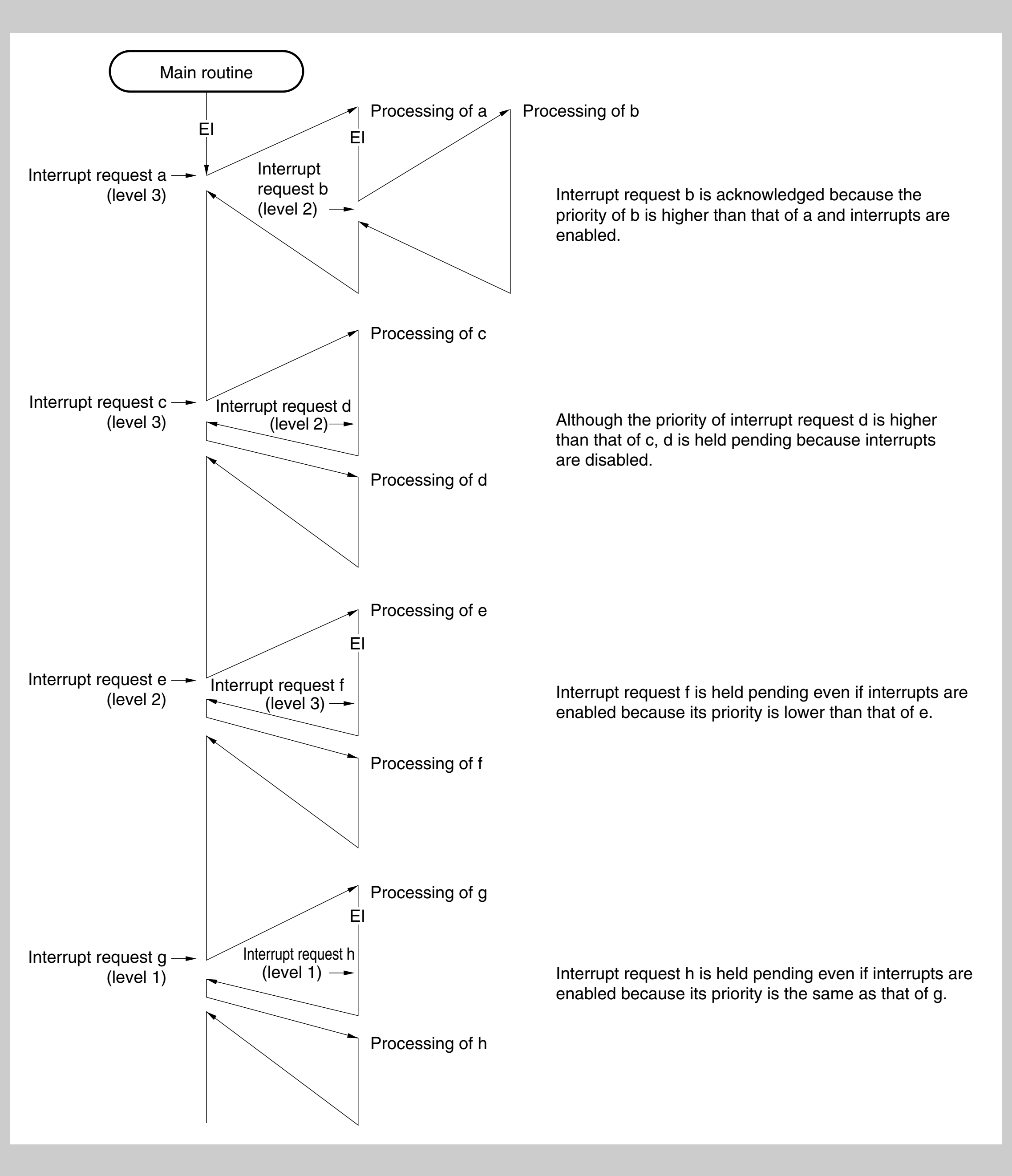
Caution When the PSW.EP bit and the PSW.NP bit are changed by the LDSR instruction during maskable interrupt processing, in order to restore the PC and PSW correctly during recovery by the RETI instruction, it is necessary to set PSW.EP back to 0 and PSW.NP back to 0 using the LDSR instruction immediately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



Interrupt request b is acknowledged because the priority of b is higher than that of a and interrupts are enabled.

Although the priority of interrupt request d is higher than that of c, d is held pending because interrupts are disabled.

Interrupt request f is held pending even if interrupts are enabled because its priority is lower than that of e.

Interrupt request h is held pending even if interrupts are enabled because its priority is the same as that of g.

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.

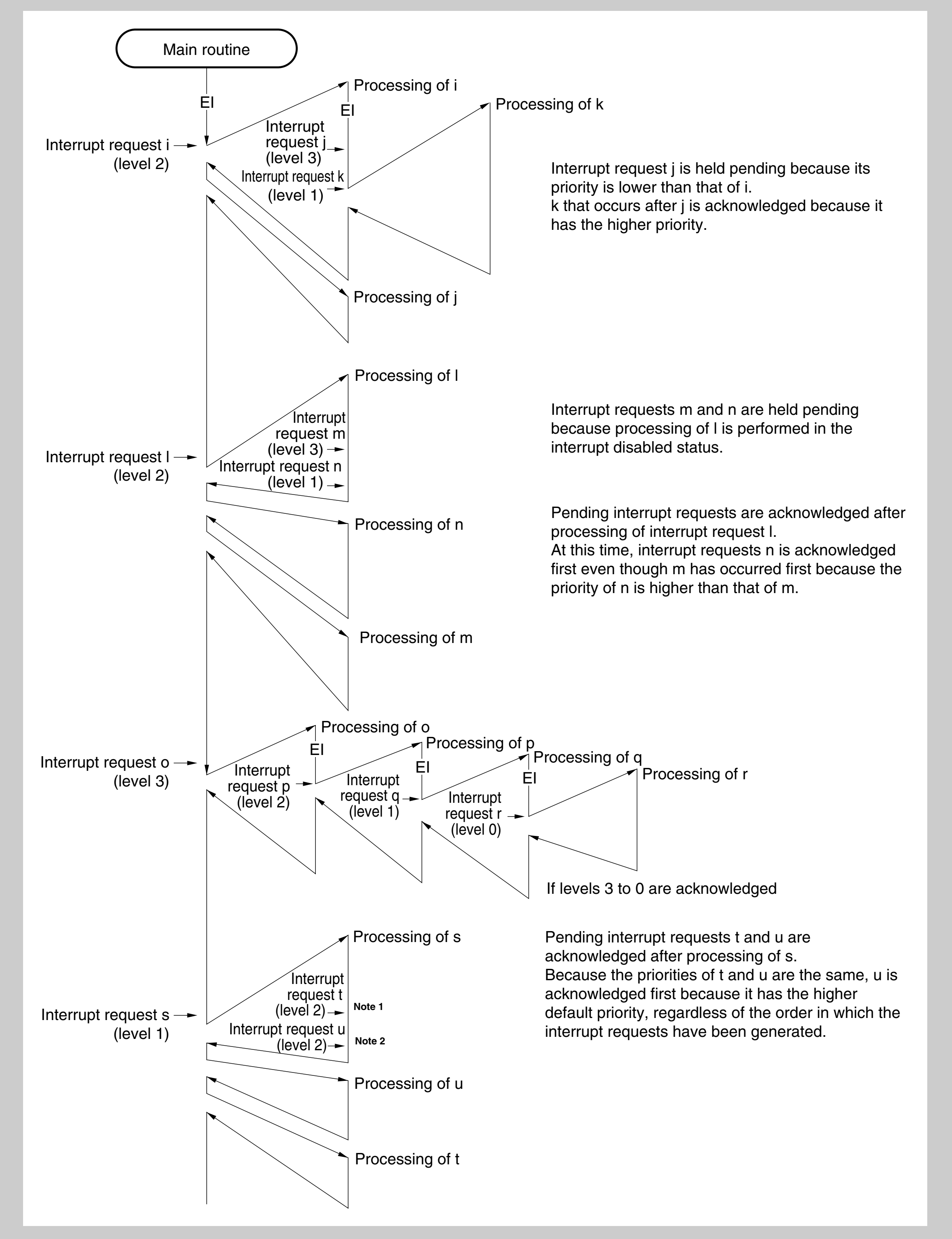


Figure 5-8 Example of processing in which another interrupt request is issued while an interrupt is being processed (2/2)

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

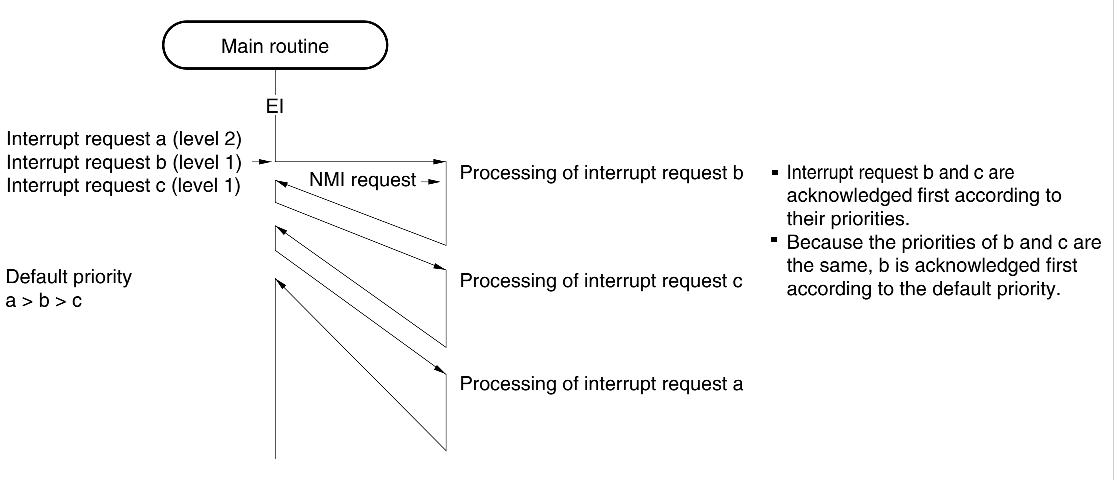


Figure 5-9 Example of processing interrupt requests simultaneously generated

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.

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

5.3.4 xxIC - Maskable interrupts control register

An interrupt control register is assigned to each interrupt request (maskable interrupt) and sets the control conditions for each maskable interrupt request.

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

	7	6	5	4	3	2	1	0	Address	Initial value
xxIC	xxIF	xxMK	0	0	0	xxPR2	xxPR1	xxPR0	FFFF F110H to FFFF F18EH	47H

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>xxIF</td><td>This is an interrupt request flag.
0: Interrupt request not issued
1: Interrupt request issued
The flag xxIFn is reset automatically by the hardware if an interrupt request is acknowledged.</td></tr>
<tr><td>6</td><td>xxMK</td><td>This is an interrupt mask flag.
0: Enables interrupt processing
1: Disables interrupt processing (pending)</td></tr>
<tr><td>2 to 0</td><td>xxPR2 to xxPR0</td><td>8 levels of priority order are specified for each interrupt.
<table>
<tr><th>xxPR2</th><th>xxPR1</th><th>xxPR0</th><th>Interrupt priority specification bit</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Specifies level 0 (highest)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Specifies level 1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Specifies level 2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Specifies level 3</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Specifies level 4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Specifies level 5</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Specifies level 6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Specifies level 7 (lowest)</td></tr>
</table></td></tr>
</table>

Note xx: identification name of each peripheral unit (VC0-VC1, WT0UV-WT1UV, TM01, P0-P7, TZ0UV-TZ9UV, TP0OV-TP3OV, TP0CC0-TP3CC0, TP0CC1-TP3CC1, TG0OV0-TG2OV0, TG0OV1-TG2OV1, TG0CC0-TG2CC0, TG0CC1-TG2CC1, TG0CC2-TG2CC2, TG0CC3-TG2CC3, TG0CC4-TG2CC4, TG0CC5-TG2CC5, AD, C0ERR, C1ERR, C0WUP, C1WUP, C0REC, C1REC, C0TRX, C1TRX, CB0RE-CB2RE, CB0R-CB2R, CB0T-CB2T, UA0RE-UA1RE, UA0R-UA1R, UA0T-UA1T, IIC0-IIC1, DMA0-DMA3, INT70, INT71, LCD)

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

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

Address	Register	Bit							
		7	6	5	4	3	2	1	0
FFFFF110H	VC0IC	VC0IF	VC0MK	0	0	0	VC0PR2	VC0PR1	VC0PR0
FFFFF112H	VC1IC	VC1IF	VC1MK	0	0	0	VC1PR2	VC1PR1	VC1PR0
FFFFF114H	WT0UVIC	WT0UVIF	WT0UVMK	0	0	0	WT0UVPR2	WT0UVPR1	WT0UVPR0
FFFFF116H	WT1UVIC	WT1UVIF	WT1UVMK	0	0	0	WT1UVPR2	WT1UVPR1	WT1UVPR0
FFFFF118H	TM00IC	TM00IF	TM00MK	0	0	0	TM00PR2	TM00PR1	TM00PR0
FFFFF11CH	P0IC	P0IF	P0MK	0	0	0	P0PR2	P0PR1	P0PR0
FFFFF11EH	P1IC	P1IF	P1MK	0	0	0	P1PR2	P1PR1	P1PR0
FFFFF120H	P2IC	P2IF	P2MK	0	0	0	P2PR2	P2PR1	P2PR0
FFFFF122H	P3IC	P3IF	P3MK	0	0	0	P3PR2	P3PR1	P3PR0
FFFFF124H	P4IC	P4IF	P4MK	0	0	0	P4PR2	P4PR1	P4PR0
FFFFF126H	P5IC	P5IF	P5MK	0	0	0	P5PR2	P5PR1	P5PR0
FFFFF128H	P6IC	P6IF	P6MK	0	0	0	P6PR2	P6PR1	P6PR0
FFFFF12AH	TZ0UVIC	TZ0UVIF	TZ0UVMK	0	0	0	TZ0UVPR2	TZ0UVPR1	TZ0UVPR0
FFFFF12CH	TZ1UVIC	TZ1UVIF	TZ1UVMK	0	0	0	TZ1UVPR2	TZ1UVPR1	TZ1UVPR0
FFFFF12EH	TZ2UVIC	TZ2UVIF	TZ2UVMK	0	0	0	TZ2UVPR2	TZ2UVPR1	TZ2UVPR0
FFFFF130H	TZ3UVIC	TZ3UVIF	TZ3UVMK	0	0	0	TZ3UVPR2	TZ3UVPR1	TZ3UVPR0
FFFFF132H	TZ4UVIC	TZ4UVIF	TZ4UVMK	0	0	0	TZ4UVPR2	TZ4UVPR1	TZ4UVPR0
FFFFF134H	TZ5UVIC	TZ5UVIF	TZ5UVMK	0	0	0	TZ5UVPR2	TZ5UVPR1	TZ5UVPR0
FFFFF136H	TP0OVIC	TP0OVIF	TP0OVMK	0	0	0	TP0OVPR2	TP0OVPR1	TP0OVPR0
FFFFF138H	TP0CC0IC	TP0CC0IF	TP0CC0MK	0	0	0	TP0CC0PR2	TP0CC0PR1	TP0CC0PR0
FFFFF13AH	TP0CC1IC	TP0CC1IF	TP0CC1MK	0	0	0	TP0CC1PR2	TP0CC1PR1	TP0CC1PR0
FFFFF13CH	TP1OVIC	TP1OVIF	TP1OVMK	0	0	0	TP1OVPR2	TP1OVPR1	TP1OVPR0
FFFFF13EH	TP1CC0IC	TP1CC0IF	TP1CC0MK	0	0	0	TP1CC0PR2	TP1CC0PR1	TP1CC0PR0
FFFFF140H	TP1CC1IC	TP1CC1IF	TP1CC1MK	0	0	0	TP1CC1PR2	TP1CC1PR1	TP1CC1PR0
FFFFF142H	TP2OVIC	TP2OVIF	TP2OVMK	0	0	0	TP2OVPR2	TP2OVPR1	TP2OVPR0
FFFFF144H	TP2CC0IC	TP2CC0IF	TP2CC0MK	0	0	0	TP2CC0PR2	TP2CC0PR1	TP2CC0PR0
FFFFF146H	TP2CC1IC	TP2CC1IF	TP2CC1MK	0	0	0	TP2CC1PR2	TP2CC1PR1	TP2CC1PR0
FFFFF148H	TP3OVIC	TP3OVIF	TP3OVMK	0	0	0	TP3OVPR2	TP3OVPR1	TP3OVPR0
FFFFF14AH	TP3CC0IC	TP3CC0IF	TP3CC0MK	0	0	0	TP3CC0PR2	TP3CC0PR1	TP3CC0PR0
FFFFF14CH	TP3CC1IC	TP3CC1IF	TP3CC1MK	0	0	0	TP3CC1PR2	TP3CC1PR1	TP3CC1PR0
FFFFF14EH	TG0OV0IC	TG0OV0IF	TG0OV0MK	0	0	0	TG0OV0PR2	TG0OV0PR1	TG0OV0PR0
FFFFF150H	TG0OV1IC	TG0OV1IF	TG0OV1MK	0	0	0	TG0OV1PR2	TG0OV1PR1	TG0OV1PR0
FFFFF152H	TG0CC0IC	TG0CC0IF	TG0CC0MK	0	0	0	TG0CC0PR2	TG0CC0PR1	TG0CC0PR0
FFFFF154H	TG0CC1IC	TG0CC1IF	TG0CC1MK	0	0	0	TG0CC1PR2	TG0CC1PR1	TG0CC1PR0
FFFFF156H	TG0CC2IC	TG0CC2IF	TG0CC2MK	0	0	0	TG0CC2PR2	TG0CC2PR1	TG0CC2PR0
FFFFF158H	TG0CC3IC	TG0CC3IF	TG0CC3MK	0	0	0	TG0CC3PR2	TG0CC3PR1	TG0CC3PR0
FFFFF15AH	TG0CC4IC	TG0CC4IF	TG0CC4MK	0	0	0	TG0CC4PR2	TG0CC4PR1	TG0CC4PR0
FFFFF15CH	TG0CC5IC	TG0CC5IF	TG0CC5MK	0	0	0	TG0CC5PR2	TG0CC5PR1	TG0CC5PR0
FFFFF15EH	TG1OV0IC	TG1OV0IF	TG1OV0MK	0	0	0	TG1OV0PR2	TG1OV0PR1	TG1OV0PR0
FFFFF160H	TG1OV1IC	TG1OV1IF	TG1OV1MK	0	0	0	TG1OV1PR2	TG1OV1PR1	TG1OV1PR0

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

Address	Register	Bit							
		7	6	5	4	3	2	1	0
FFFFF162H	TG1CC0IC	TG1CC0IF	TG1CC0MK	0	0	0	TG1CC0PR2	TG1CC0PR1	TG1CC0PR0
FFFFF164H	TG1CC1IC	TG1CC1IF	TG1CC1MK	0	0	0	TG1CC1PR2	TG1CC1PR1	TG1CC1PR0
FFFFF166H	TG1CC2IC	TG1CC2IF	TG1CC2MK	0	0	0	TG1CC2PR2	TG1CC2PR1	TG1CC2PR0
FFFFF168H	TG1CC3IC	TG1CC3IF	TG1CC3MK	0	0	0	TG1CC3PR2	TG1CC3PR1	TG1CC3PR0
FFFFF16AH	TG1CC4IC	TG1CC4IF	TG1CC4MK	0	0	0	TG1CC4PR2	TG1CC4PR1	TG1CC4PR0
FFFFF16CH	TG1CC5IC	TG1CC5IF	TG1CC5MK	0	0	0	TG1CC5PR2	TG1CC5PR1	TG1CC5PR0
FFFFF172H	ADIC	ADIF	ADMK	0	0	0	ADPR2	ADPR1	ADPR0
FFFFF174H	C0ERRIC	C0ERRIF	C0ERRMK	0	0	0	C0ERRPR2	C0ERRPR1	C0ERRPR0
FFFFF176H	C0WUPIC	C0WUPIF	C0WUPMK	0	0	0	C0WUPPR2	C0WUPPR1	C0WUPPR0
FFFFF178H	C0RECIC	C0RECIF	C0RECMK	0	0	0	C0RECPR2	C0RECPR1	C0RECPR0
FFFFF17AH	C0TRXIC	C0TRXIF	C0TRXMK	0	0	0	C0TRXPR2	C0TRXPR1	C0TRXPR0
FFFFF17CH	CB0REIC	CB0REIF	CB0REMK	0	0	0	CB0REPR2	CB0REPR1	CB0REPR0
FFFFF17EH	CB0RIC	CB0RIF	CB0RMK	0	0	0	CB0RPR2	CB0RPR1	CB0RPR0
FFFFF180H	CB0TIC	CB0TIF	CB0TMK	0	0	0	CB0TPR2	CB0TPR1	CB0TPR0
FFFFF182H	UA0REIC	UA0REIF	UA0REMK	0	0	0	UA0REPR2	UA0REPR1	UA0REPR0
FFFFF184H	UA0RIC	UA0RIF	UA0RMK	0	0	0	UA0RPR2	UA0RPR1	UA0RPR0
FFFFF186H	UA0TIC	UA0TIF	UA0TMK	0	0	0	UA0TPR2	UA0TPR1	UA0TPR0
FFFFF188H	UA1REIC	UA1REIF	UA1REMK	0	0	0	UA1REPR2	UA1REPR1	UA1REPR0
FFFFF18AH	UA1RIC	UA1RIF	UA1RMK	0	0	0	UA1RPR2	UA1RPR1	UA1RPR0
FFFFF18CH	UA1TIC	UA1TIF	UA1TMK	0	0	0	UA1TPR2	UA1TPR1	UA1TPR0
FFFFF18EH	IIC0IC	IIC0IF	IIC0MK	0	0	0	IIC0PR2	IIC0PR1	IIC0PR0
FFFFF190H	IIC1IC	IIC1IF	IIC1MK	0	0	0	IIC1PR2	IIC1PR1	IIC1PR0
FFFFF194H	DMA0IC	DMA0IF	DMA0MK	0	0	0	DMA0PR2	DMA0PR1	DMA0PR0
FFFFF196H	DMA1IC	DMA1IF	DMA1MK	0	0	0	DMA1PR2	DMA1PR1	DMA1PR0
FFFFF198H	DMA2IC	DMA2IF	DMA2MK	0	0	0	DMA2PR2	DMA2PR1	DMA2PR0
FFFFF19AH	DMA3IC	DMA3IF	DMA3MK	0	0	0	DMA3PR2	DMA3PR1	DMA3PR0
FFFFF19CH	INT70IC	INT70IF	INT70MK	0	0	0	INT70PR2	INT70PR1	INT70PR0
FFFFF19EH	INT71IC	INT71IF	INT71MK	0	0	0	INT71PR2	INT71PR1	INT71PR0
FFFFF1A0H	P7IC[a]	P7IF	P7MK	0	0	0	P7PR2	P7PR1	P7PR0
FFFFF1A2H	C1ERRIC	C1ERRIF	C1ERRMK	0	0	0	C1ERRPR2	C1ERRPR1	C1ERRPR0
FFFFF1A4H	C1WUPIC	C1WUPIF	C1WUPMK	0	0	0	C1WUPPR2	C1WUPPR1	C1WUPPR0
FFFFF1A6H	C1RECIC	C1RECIF	C1RECMK	0	0	0	C1RECPR2	C1RECPR1	C1RECPR0
FFFFF1A8H	C1TRXIC	C1TRXIF	C1TRXMK	0	0	0	C1TRXPR2	C1TRXPR1	C1TRXPR0
FFFFF1AAH	TZ6UVIC[a]	TZ6UVIF	TZ6UVMK	0	0	0	TZ6UVPR2	TZ6UVPR1	TZ6UVPR0
FFFFF1ACH	TZ7UVIC[a]	TZ7UVIF	TZ7UVMK	0	0	0	TZ7UVPR2	TZ7UVPR1	TZ7UVPR0
FFFFF1AEH	TZ8UVIC[a]	TZ8UVIF	TZ8UVMK	0	0	0	TZ8UVPR2	TZ8UVPR1	TZ8UVPR0
FFFFF1B0H	TZ9UVIC[a]	TZ9UVIF	TZ9UVMK	0	0	0	TZ9UVPR2	TZ9UVPR1	TZ9UVPR0
FFFFF1B2H	TG2OV0IC	TG2OV0IF	TG2OV0MK	0	0	0	TG2OV0PR2	TG2OV0PR1	TG2OV0PR0
FFFFF1B4H	TG2OV1IC	TG2OV1IF	TG2OV1MK	0	0	0	TG2OV1PR2	TG2OV1PR1	TG2OV1PR0
FFFFF1B6H	TG2CC0IC	TG2CC0IF	TG2CC0MK	0	0	0	TG2CC0PR2	TG2CC0PR1	TG2CC0PR0

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

Address	Register	Bit							
		7	6	5	4	3	2	1	0
FFFFF1B8H	TG2CC1IC	TG2CC1IF	TG2CC1MK	0	0	0	TG2CC1PR2	TG2CC1PR1	TG2CC1PR0
FFFFF1BAH	TG2CC2IC	TG2CC2IF	TG2CC2MK	0	0	0	TG2CC2PR2	TG2CC2PR1	TG2CC2PR0
FFFFF1BCH	TG2CC3IC	TG2CC3IF	TG2CC3MK	0	0	0	TG2CC3PR2	TG2CC3PR1	TG2CC3PR0
FFFFF1BEH	TG2CC4IC	TG2CC4IF	TG2CC4MK	0	0	0	TG2CC4PR2	TG2CC4PR1	TG2CC4PR0
FFFFF1C0H	TG2CC5IC	TG2CC5IF	TG2CC5MK	0	0	0	TG2CC5PR2	TG2CC5PR1	TG2CC5PR0
FFFFF1C2H	CB1REIC	CB1REIF	CB1REMK	0	0	0	CB1REPR2	CB1REPR1	CB1REPR0
FFFFF1C4H	CB1RIC	CB1RIF	CB1RMK	0	0	0	CB1RPR2	CB1RPR1	CB1RPR0
FFFFF1C6H	CB1TIC	CB1TIF	CB1TMK	0	0	0	CB1TPR2	CB1TPR1	CB1TPR0
FFFFF1C8H	CB2REIC[a]	CB2REIF	CB2REMK	0	0	0	CB2REPR2	CB2REPR1	CB2REPR0
FFFFF1CAH	CB2RIC[a]	CB2RIF	CB2RMK	0	0	0	CB2RPR2	CB2RPR1	CB2RPR0
FFFFF1CCH	CB2TIC[a]	CB2TIF	CB2TMK	0	0	0	CB2TPR2	CB2TPR1	CB2TPR0
FFFFF1CEH	LCDIC	LCDIF	LCDMK	0	0	0	LCDPR2	LCDPR1	LCDPR0

a) µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 only

5.3.5 IMR0 to IMR5 - Interrupt mask registers

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

The xxMK bit of the IMRm (m = 0 to 5) registers is equivalent to the xxMK bit of the xxIC register.

IMRm registers can be read/written in 16- and 8-bit units.

The address of the lower 8-bit register IMRmL is equal to that of the 16-bit IMRm register, and the higher 8-bit register IMRmH can be accessed on the following address (address (IMRm) + 1).

Caution Mask bits without function, indicated with “1”, must not be altered. Make sure to set them “1” when writing to the register.

	15	14	13	12	11	10	9	8	Address	Initial value
IMR0	TZ2UVMK	TZ1UVMK	TZ0UVMK	P6MK	P5MK	P4MK	P3MK	P2MK	FFFFF100H	FFFFH
	7	6	5	4	3	2	1	0		
	P1MK	P0MK	1	TM00MK	WT1UVMK	WT0UVMK	VC1MK	VC0MK		

	15	14	13	12	11	10	9	8	Address	Initial value
IMR1	TG0OV0MK	TP3CC1MK	TP3CC0MK	TP3OVMK	TP2CC1MK	TP2CC0MK	TP2OVMK	TP1CC1MK	FFFFF102H	FFFFH
	7	6	5	4	3	2	1	0		
	TP1CC0MK	TP1OVMK	TP0CC1MK	TP0CC0MK	TP0OVMK	TZ5UVMK	TZ4UVMK	TZ3UVMK		

	15	14	13	12	11	10	9	8	Address	Initial value
IMR2	TY0UV0MK	TG1CC5MK	TG1CC4MK	TG1CC3MK	TG1CC2MK	TG1CC1MK	TG1CC0MK	TG1OV1MK	FFFFF104H	FFFFH
	7	6	5	4	3	2	1	0		
	TG1OV0MK	TG0CC5MK	TG0CC4MK	TG0CC3MK	TG0CC2MK	TG0CC1MK	TG0CC0MK	TG0OV1MK		

	15	14	13	12	11	10	9	8	Address	Initial value
IMR3	IIC0MK	UA1TMK	UA1RMK	UA1REMK	UA0TMK	UA0RMK	UA0REMK	CB0TMK	FFFFF106H	FFFFH
	7	6	5	4	3	2	1	0		
	CB0RMK	CB0REMK	C0TRXMK	C0RECMK	C0WUPMK	C0ERRMK	ADMK	TY0UV1MK		

For µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 only:

	15	14	13	12	11	10	9	8	Address	Initial value
IMR4	1	1	1	C1TRXMK	C1RECMK	C1WUPMK	C1ERRMK	1	FFFFF108H	FFFFH
	7	6	5	4	3	2	1	0		
	INT71MK	INT70MK	DMA3MK	DMA2MK	DMA1MK	DMA0MK	SG0MK	IIC1MK		

	15	14	13	12	11	10	9	8	Address	Initial value
IMR5	LCDMK	1	1	1	CB1TMK	CB1RMK	CB1REMK	TG2CC5MK	FFFFF10AH	FFFFH
	7	6	5	4	3	2	1	0		
	TG2CC4MK	TG2CC3MK	TG2CC2MK	TG2CC1MK	TG2CC0MK	TG2OV1MK	TG2OV0MK	1		

For µPD70F3424, µPD70F3425, µPD70F3426, µPD70F3427 only:

	15	14	13	12	11	10	9	8	Address	Initial value
IMR4	TZ8UVMK	TZ7UVMK	TZ6UVMK	C1TRXMK	C1RECMK	C1WUPMK	C1ERRMK	P7MK	FFFFF108H	FFFFH
	7	6	5	4	3	2	1	0		
	INT71MK	INT70MK	DMA3MK	DMA2MK	DMA1MK	DMA0MK	SG0MK	IIC1MK		

	15	14	13	12	11	10	9	8	Address	Initial value
IMR5	LCDMK	CB2TMK	CB2RMK	CB2REMK	CB1TMK	CB1RMK	CB1REMK	TG2CC5MK	FFFFF10AH	FFFFH
	7	6	5	4	3	2	1	0		
	TG2CC4MK	TG2CC3MK	TG2CC2MK	TG2CC1MK	TG2CC0MK	TG2OV1MK	TG2OV0MK	TZ9UVMK		

Bit position	Bit name	Function
15 to 0	xxMK	Interrupt mask flag. 0: Interrupt servicing enabled 1: Interrupt servicing disabled (pending)

Note xx: identification name of each peripheral unit (VC0-VC1, WT0UV-WT1UV, TM00, P0-P7, TZ0UV-TZ9UV, TP0OV-TP3OV, TP0CC0-TP3CC0, TP0CC1-TP3CC1, TG0OV0-TG2OV0, TG0OV1-TG2OV1, TG0CC0-TG2CC0, TG0CC1-TG2CC1, TG0CC2-TG2CC2, TG0CC3-TG2CC3, TG0CC4-TG2CC4, TG0CC5-TG2CC5, AD, C0ERR, C1ERR, C0WUP, C1WUP, C0REC, C1REC, C0TRX, C1TRX, CB0RE-CB2RE, CB0R-CB2R, CB0T-CB2T, UA0RE-UA1RE, UA0R-UA1R, UA0T-UA1T, IIC0-IIC1, DMA0-DMA3, INT70, INT71, LCD)

5.3.6 ISPR - In-service priority register

This register holds the priority level of the maskable interrupt currently acknowledged. When an interrupt request is acknowledged, the bit of this register corresponding to the priority level of that interrupt request is set to 1 and remains set while the interrupt is serviced.

When the RETI instruction is executed, the bit corresponding to the interrupt request having the highest priority is automatically reset to 0 by hardware. However, it is not reset to 0 when execution is returned from non-maskable interrupt servicing or exception processing.

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

	7	6	5	4	3	2	1	0	Address	Initial value
ISPR	ISPR7	ISPR6	ISPR5	ISPR4	ISPR3	ISPR2	ISPR1	ISPR0	FFFFF19AH	00H

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)

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.

	31																							8	7	6	5	4	3	2	1	0	Initial value
PSW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NP	EP	ID	SAT	CY	OV	S	Z	00000020H

Bit position	Bit name	Function
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 102*)
- Interrupt detection selectable for each interrupt input:
  - Rising edge
  - Falling edge
  - Both edges: rising and falling edge
  - High level
  - Low level
- Wakeup capability from stand-by mode of INTPn upon
  - Rising edge
  - Falling edge
  - Both edges: rising and falling edge

For configuration of the external interrupt events refer to *"Edge and Level Detection Configuration" on page 218*.

5.3.9 Software interrupts

This microcontroller provides maskable software interrupts to for processing of an interrupt service routine by the application software.

For initiating a software interrupt the interrupt request flag xxIC.xxIF of the concerned software interrupt "xx" must be set to 1. The following processing is identical to that of all other maskable interrupts.

5.4 Edge and Level Detection Configuration

The microcontroller provides the maskable external interrupts INTPn and one non-maskable interrupt (NMI).

INTPn can be configured to generate interrupts upon edges or levels, the NMI can be set up to react on edges.

(1) INTM0 to INTM3 - External interrupt configuration register

External interrupt function is configured by the registers INTM0…INTM3.

	7	6	5	4	3	2	1	0	Address	Initial value
INTM0	0	ELSEL1	ESEL11	ESEL10	NMIEN	ELSEL0	ESEL01	ESEL00	FFFF F700H	00H
R/W	R	R/W	R/W	R/W	R/(W)	R/W	R/W	R/W		
Reset	0	0	0	0	0	0	0	0		

	7	6	5	4	3	2	1	0	Address	Initial value
INTM1	0	ELSEL3	ESEL31	ESEL30	0	ELSEL2	ESEL21	ESEL20	FFFF F702H	00H
R/W	R	R/W	R/W	R/W	R	R/W	R/W	R/W		
Reset	0	0	0	0	0	0	0	0		

	7	6	5	4	3	2	1	0	Address	Initial value
INTM2	0	ELSEL5	ESEL51	ESEL50	0	ELSEL4	ESEL41	ESEL40	FFFF F704H	00H
R/W	R	R/W	R/W	R/W	R	R/W	R/W	R/W		
Reset	0	0	0	0	0	0	0	0		

	7	6	5	4	3	2	1	0	Address	Initial value
INTM3	0	ELSEL7	ESEL71	ESEL70	0	ELSEL6	ESEL61	ESEL60	FFFF F706H	00H
R/W	R	R/W	R/W	R/W	R	R/W	R/W	R/W		
Reset	0	0	0	0	0	0	0	0		

The register bits ELSELn, ESELn1 and ESELn0 configure the INTPn interrupt function:

ELSELn	ESELn1	ESELn0	Function
0	0	0	falling edge
0	0	1	rising edge
0	1	0	prohibited to use
0	1	1	falling and rising edge
1	0	0	low level detection
1	0	1	high level detection
1	1	0	low level detection
1	1	1	high level detection

The NMI and INTP0 share the same pin. The register bits NMIEN, ESEL0, ESEL01 and ESEL00 configure the NMI and INTP0 interrupt function:

NMIEN	**ESEL0**	**ESEL01**	**ESEL00**	**Function**	
				NMI	**INTP0**
0	0	0	0	masked	falling edge
	0	0	1		rising edge
	0	1	0		prohibited
	0	1	1		both edges
	1	0	0		low level
	1	0	1		high level
	1	1	0		low level
	1	1	1		high level
1	0	0	0	falling edge	falling edge
	0	0	1	rising edge	rising edge
	0	1	0	prohibited	prohibited
	0	1	1	both edges	both edges
	1	0	0	falling edge	low level
	1	0	1	rising edge	high level
	1	1	0	prohibited	low level
	1	1	1	both edges	high level

Caution The NMI configuration bits INTM0.NMIEN and INTM0.ESEL0[1:0] can only be changed if INTM0.NMIEN = 0.
Due to INTM0.NMIEN = 0 after reset the NMI function is disabled and must be enabled by the application software. Once enabled, the NMI function cannot be disabled by software.
Specify INTM0.ESEL0[1:0] before or at the same time with setting INTM0.NMIEN = 1.

Note that INTM0.ESEL0 can be written independently of INTM0.NMIEN.

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 (00000040H or 00000050H) corresponding to the software exception to the PC, and transfers control.

Figure 5-10 illustrates the processing of a software exception.

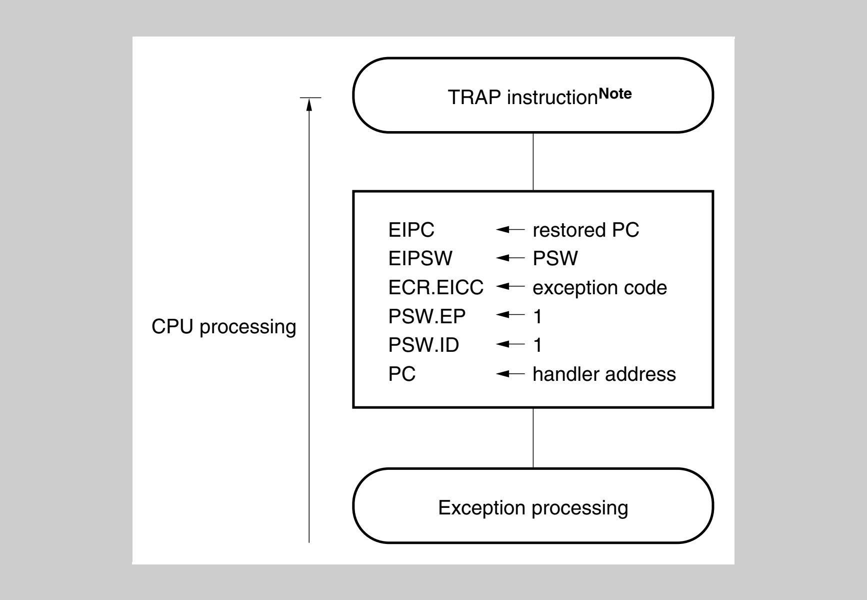


Figure 5-10 **Software exception processing**

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 0FH, it becomes 00000040H, and if the vector is 10H to 1FH, it becomes 00000050H.

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.

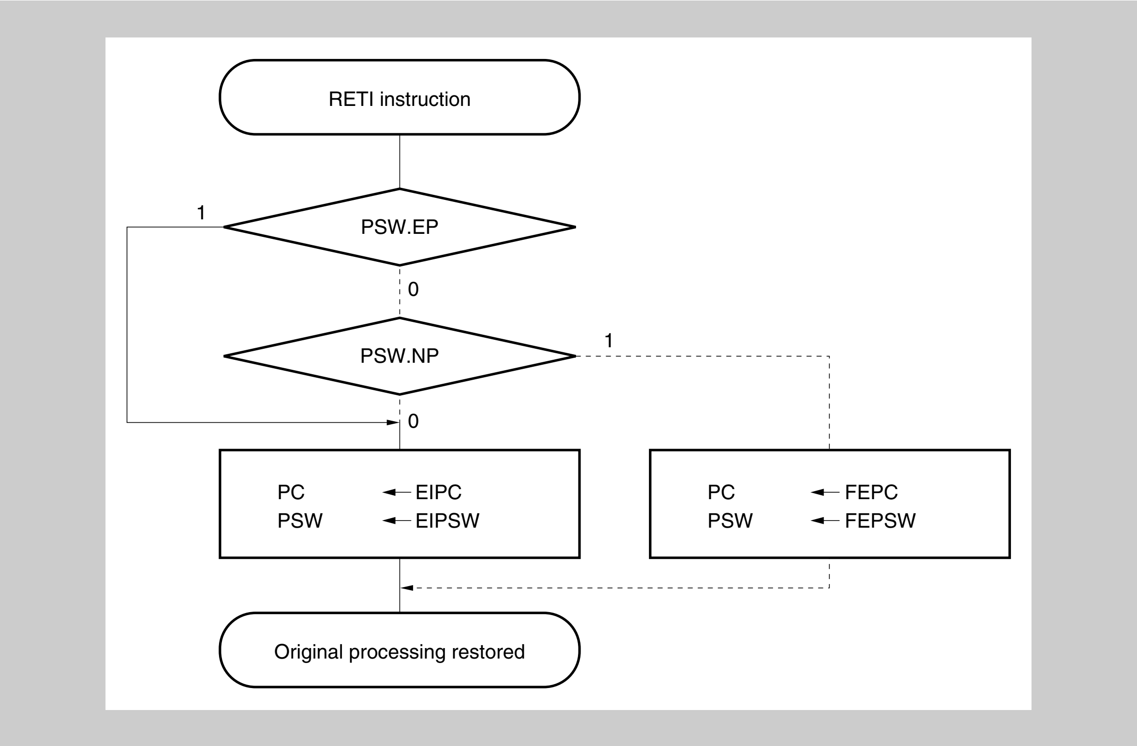


Figure 5-11 RETI instruction processing

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.

	31 8	7	6	5	4	3	2	1	0	Initial value
PSW	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	NP	EP	ID	SAT	CY	OV	S	Z	00000020H

Bit position	Bit name	Function
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 111111B, a sub-opcode (bits 23 to 26) of 0111B to 1111B, and a sub-opcode (bit 16) of 0B. 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 (00000060H) corresponding to the exception trap to the PC, and transfers control.

Figure 5-12 illustrates the processing of the exception trap.

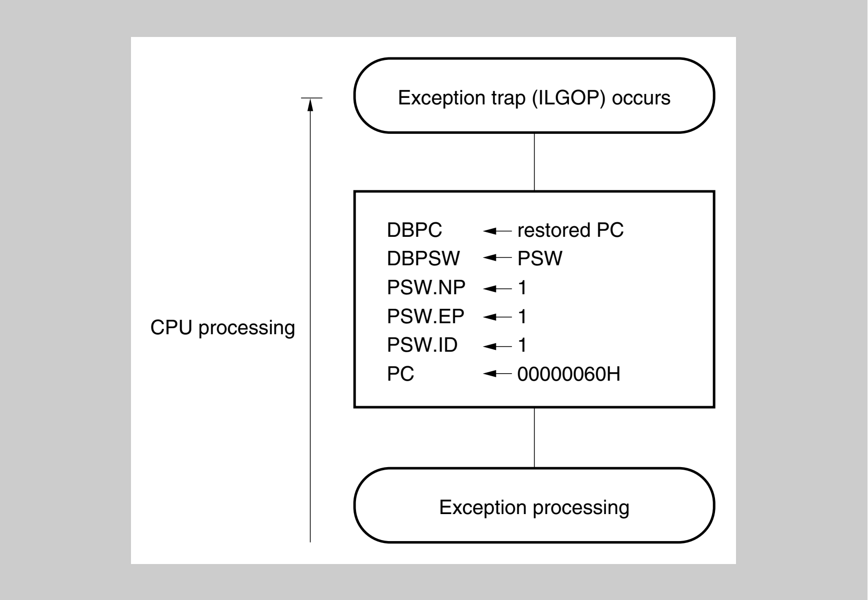


Figure 5-12 Exception trap processing

(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.

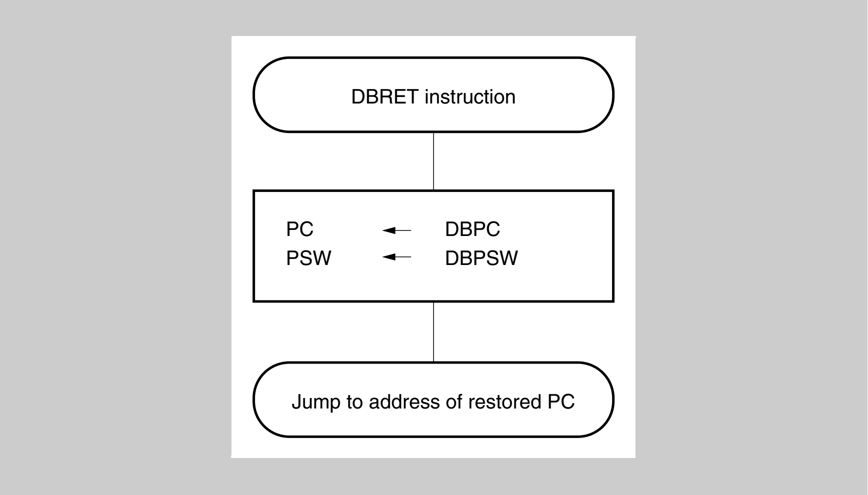


Figure 5-13 Restore processing from exception trap

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 (00000060H) corresponding to the debug trap to the PC and transfers control.

Figure 5-14 illustrates the processing of the debug trap.

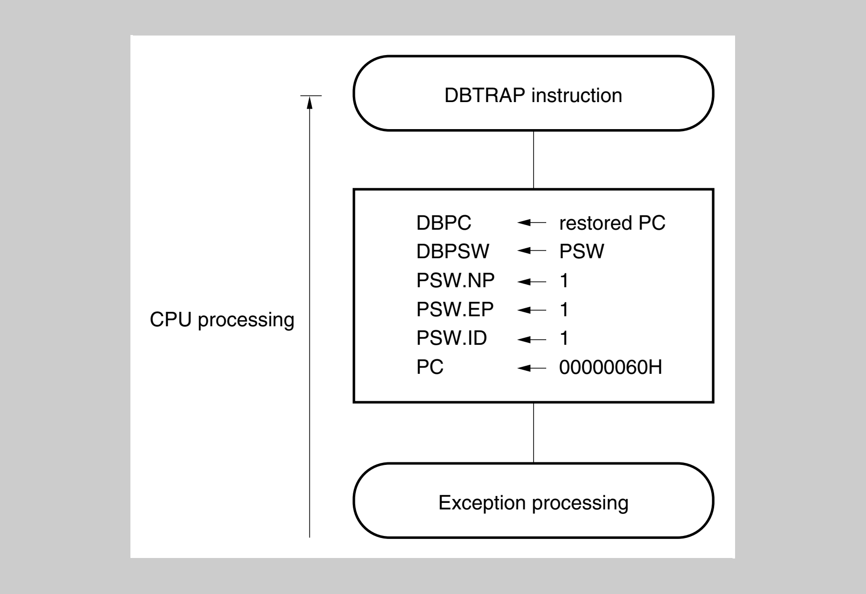


Figure 5-14 Debug trap processing

(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.

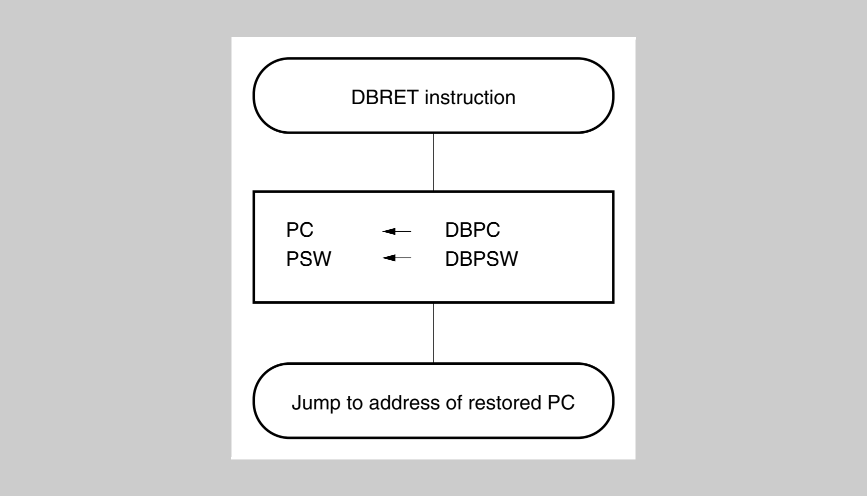


Figure 5-15 Restore processing from debug trap

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)	
...	
	...
	Higher priority maskable interrupt acknowledgment
	...
...	
• DI instruction (interrupt acknowledgment disabled)	
• Saved value restored to EIPSW	
• Saved value restored to EIPC	
• RETI instruction	

(2) Generation of exception in service program

Service program of maskable interrupt or exception

...	
...	
• EIPC saved to memory or register	
• EIPSW saved to memory or register	
...	
• TRAP instruction	
	...
	TRAP/exception acknowledgment
	...
...	
• Saved value restored to EIPSW	
• Saved value restored to EIPC	
• RETI instruction	

The priority order for multiple interrupt processing control has 8 levels, from 0 to 7 for each maskable interrupt request (0 is the highest priority), but it can be set as desired via software. Setting of the priority order level is done using the PPRn0 to PPRn2 bits of the interrupt control request register (PICn), which is provided for each maskable interrupt request. After system reset, an interrupt request is masked by the PMKn bit and the priority order is set to level 7 by the PPRn0 to PPRn2 bits.

The priority order of maskable interrupts is as follows.

(High) Level 0 > Level 1 > Level 2 > Level 3 > Level 4 >
Level 5 > Level 6 > Level 7 (Low)

Interrupt processing that has been suspended as a result of multiple processing control is resumed after the processing of the higher priority interrupt has been completed and the RETI instruction has been executed.

A pending interrupt request is acknowledged after the current interrupt processing has been completed and the RETI instruction has been executed.

Caution In a non-maskable interrupt processing routine (time until the RETI instruction is executed), maskable interrupts are suspended and not acknowledged.

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 228*).
- When the interrupt control register is accessed

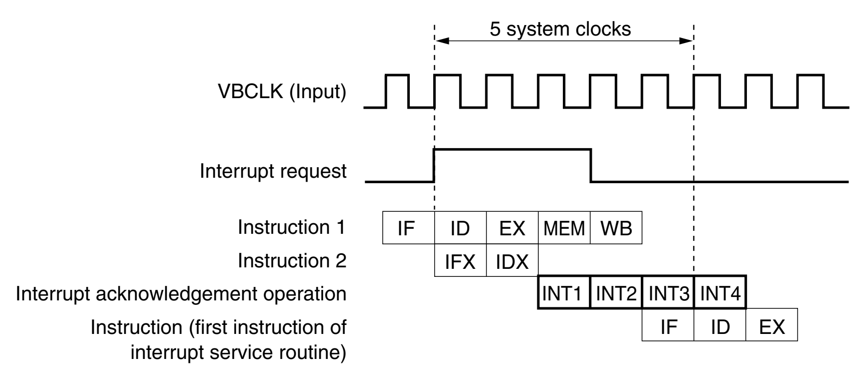


Figure 5-16 Pipeline operation at interrupt request acknowledgment (outline)

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

Note If the same interrupt occures during the interrupt acknowledge time of 5 cycles, this new interrupt will discarded. The next interrupt of the same source will only be registered after these 5 cycles.

Table 5-4 Interrupt response time

Interrupt response time (internal system clocks)			Condition
	Internal interrupt	External interrupt	
Minimum	5	5 + analog delay time	The following cases are exceptions: • In IDLE/software STOP mode • External bit access • Two or more interrupt request non-sample instructions are executed • Access to interrupt control register
Maximum	11	11 + analog delay time	

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).

Chapter 6 Flash Memory

The µPD70F3420, µPD70F3421, µPD70F3422, µPD70F3422, µPD70F3423, µPD70F3424, µPD70F3425, µPD70F3426 and µPD70F3427 microcontrollers are equipped with internal flash memory. The flash memory is attached to the V850 Fetch Bus VFB interface of the V850E CPU core. It is used for program code and storage of constant data.

When fetching an instruction, 4 bytes of the flash memory can be accessed in 1 clock.

The flash memory can be written mounted on the target board (on-board write), by connecting a dedicated flash programmer to the target system.

Flash memory is commonly used in the following development environments and applications:

- For altering software after solder-mounting of the microcontroller on the target system.
- For differentiating software in small-scale production of various models.
- For data adjustment when starting mass production.

6.1 Overview

Features summary

- Internal VFB flash memory:
  - µPD70F3427, µPD70F3426, µPD70F3425: 1 MB
  - µPD70F3424, µPD70F3423: 512 KB
  - µPD70F3422: 384 KB
  - µPD70F3421: 256 KB
  - µPD70F3420: 128 KB
- µPD70F3427, µPD70F3426, µPD70F3425, µPD70F3424 operation speed: up to 50.4 MHz by 2-way interleaved access
  - 4-byte/1 CPU clock cycle access for consecutive instruction fetches
  - 4-byte/5 CPU clock cycles access for random instruction and data fetches
- µPD70F3423 operation speed: up to 25.2 MHz by 2-way interleaved access
  - 4-byte/1 CPU clock cycle access for consecutive instruction fetches
  - 4-byte/3 CPU clock cycles access for random instruction and data fetches
- µPD70F3422, µPD70F3421, µPD70F3420 operation speed: up to 25.2 MHz with non-interleaved access
- All-blocks batch erase or single block erase
- Erase/write with single power supply
- Communication with dedicated flash programmer via various serial interfaces
- On-board and off-board programming
- Flash memory programming by self-programming

6.1.1 Flash memory address assignment

The µPD70F3427, µPD70F3426, µPD70F3425 1 MB flash memory is made up of 256 blocks. *Figure 6-1* shows the address assignment of the flash memory blocks.

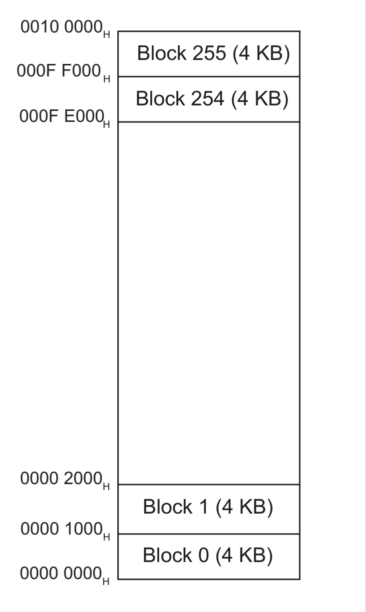


Figure 6-1 **Address assignment of µPD70F3427, µPD70F3426, µPD70F3425 flash memory blocks**

The µPD70F3424, µPD70F3423 512 KB flash memory is made up of 128 blocks. *Figure 6-2* shows the address assignment of the flash memory blocks.

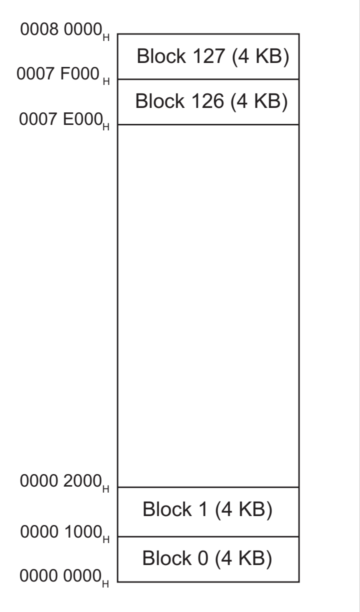


Figure 6-2 **Address assignment of µPD70F3424, µPD70F3423 flash memory blocks**

The µPD70F3422 384 KB flash memory is made up of 96 blocks. *Figure 6-3* shows the address assignment of the flash memory blocks.

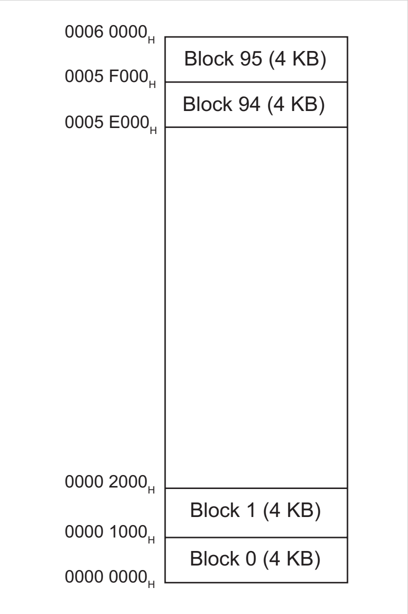


Figure 6-3 Address assignment of µPD70F3422 flash memory blocks

The µPD70F3421 256 KB flash memory is made up of 64 blocks. *Figure 6-4* shows the address assignment of the flash memory blocks.

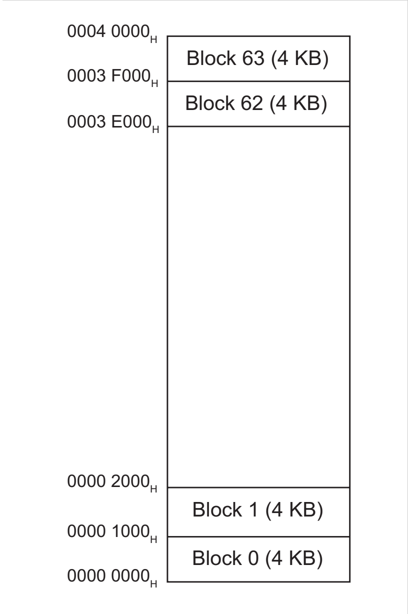


Figure 6-4 Address assignment of µPD70F3421 flash memory blocks

The µPD70F3420 128 KB flash memory is made up of 32 blocks. *Figure 6-3* shows the address assignment of the flash memory blocks.

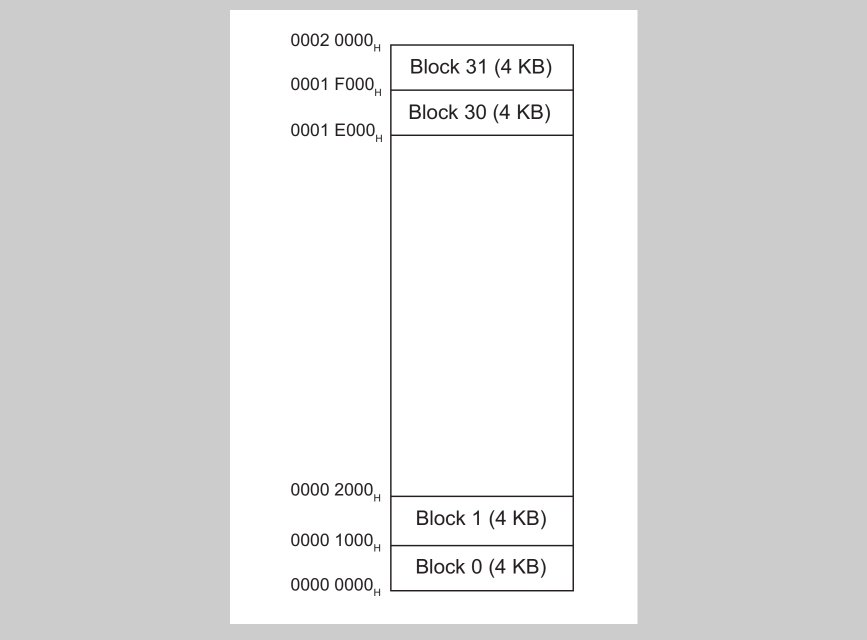


Figure 6-5 Address assignment of µPD70F3420 flash memory blocks

6.1.2 Flash memory erasure and rewrite

The following functions can be carried out by use of the flash memory self-programming library.

(1) Flash memory erasure

According to it’s block structure the flash memory can be erased in two different modes.

- All-blocks batch erasure
  Following areas can be erased all together:
  - µPD70F3427, µPD70F3426, µPD70F3425: 0000 0000_H to 000F $FFFF_H$
  - µPD70F3424, µPD70F3423: 0000 0000_H to 0007 $FFFF_H$
  - µPD70F3422: 0000 0000_H to 0005 $FFFF_H$
  - µPD70F3421: 0000 0000_H to 0003 $FFFF_H$
  - µPD70F3420: 0000 0000_H to 0001 $FFFF_H$
- Block erasure
  Each 4 KB flash memory block can be erased separately.

(2) Flash memory rewrite

Once a complete block has been erased it can be rewritten in units of 8 byte. Each unit can be rewritten only once after erasure of the complete block.

6.1.3 Flash memory programming

The internal flash memory can be programmed in three different ways:

- Programming via self-programming
- Programming via N-Wire interface
- Programming with external flash programmer

While the self-programming mode can be initiated from the normal operation mode the external flash programmer mode is entered immediately after release of a system reset. Refer to *“Operation Modes“ on page 114* for details on how to enter normal operation or external flash programming mode.

6.1.4 Boot block swapping

The microcontrollers with flash memory support secure boot block swapping.

For comprehensive information concerning secure boot block swapping refer to the application note “Self-Programming” (document nr. U16929EE), which explains also the functions of the self-programming library. The latest version of this document can be loaded via the URL

http://www.ee.nec.de/updates

6.2 Flash Self-Programming

The internal flash memory can be programmed via the secure self-programming facility. This feature enables the user's application to re-program the flash memory. The self-programming functions are part of the internal firmware, which resides in an extra internal ROM. The user's application can call the self-programming functions via the self-programming library, provided by NEC.

Caution During self-programming make sure to disable all ROM correction facilities, as enabled ROM corrections may conflict with the internal firmware.

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

Self-programming must be in particular enabled in order to avoid unintended re-programming of the flash. Two ways to enable self-programming are provided:

- by setting the external FLMD0 pin to high level
  This requires some external components or wiring, e.g. connecting an output port to FLMD0.
- by setting an internal register bit
  This way does not need any special external components or wiring.

The following registers are used to enable self-programming internally by software.

6.2.1 Flash self-programming registers

For safety reasons flash self-programming needs to be explicitly enabled by use of two registers:

Table 6-1 Flash self-programming enable register overview

Register name	Shortcut	Address
Self-programming enable control register	SELFEN	FFFF $FCA0_H$
Self-programming enable protection register	SELFENP	FFFF $FCA8_H$

(1) SELFEN - Self-programming enable control register

The 8-bit SELFEN register enables the self-programming functions by software. It is an internal substitute to enabling self-programming by rising the FLMD0 pin to high level.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *“SELFENP - Self-programming enable protection register“ on page 235* for details.

Address FFFF $FCA0_H$

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

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

Bit position	Bit name	Function
0	FLEN	Enable self-programming 0: Flash write/erase function is controlled by the FLMD0 pin 1: Flash write/erase function is enabled

(2) SELFENP - Self-programming enable protection register

The 8-bit SELFENP register protects the register SELFEN from inadvertent write access, so that the system does not stop in case of a program hang-up.

After data has been written to the SELFENP register, the first write access to register SELFEN is valid. All subsequent write accesses are ignored. Thus, the value of SELFEN can only be rewritten in a specified sequence, and illegal write access is inhibited.

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

Address FFFF $FCA8_H$

Initial Value The contents of this register is undefined.

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

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

Peripherals and pin functions All peripheral functions of the microcontroller continue operation during the self-programming process. Further the functions of all pins do not change.

6.2.2 Interrupt handling during flash self-programming

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

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

http://www.ee.nec.de/updates

Since neither the interrupt vector table nor the interrupt handler routines, which are normally located in the flash memory, are accessible during self-programming, interrupt acknowledges have to be re-routed to non-flash memory, i.e. to the internal RAM or - for µPD70F3427 only - to the external memory.

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 concerned interrupt acknowledge has to be re-routed to that handler.

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, respectively external memory, by use of the SelfLib_UsrIntToRam self-programming library function.

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

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 107*)
- 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 107*).

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.).

6.3 Flash Programming via N-Wire

The microcontroller's flash memory is programmable via the N-Wire debug interface.

Programming of the flash memory can be performed by the debug tool running on the host machine.

Caution Programming the flash memory during debug sessions by the debug tool adds to the performed number of write/erase cycles of the flash memory.

Thus devices used for debugging shall not be used for mass production purposes afterwards.

6.4 Flash Programming with Flash Programmer

A dedicated flash programmer can be used for on-board or off-board writing of the flash memory.

(1) On-board programming

The contents of the flash memory can be rewritten with the microcontroller mounted on the target system. Mount a connector that connects the flash programmer on the target system.

A CSI or a UART interface can optionally be used for the communication between the external flash programmer and the V850 microcontroller.

All signals, including clock and power supply, can be provided by the external flash programmer. However, an on-board clock to the X1 input may be used instead of the clock, provided by the flash programmer.

(2) Off-board programming

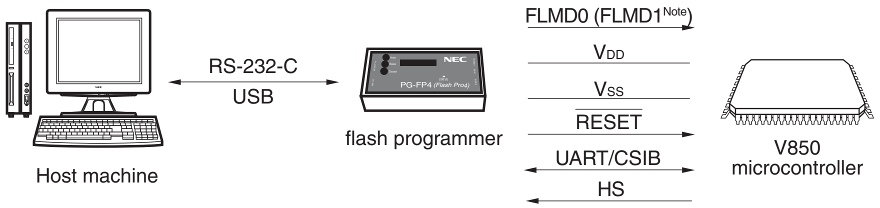
The flash memory of the microcontroller can be written before the device is mounted on the target system, by using a dedicated program adapter (FA series).

All signals, including clock and power supply, are provided by the external flash programmer.

Note The FA series is a product of Naito Densei Machida Mfg. Co., Ltd.

6.4.1 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 6-6 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:

- asynchroneous serial interface UART
- clocked serial interface CSIB

If the CSIB interface is used with handshake, the flash programmer’s HS signal is connected to a certain V850 port. The port used as the handshake port is given in *Table 6-2*.

Flash memory programming off-board requires a dedicated program adapter.

UARTA0, CSIB0 or N-Wire is used as the interface between the flash programmer and the microcontroller. Flash memory programming off-board requires a dedicated program adapter (FA series).

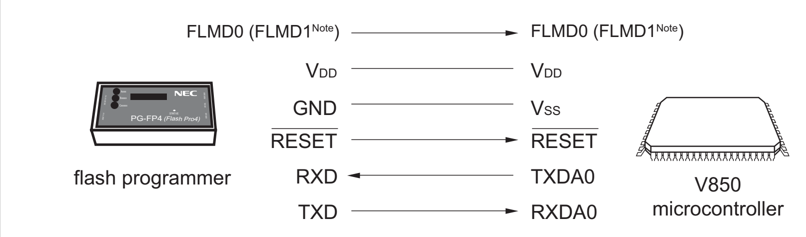
6.4.2 Communication mode

The communication between the flash programmer and the microcontroller utilizes the Asynchronous Serial Interface UARTA0 or optionally the synchronous serial interface CSIB0.

For programming via the synchronous serial interface CSIB0 without handshake and with handshake modes are supported. In the latter mode the port pin P84 is used for the programmer’s handshake signal HS.

(1) UARTA0

Transfer rate: 4.800 to 153.600 bps

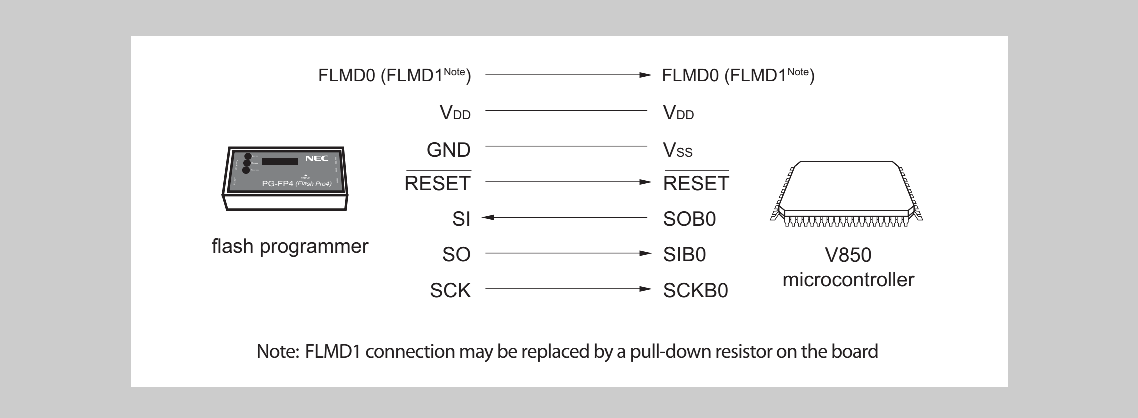


Note: FLMD1 connection may be replaced by a pull-down resistor on the board

Figure 6-7 Communication with flash programmer via UARTA0

(2) CSIB0 without handshake

Serial clock: up to 2.5 MHz (MSB first)

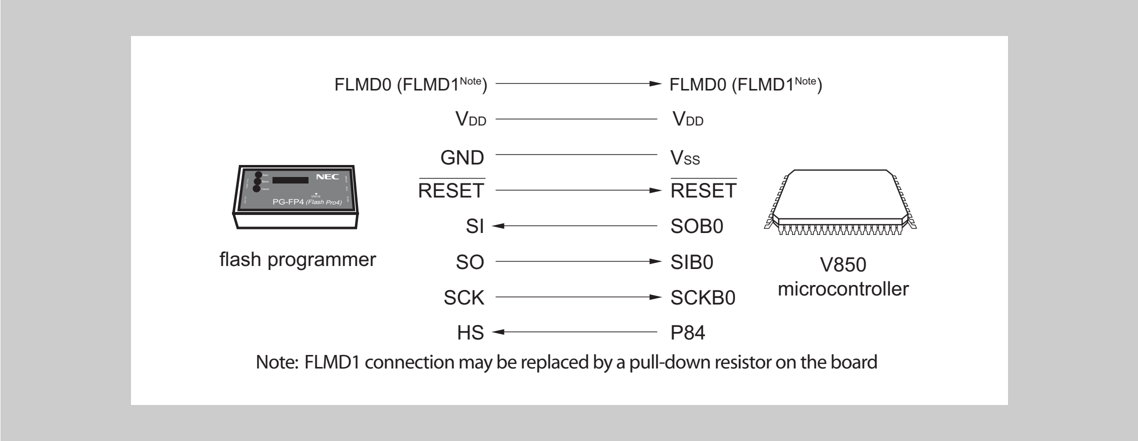


Note: FLMD1 connection may be replaced by a pull-down resistor on the board

Figure 6-8 Communication with flash programmer via CSIB0 without handshake

(3) CSIB0 with handshake (CSIB0 + HS)

Serial clock: Up to 2.5 MHz (MSB first)



Note: FLMD1 connection may be replaced by a pull-down resistor on the board

Figure 6-9 Communication with flash programmer via CSIB0 with handshake

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

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

Table 6-2 Signals generated by flash programmer PG-FP4

PG-FP4			Controller	Connection		
Signal name	**I/O**	**Pin function**	**Pin name**	**UARTA0**	**CSIB0**	**CSIB0 + HS**
FLMD0	Output	Write enable/disable, mode setting	FLMD0	○	○	○
FLMD1	Output	Mode setting	FLMD1	×	×	×
V_{DD}	I/O	V_{DD} voltage generation/voltage monitor	V_{DD}	○	○	○
GND	–	Ground	V_{SS}	○	○	○
CLK	Output	Clock output to the controller	X1	×	×	×
$\overline{\text{RESET}}$	Output	Reset signal	$\overline{\text{RESET}}$	○	○	○
SI/RxD	Input	Receive signal	SOB0/ TXDA0	○	○	○
SO/TxD	Output	Transmit signal	SIB0/RXDA0	○	○	○
SCK	Output	Transfer clock	SCKB0	×	○	○
HS	Input	Handshake signal for CSIB0 + HS communication	P84	×	×	○

Note ○: must be connected
×: does not need to be connected

6.4.3 Pin connection

A connector must be mounted on the target system to connect the flash programmer for on-board writing. In addition, a function to switch between the normal operation mode and flash memory programming mode must be provided on the board.

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

(1) Connection to flash programmer

In the normal operation mode, 0 V is input to the FLMD0 pin. The pull-down resistor at the FLMD0 pin ensures normal operation mode if no flash programmer is connected. In the flash memory programming mode, the V_{DD} write voltage is supplied to the FLMD0 pin. Additionally the FLMD1 pin, shared with port P07, has to hold 0 V level.

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

Table 6-3 Operation mode setting

Pins		Operation mode
FLMD0	FLMD1 (P07)	
0	X	Normal operation mode (fetch from flash)
1	0	Flash programming mode
	1	Setting prohibited

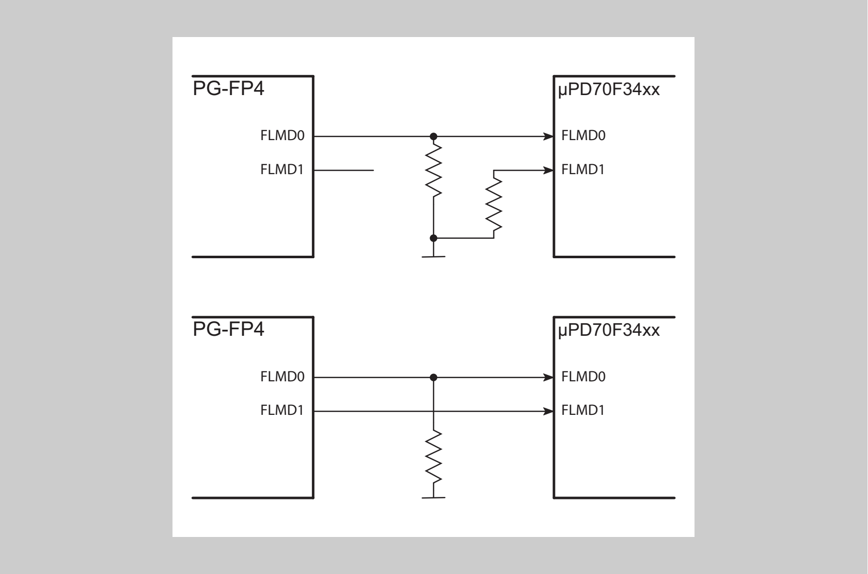


Figure 6-10 Example of connection to flash programmer PG-FP4

(2) Serial interface pins

The pins used by each serial interface are shown in the table below.

Table 6-4 Pins used by each serial interface

Serial interface	Pins
UARTA0	TXDA0, RXDA0 at pins P30/P31
CSIB0	SOB0, SIB0, SCKB0 at pins P40 - P42
CSIB0 + HS	SOB0, SIB0, SCKB0, P84

In flash programming mode the output drive strength control of the pins TXDA0, SOB0 and P84 is disabled. By this means the port pins provide maximum driver capability in order to maximize the transmission data rate to the flash programmer.

Caution

1. Since the output drive strength control of the pins TXDA0, SOB0 and P84 is disabled during programming these pins are not short-circuit proof any more. Short circuits at these pins may permanently damage the device.
2. If other devices are connected to the serial interface pins in use for flash memory programming in on-board programming mode take care that the concerned signals do not conflict with the signals of the flash programmer and the microcontroller. Output pins of the other devices must be isolated or set in high impedance state. Ensure that the other devices do not malfunction because of flash programmer signals.
3. Pay attention in particular if the flash programmer's $\overline{\text{RESET}}$ signal is connected also to an on-board reset generation circuit. The reset output of the reset generator may ruin the flash programming process and may need to be isolated.
4. All the port pins, including the pin connected to the flash programmer, go into an output high-impedance state in the flash memory programming mode. If there is a problem such as that an external device connected to a port prohibits the output high-impedance state, connect the port to V_{DD} or V_{SS} via a resistor.
5. Connect all oscillator pins in the same way as in the normal operation mode.
6. Supply the same power to all power supply pins, including reference voltages, power regulator pins, etc., as in the normal operation mode.

6.4.4 Programming method

(1) Flash memory control

The procedure to manipulate the flash memory is illustrated below.

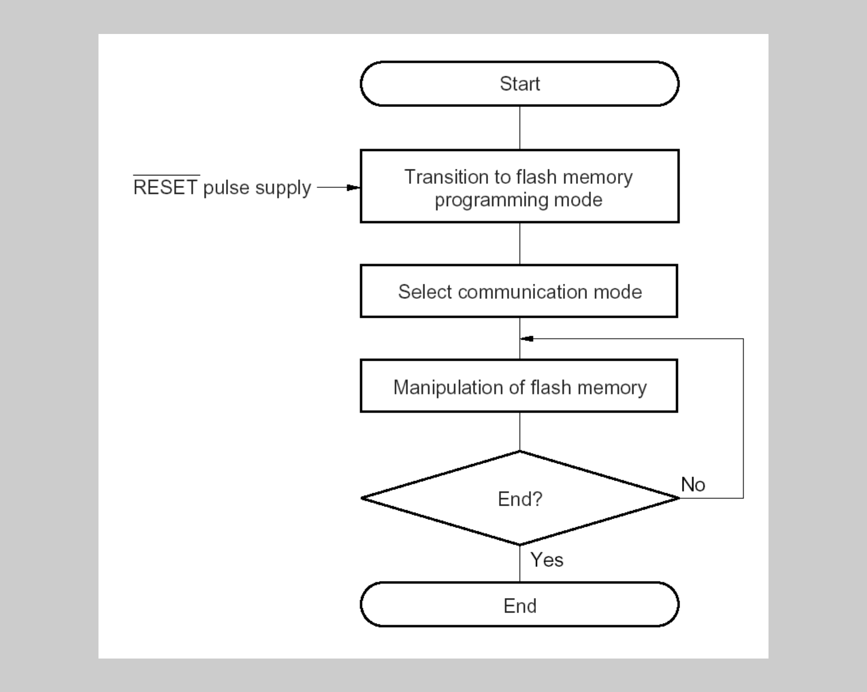


Figure 6-11 Flash memory manipulation procedure

(2) Flash memory programming mode

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

To set this mode, set the FLMD0 and FLMD1 pins as shown in *Table 6-3* and release $\overline{\text{RESET}}$.

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

Figure 6-12 gives an example how the UARTA0 is established for the communication between the flash programmer and the microcontroller.

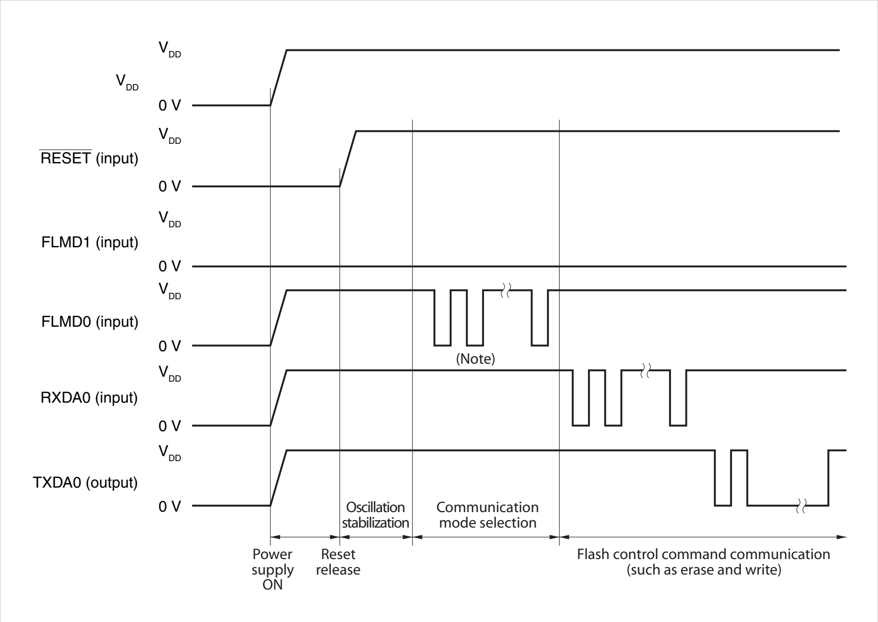


Figure 6-12 Flash memory programming mode start-up

Note The number of clocks to be inserted differs depending on the chosen communication mode. For details, refer to *Table 6-5*.

(3) Selecting communication mode

The communication mode is selected by applying a specified number of pulses to the MODE pin after the flash memory programming mode is set. These MODE pulses are generated by the flash programmer.

The relationship between the number of pulses and the communication mode is shown in the table below.

Table 6-5 Communication modes

MODE pulses	Communication mode	Remark
0	UARTA0	Communication rate: 9,600 bps (after reset), LSB first
8	CSIB0	microcontroller operates as slave, MSB first
11	CSIB0 + HS	microcontroller operates as slave, MSB first
Others	-	Setting prohibited

Note When UARTA0 is selected, the receive clock is calculated based on the reset command that is sent from the flash programmer after reception of the MODE pulses.

(4) Communication commands

The microcontroller communicates with the flash programmer via commands. The commands sent to the microcontroller are called commands, and the response signals sent by the microcontroller to the flash programmer are called response commands.

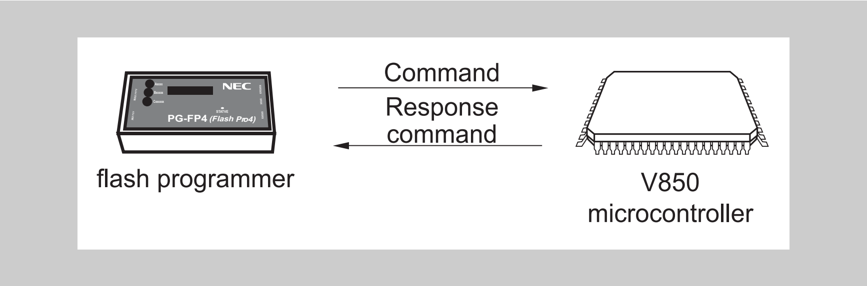


Figure 6-13 Communication commands

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

Table 6-6 Flash memory control commands

Classification	**Command name**	**Support**			**Function**
		CSIB	**CSIB + HS**	**UARTA**	
Blank check	Block blank check command	√	√	√	Checks erasure status of entire memory.
Erase	Chip erase command	√	√	√	Erase all memory contens including area that holds security flags, reset vector and other flash options
	Block erase command	√	√	√	Erases memory contents of specified block.
Write	Write command	√	√	√	Writes data by specifying write address and number of bytes to be written, and executes verify check.
Verify	Verify command	√	√	√	Compares input data with all memory contents.
Read	Read command	√	√	√	Read flash memory contents
System setting and control	Reset command	√	√	√	Escapes from each status.
	Oscillation frequency setting command	√	√	√	Sets oscillation frequency.
	Baud rate setting command	–	–	√	Sets baud rate when UART is used.
	Silicon signature command	√	√	√	Reads silicon signature information.
	Version acquisition command	√	√	√	Reads version information of device.
	Status command	√	√	–	Acquires operation status.
	Security setting command	√	√	√	Sets security of chip erasure, block erasure, and writing.

The microcontroller returns a response command to the command issued by the flash programmer. The response commands sent by the microcontroller are listed below.

Table 6-7 Response commands

Response command name	Function
ACK	Acknowledges command/data.
NAK	Acknowledges illegal command/data.

Chapter 7 Bus and Memory Control (BCU, MEMC)

Besides providing access to on-chip peripheral I/Os, the µPD70F3427 microcontroller device supports 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.

Since the BCU controls access to the on-chip peripherals, the registers BPC and VSWC have to be set up correctly for all devices.

Note Throughout this chapter, the individual chip select areas are identified by “k” (k = 0 to 7), for example $\overline{\text{CSk}}$ for the chip select signal k or BEC.BEk0 for setting the endian format of chip select area k.

7.1 Overview

The following external devices can be connected to the microcontroller device:

- SRAM / RAM
- ROM
- External I/O

Features summary The bus and memory control of the microcontroller device provides:

- 24 address signals (A0 to A23)
- Selectable data bus width for each chip select area (8 bits, 16 bits and 32 bits)
- 4 chip select signals externally available ($\overline{\text{CS0}}$, $\overline{\text{CS1}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$)
- Access to memory takes a minimum of two CPU clock cycles
- Up to 3 address setup wait states 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
- Up to 3 idle states can be inserted for each chip select area
- Up to 2 write strobe delay cycles can be inserted
- Direct Memory Access (DMA) support
- External bus mute function

- Page ROM controller
  - Direct connection to 8-bit/16-bit/32-bit page ROM supported
  - Page ROM controller handles page widths from 8 to 128 bytes
  - On-page judgement function
  - Masking addresses can be changed by register setting
  - Register for controlling the programmable wait during page access
  - Supported page access depends on data bus width:

Data bus width	Supported page access
32 bit	2/4/8/16/32 x 32 bits
16 bit	4/8/16/32/64 x 16 bits
8 bit	8/16/32/64/128 x 8 bits

7.2 Description

The figure below shows a block diagram of the modules that are necessary for accessing on-chip peripherals, external memory, or external I/O.

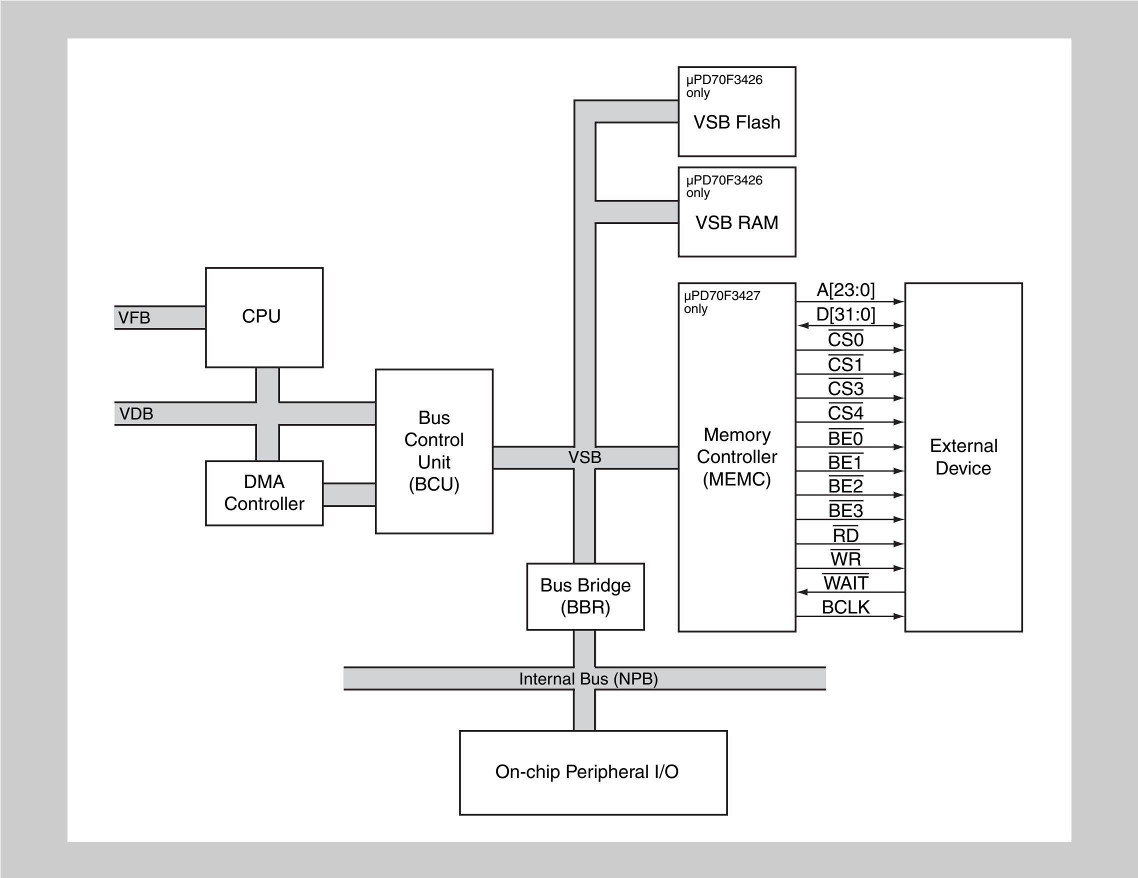


Figure 7-1 Bus and Memory Control diagram

Busses The busses are abbreviated as follows:

- NPB: NEC peripheral bus

- VSB: V850 system bus
- VDB: V850 data bus
- VFB: V850 fetch bus

BCU The Bus Control Unit (BCU) controls the access to on-chip peripherals, to external memory controller (MEMC), the VSB RAM and VSB Flash of the µPD70F3426 device.

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 banks. Each of the memory banks can be assigned to an external device via the chip area select control registers CSC0 and CSC1.

If an instruction uses such an address, a chip select signal is generated. The device supports four chip select signals ($\overline{\text{CS0}}$, $\overline{\text{CS1}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$). Each chip select signal covers a certain address range, also called “chip select area”. For details see *“Memory banks and chip select signals” on page 252*.

Additional byte enable signals $\overline{\text{BE0}}$ to $\overline{\text{BE3}}$ indicate valid data on any of the four bytes of the 32-bit data bus D[31:0].

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 strobe ($\overline{\text{WR}}$). From the 26 bit address of the CPU, the lower 24 bits are passed to the external device.

If two chip select signals are specified in the CSCn registers for a single memory bank, the priority control selects one of the chip select signals. The priority order is given in *“CSCn - Chip area select control registers” on page 265*.

The external signals of the Memory Controller and their state during and after reset are listed in the following table:

Table 7-1 **Memory Controller external connections and reset states**

Signal name	I/O	State during reset	State after reset	Function
A[23:0]	O	tbd	O	Address bus
D[31:16]	I/O	Hi-Z (3-state)	Port input	Data bus
D[15:0]		tbd	O	
$\overline{\text{CS0}}$	O	tbd	I	Chip select signal
$\overline{\text{CS1}}$				
$\overline{\text{CS3}}$				
$\overline{\text{CS4}}$				
$\overline{\text{BE0}}$	O	tbd	O	Byte enable signal
$\overline{\text{BE1}}$				
$\overline{\text{BE2}}$	O	Hi-Z (3-state)	Port input	
$\overline{\text{BE3}}$				
$\overline{\text{RD}}$	O	tbd	O	Read strobe
$\overline{\text{WR}}$	O	tbd	O	Write strobe
$\overline{\text{WAIT}}$	I	tbd	I	Data wait
BCLK	O	Hi-Z (3-state)	Port input	Bus clock

All pins are in input port mode after reset. Refer to *“Pin Functions” on page 33*.

ROMC To access external ROM with page access function (page ROM), the Page ROM Controller (ROMC) is provided. It can handle page widths from 8 to 128 bytes.

For more details, see *“Page ROM Controller” on page 279*.

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 endian format, wait functions and idle state insertions can be configured.

For a detailed description, see *“Configuration of Memory Access” on page 282*.

7.2.1 Memory banks and chip select signals

The 64 MB address range is divided into memory banks. Each memory bank is assigned to one or more chip select ($\overline{\text{CSn}}$) signals. If a memory bank is configured for external access, access to that memory bank generates the corresponding chip select signal (see *Figure 7-3 on page 254*). The combination of memory banks that activate the same chip select signal is called chip select area.

µPD70F3426 *Figure 7-2* shows the memory map of the µPD70F3426.

The 1 MB VSB Flash memory is mapped to the address range $0010\ 0000_H$ to $001F\ FFFF_H$ within bank 0. $\overline{\text{CS0}}$ is assigned to the VSB Flash memory.

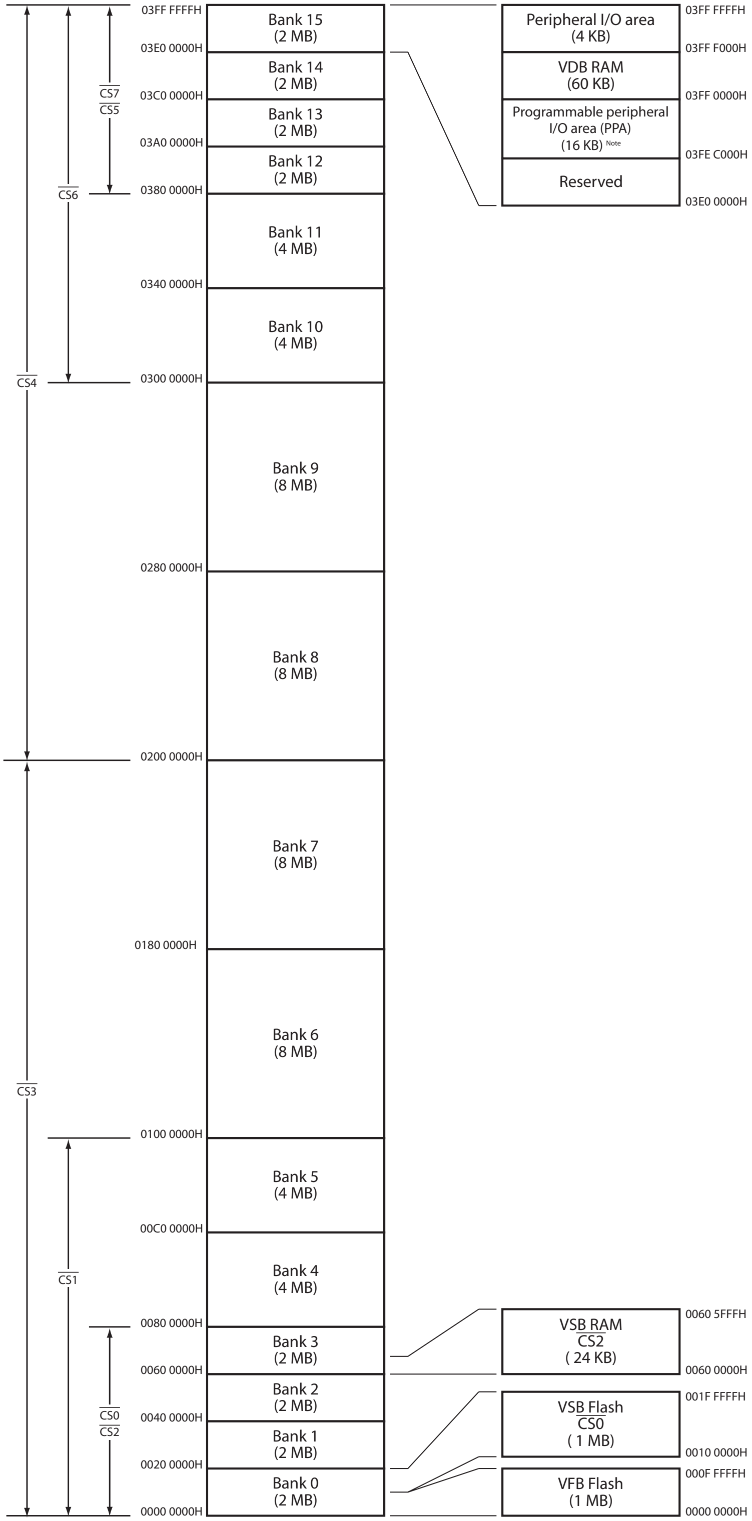
The 32 KB VSB RAM memory is mapped to the address range $0060\ 0000_H$ to $0060\ 5FFF_H$ within bank 3. $\overline{\text{CS2}}$ is assigned to the VSB RAM memory.

For access to the VSB Flash and VSB RAM the concerned BCU registers have to be set up as shown in *Figure 7-2*

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

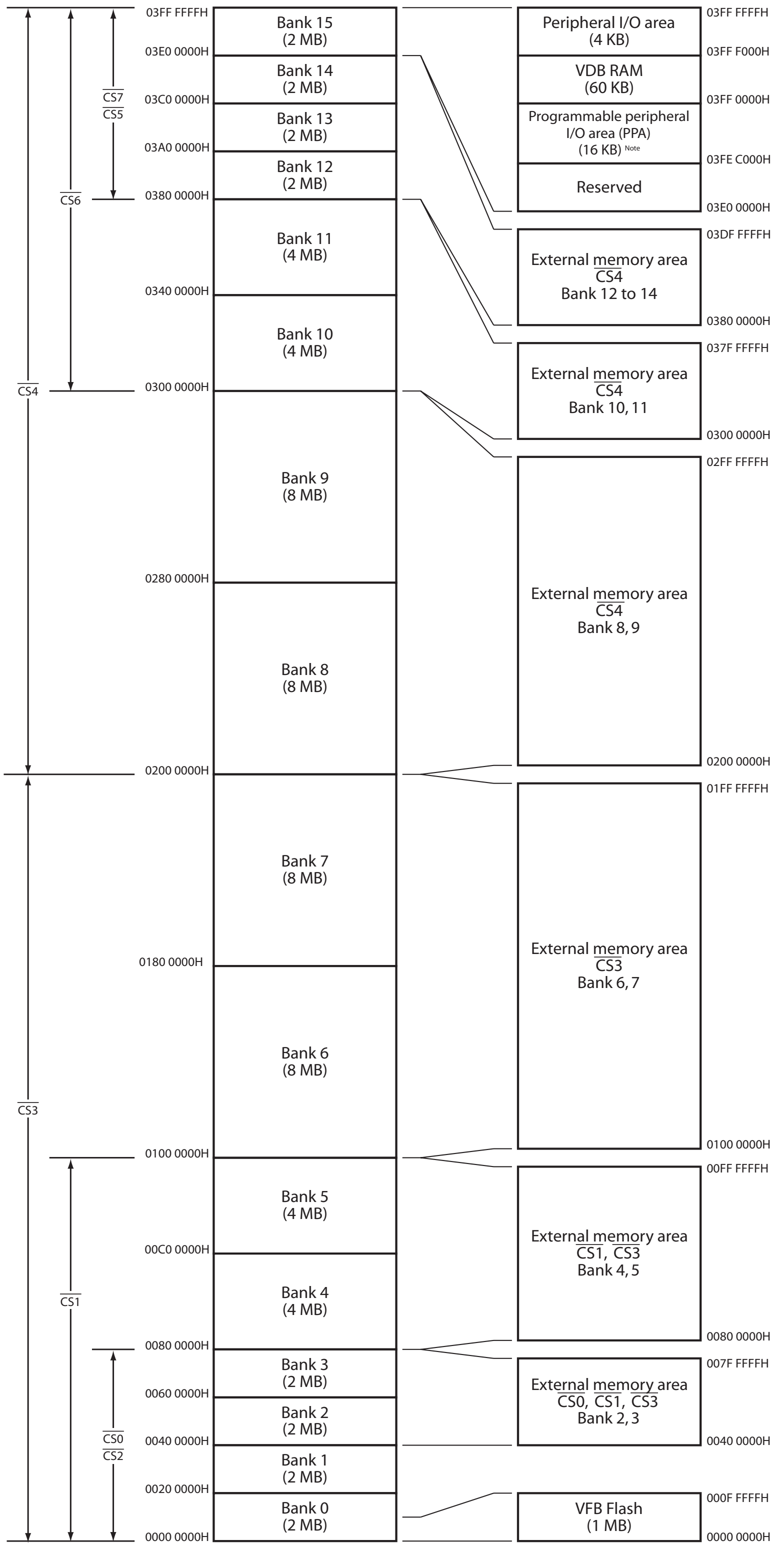
Table 7-2 BCU register settings for µPD70F3426 VSB Flash and VSB RAM access

Control bit	Required setting	Comment
CSC0.CS0[3:0]	0001_B	• bank 0 assigned to $\overline{\text{CS0}}$ for VSB Flash • default, don’t change
CSC0.CS2[3:0]	1000_B	• bank 3 assigned to $\overline{\text{CS2}}$ for VSB RAM • no default, must be changed
BEC.BE00	0	• little endian for VSB Flash • default, don’t change
BEC.BE20	0	• little endian for VSB RAM • default, don’t change



Note: The shown address range of the PPA assumes the BPC register to be set to 8FFBH.

Figure 7-2 Memory banks and chip select signals for µPD70F3426



Note: The shown address range of the PPA assumes the BPC register to be set to 8FFBH.

Figure 7-3 Memory banks and chip select signals for µPD70F3427

7.2.2 Chips select priority control

The chip select signals $\overline{\text{CS0}}$ to $\overline{\text{CS7}}$ can be assigned to overlapping memory areas by setting the chip select area control registers CSC0 and CSC1. The chip select priority control rules the generation of chip select signals in this case.

Access to internal resources, which are concurrently mapped to an external memory areas overrules the external access. As a consequence, the assigned $\overline{\text{CSn}}$ signal is not generated externally.

If different chip select signals are set (CSC0.CSCkm = 1) for the same memory bank, the priority order is as follows:

- internal resources > $\overline{\text{CS0}}$ > $\overline{\text{CS2}}$ > $\overline{\text{CS1}}$ > $\overline{\text{CS3}}$
- internal resources > $\overline{\text{CS7}}$ > $\overline{\text{CS5}}$ > $\overline{\text{CS6}}$ > $\overline{\text{CS4}}$

Examples:

- If both chip select signal $\overline{\text{CS0}}$ and $\overline{\text{CS1}}$ are set for memory bank 2, only the chip select signal $\overline{\text{CS0}}$ will be generated.
- If during access to bank 2 $\overline{\text{CS2}}$ should *not* be active, activate $\overline{\text{CS0}}$ for this bank (CSC0.CS02 = 1). Due to the priority order, only chip select signal $\overline{\text{CS0}}$ will be active for bank 2.

7.2.3 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 7-3 Peripheral I/O areas

Name	Address range	Size
Fixed peripheral I/O area	03FF F000$_H$ to 03FF FFFF$_H$	4 KB
Programmable peripheral I/O area (PPA)	Can be allocated at arbitrary addresses. Base address is defined in the BPC register.	16 KB

(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)

The usage and the address range of the PPA is configurable. The PPA extends the fixed peripheral I/O area and assigns an additional 12 KB address space for accessing on-chip peripherals.

The figure below illustrates the programmable peripheral I/O area (PPA).

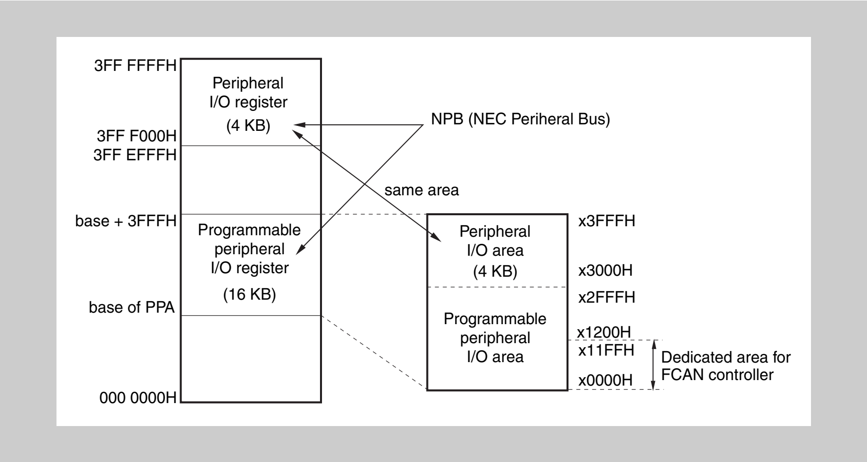


Figure 7-4 Programmable peripheral I/O area

The CAN modules registers and message buffers are allocated to the PPA. Refer to *“CAN module register and message buffer addresses” on page 666* for information how the calculate the register and message buffer addresses of the CAN modules.

Caution If the programmable peripheral I/O area overlaps one of the following areas, the programmable peripheral I/O area becomes ineffective:

- Peripheral I/O area
- ROM area
- RAM area

Note

1. The *fixed* peripheral I/O area is mirrored to the upper 4 KB of the *programmable* peripheral I/O area – regardless of the base address of the PPA. If data is written in one area, data having the same contents is also written in the other area.
2. All address definitions in this manual that refer to the programmable peripheral area assume that the base address of the PPA is 03FE C000$_H$, that means BPC = 8FFB$_H$.

7.2.4 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 911*), the system clock VBCLK and the setting of the VSWC register.

The CPU operation during an access to a register via the NPB depends also on the kind of peripheral I/O area:

- Fixed peripheral I/O area
  During a read or write access the CPU operation stops until the access via the NPB is completed.
- Programmable peripheral I/O area
  During a read access the CPU operation stops until the read access via the NPB is completed.
  During a write access the CPU operation continues operation, provided any preceded NPB access is already finished. If a preceded NPB access is still ongoing the CPU stops until this access is finished and the NPB is cleared.

Caution Pay attention at write accesses to NPB peripheral I/O registers via the programmable peripheral I/O area.

Since the CPU may continue operation, even though the data has not yet been transferred to its destination register, inconsistencies may occur between the program flow and the status of the registers.

In particular register set-ups which change an operational status of a certain module require special notice, like, for instance, masking/unmasking of interrupts via maskable interrupt control registers xxIC, enabling/disabling timers, etc.

7.2.5 Bus properties

This section summarizes the properties of the external bus.

(1) Bus width

The microcontroller device accesses external memory and external I/O in 8-bit, 16-bit, or 32-bit units.

The data bus size for each chip select area is specified in the local bus size configuration register (LBS).

The operation for each type of access is given in *“Access to 8-bit data busses” on page 296* and in *“Access to 16-bit data busses” on page 302*.

(2) Bus priority order

There are three kinds of external bus cycles as shown below. The DMA cycle has the highest priority, followed by the operand data access, and instruction fetch, in that order.

Table 7-4 Bus priority order

Priority	External bus cycle	Bus master
High	DMA cycle	DMA Controller
	Operand data access	CPU
Low	Instruction fetch	CPU

(3) Bus access

The number of CPU clocks necessary for accessing each resource – independent of the bus width – is as follows:

Table 7-5 Number of bus access clocks

Bus cycle configuration		Internal RAM	External I/O	External memory
Instruction fetch	Normal access	1[a]	–	2[b]
	Branch	1	–	2[b]
Operand data access		1	3[b]	2[b]

[a] In case of contention with data access, the instruction fetch from internal RAM takes 2 clocks.

[b] This is the minimum value.

7.2.6 Boundary operation conditions

The microcontroller device has the following boundary operation conditions:

(1) Program space

Instruction fetches from the internal peripheral I/O area are inhibited and yield NOP operations.

If a branch instruction exists at the upper limit of the internal RAM area, a prefetch operation (invalid fetch) that straddles over the internal peripheral I/O area does not occur.

(2) Data space

The microcontroller device is provided with an address misalign function.

By this function, data of any format (word: 32 bit, halfword: 16 bit, byte: 8 bit) can be placed to any address in memory, even though the address is not aligned to the data format (that means address 4n for words, address 2n for halfwords).

- Unaligned halfword data access
  When the LSB of the address is A0 =1, two byte accesses are performed.
- Unaligned word data access
  When the LSB of the address is A0 =1, two byte and one halfword accesses are performed. In total it takes 3 bus cycles.
  - When the LSBs of the address are A[1:0] =10_B, two halfword accesses are performed.

Note Accessing data on misaligned addresses takes more than one bus cycle to complete data read/write. Consequently, the bus efficiency will drop.

7.2.7 Initialization for access to external devices

To enable access to external devices, initialize the following registers after any reset.

1. Chip area select control registers CSCn
   Define the memory banks that are allocated to external devices. Memory banks that are not allocated to external devices, must be deactivated.
2. Bus cycle type configuration registers BCTn
   Specify the external devices that are connected to the microcontroller device. For memory banks that are not allocated to external devices, the corresponding bits in registers BCT0 and BCT1 should be reset.
3. LOcal bus size configuration register LBS
   Set the data bus width for the active chip select areas.
4. Data wait control registers DWCn
   Set the number of data wait states with respect to the starting bus cycle.
5. Bus cycle control register BCC
   Set the number of idle states for each chip select area.
6. Page ROM configuration register PRC
   If page ROM mode is selected (BCTn.BTk0 = 1), set whether a page ROM cycle is on-page or off-page.
7. Endian configuration register (BEC)
   Set the endian format for each chip select area.
8. Address setup wait control register (ASC)
   Set the number of address setup wait states for each chip select area.
9. Read delay control register (RDDLY)
   Activate the delay of the rising edge of $\overline{\text{RD}}$ strobe, as required.

Caution

1. Do not change these registers after initialization.

2. Do not access external devices before initialization is finished.

7.2.8 External bus mute function

If no access via the external memory interface is performed the external bus is set into a mute status. During mute the external bus interface pins take following states:

- A[22:0]: hold the address of the last external access
- D[31:0]: 3-state
- $\overline{\text{WR}}$, $\overline{\text{RD}}$, $\overline{\text{CS0}}$, $\overline{\text{CS1}}$, $\overline{\text{CS3}}$, $\overline{\text{CS4}}$: high level (inactive state)

7.3 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 of the Memory Controller (MEMC):

Table 7-6 Bus and memory control register overview

Module	Register name	Shortcut	Address
Bus Control Unit (BCU)	Peripheral area selection control register	BPC	FFFF F064$_H$
	Internal peripheral function wait control register	VSWC	FFFF F06E$_H$
	Chip area select control registers	CSC0	FFFF F060$_H$
		CSC1	FFFF F062$_H$
	Endian configuration register	BEC	FFFF F068$_H$
µPD70F3427 only:			
Memory Controller (MEMC)	Bus cycle configuration registers	BCT0	FFFF F480$_H$
		BCT1	FFFF F482$_H$
	Address setup wait control register	ASC	FFFF F48A$_H$
	Local bus size configuration register	LBS	FFFF F48E$_H$
	Data wait control registers	DWC0	FFFF F484$_H$
		DWC1	FFFF F486$_H$
	Bus cycle control register	BCC	FFFF F488$_H$
	Read delay control register	RDDLY	FFFF FF00$_H$
	Page ROM configuration register	PRC	FFFF F49A$_H$

7.3.1 BCU registers

The following registers are part of the BCU. They define the usage of the programmable peripheral I/O area (PPA), the data bus width, the endian format of word data, and they control access to external devices.

(1) BPC - Peripheral area selection control register

The 16-bit BPC register defines whether the programmable peripheral I/O area (PPA) is used or not and determines the starting address of the PPA.

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

Address FFFF F064$_H$

Initial Value 0000$_H$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA15	0	PA13	PA12	PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0

Table 7-7 BPC register contents

Bit Position	Bit Name	Function
15	PA15	Select usage of programmable peripheral I/O area (PPA). 0: PPA disabled 1: PPA enabled
13 to 0	PA[13:0]	Bits PA[13:0] specify bits 27 to 14 of the starting address of the PPA. The other bits of the address are fixed to 0.

Caution Bit 14 must always be 0.
The base address PBA of the programmable peripheral area sets the start address of the 16 KB PPA in a range of 256 MB. The 256 MB page is mirrored 16 times to the entire 32-bit address range.

The base address PBA is calculated by

$$PBA = BPC.PA[13:0] \times 2^{14}$$

Table 7-8 shows how the base address PBA of the programmable peripheral area is assembled.

Table 7-8 Address range of programmable peripheral area (16 KB)

31	…	28	27 … 14	13	…	1	0	bit
0	…	0	BPC.PA[13:0]	1	…	1	1	
	…		…			…		
0	…	0	BPC.PA[13:0]	0	…	0	1	
0	…	0	BPC.PA[13:0]	0	…	0	0	PBA

Note The recommended setting for the BPC register is $8FFB_H$. With this configuration the programmable peripheral area is mapped to the address range 03FE $C000_H$ to 03FE $FFFF_H$. With this setting the CAN message buffer registers are accessible via the addresses given in *“CAN Controller (CAN)” on page 639*.
The fixed peripheral area is mirrored to the address range 03FE $E000_H$ to 03FE $FFFF_H$.

(2) VSWC - Internal peripheral function wait control register

The 8-bit VSWC register defines the wait states inserted when accessing peripheral special function registers via the internal bus. Both address setup and data wait states are based on the system clock.

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

Address FFFF F06E$_H$

Initial Value 77$_H$

7	6	5	4	3	2	1	0
0	SUWL2	SUWL1	SUWL0	0	VSWL2	VSWL1	VSWL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 7-9 VSWC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>6 to 4</td><td>SUWL[2:0]</td><td>Address setup wait for internal bus:
<table>
<tr><th>SUWL2</th><th>SUWL1</th><th>SUWL0</th><th>Number of address setup wait states</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1 CPU system clock (VBCLK)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2 CPU system clock (VBCLK)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7 CPU system clock (VBCLK)</td></tr>
</table></td></tr>
<tr><td>2 to 0</td><td>VSWL[2:0]</td><td>Data wait for internal bus:
<table>
<tr><th>VSWL2</th><th>VSWL1</th><th>VSWL0</th><th>Number of data wait states</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1 CPU system clock (VBCLK)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2 CPU system clock (VBCLK)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7 CPU system clock (VBCLK)</td></tr>
</table></td></tr>
</table>

The following setups are recommended for VSWC:

Table 7-10 Recommended timing for internal bus

System clock[a] f_{VBCLK}	≤ 16 MHz	≤ 25 MHz	≤ 33 MHz	≤ 50 MHz	≤ 66 MHz	≤ 75 MHz
SUWL	0	0	1	1	1	1
VSWL	0	1	1	2	3	4
VSWC	00_H	01_H	11_H	12_H	13_H	14_H

[a] When deriving the system clock from the modulated clock of the SSCG, the maximum clock determines the correct register setting.

(3) CSCn - Chip area select control registers

The 16-bit registers CSC0 and CSC1 assign the chip select signals $\overline{CS0}$ to $\overline{CS3}$ and $\overline{CS4}$ to $\overline{CS7}$ to memory blanks (see also *“Memory banks and chip select signals” on page 252*). If a bit in CSCn is set, access to the corresponding memory bank will generate the corresponding chip select signal and activate the Memory Controller.

If several chip select signals are assigned to identical memory areas, a priority control rules the generation of the signals (refer to *“Chips select priority control” on page 255*).

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

Address CSC0: FFFF $F060_H$
CSC1: FFFF $F062_H$

Initial Value $2C11_H$

Thess registers must be initialized as described in *Table 7-13* and *Table 7-14*.

CSC0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CS33	CS32	CS31	CS30	CS23	CS22	CS21	CS20	CS13	CS12	CS11	CS10	CS03	CS02	CS01	CS00
$\overline{CS3}$				$\overline{CS2}$				$\overline{CS1}$				$\overline{CS0}$			

CSC1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CS43	CS42	CS41	CS40	CS53	CS52	CS51	CS50	CS63	CS62	CS61	CS60	CS73	CS72	CS71	CS70
$\overline{CS4}$				$\overline{CS5}$				$\overline{CS6}$				$\overline{CS0}$			

The register contents in *Table 7-11* and *Table 7-12* read as follows:

- CSkm = 0: Corresponding chip select signal is *not* active during access to memory bank.
- CSkm = 1: Corresponding chip select signal is active during access to memory bank.

Caution To initialize an external memory area after a reset, registers CSCn have to be set. Do not change these registers after initialization. Do not access external devices before initialization is finished.

Table 7-11 CSC0 register contents

Bit Position	Bit Name	Access to memory bank	Chip select signal
15	CS33	7	$\overline{\text{CS3}}$
14	CS32	6	
13	CS31	4 or 5	
12	CS30	0, 1, 2 or 3	
11	CS23	3	$\overline{\text{CS2}}$
10	CS22	2	
9	CS21	1	
8	CS20	0	
7	CS13	5	$\overline{\text{CS1}}$
6	CS12	4	
5	CS11	2 or 3	
0	CS10	0 or 1	
3	CS03	3	$\overline{\text{CS0}}$
2	CS02	2	
1	CS01	1	
0	CS00	0	

Table 7-12 CSC1 register contents

Bit Position	Bit Name	Access to memory bank	Chip select signal
15	CS43	8	$\overline{\text{CS4}}$
14	CS42	9	
13	CS41	10 or 11	
12	CS40	12, 13, 14 or 15	
11	CS53	12	$\overline{\text{CS5}}$
10	CS52	13	
9	CS51	14	
8	CS50	15	
7	CS63	10	$\overline{\text{CS6}}$
6	CS62	11	
5	CS61	12 or 13	
0	CS60	14 or 15	
3	CS73	12	$\overline{\text{CS7}}$
2	CS72	13	
1	CS71	14	
0	CS70	15	

Initialization Initialize the CSCn registers as shown in

- *Table 7-13* for µPD70F3426
- *Table 7-14* for µPD70F3427

Table 7-13 Initialization of the µPD70F3426 CSCn registers

Bits	Set to value	Comment
CSC0.CS0[3:0]	0001_B	$\overline{\text{CS0}}$ assigned to bank 0 to VSB Flash memory 010 0000_H - 01F $FFFF_H$. CSC0.CS0[3:0] must be left with their default value 1000_B.
CSC0.CS1[3:0]	1000_B	CSC0.CS1[3:0] must be left with their default value 1000_B.
CSC0.CS2[3:0]	1000_B	$\overline{\text{CS2}}$ assigned to bank 3 to VSB RAM memory 060 0000_H - 060 $5FFF_H$. **Caution:** CSC0.CS2[3:0] must be changed to 1000_B (default value is 1100_B).
CSC0.CS3[3:0]	0010_B	CSC0.CS3[3:0] must be left with their default value 0010_B.
CSC1.CS4[3:0]	0010_B	CSC0.CS4[3:0] must be left with their default value 0010_B.
CSC1.CS5[3:0]	1100_B	CSC0.CS5[3:0] must be left with their default value 1100_B.
CSC1.CS6[3:0]	0001_B	CSC0.CS6[3:0] must be left with their default value 0001_B.
CSC1.CS7[3:0]	0001_B	CSC0.CS7[3:0] must be left with their default value 0001_B.

Caution For the µPD70F3426 the CSC0 register must be changed to 2811_H and CSC1 must be left with its default value $2C11_H$. Do not modify these registers after setting CSC0 = 2811_H.

Table 7-14 Initialization of the µPD70F3427 CSCn registers

Bits	Set to value	Comment
CSC0.CS0[3:0]	xx00$_B$	Set CSC0.CS0[3:2] as required to assign $\overline{\text{CS0}}$ to bank 2 to 3 to external memory 040 0000$_H$ - 07F FFFF$_H$. **Caution:** CSC0.CS0[1:0] must be changed to 00$_B$ (default value is 01$_B$).
CSC0.CS1[3:0]	xxx0$_B$	Set CSC0.CS1[3:1] as required to assign $\overline{\text{CS1}}$ to bank 2 to 5 to external memory 040 0000$_H$ - 0FF FFFF$_H$. **Caution:** CSC0.CS10 must be changed to 0 (default value is 1).
CSC0.CS2[3:0]	1100$_B$	CSC0.CS2[3:0] must be left with their default value 1100$_B$.
CSC0.CS3[3:0]	xxx0$_B$	Set CSC0.CS3[3:1] as required to assign $\overline{\text{CS3}}$ to bank 4 to 7 to external memory 080 0000$_H$ - 1FF FFFF$_H$. CSC0.CS30 must be left with its default value 0.
CSC1.CS4[3:0]	xxx0$_B$	Set CSC1.CS4[3:1] as required to assign $\overline{\text{CS4}}$ to bank 8 to 11 to external memory 200 0000$_H$ - 37F FFFF$_H$. CSC0.CS40 must be left with its default value 0.
CSC0.CS5[3:0]	1100$_B$	CSC0.CS5[3:0] must be left with their default value 1100$_B$.
CSC1.CS6[3:0]	0001$_B$	CSC0.CS6[3:0] must be left with their default value 0001$_B$.
CSC1.CS7[3:0]	0001$_B$	CSC0.CS7[3:0] must be left with their default value 0001$_B$.

(4) BEC - Endian configuration register

The 16-bit BEC register defines the endian format in which word data in the memory is processed. Each chip select area is controlled separately.

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

Address FFFF $F068_H$

Initial Value 0000_H

This register must be initialized as described in *Table 7-16* and *Table 7-17*.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	BE70	0	BE60	0	BE50	0	BE40	0	BE30	0	BE20	0	BE10	0	BE00
$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$	

Table 7-15 **BEC register contents**

Bit Position	Bit Name	Function
14, 12, 10, 8, 6, 4, 0	BEk0	Sets the endian format for processing of word data for each chip select area. 0: Little endian 1: Big endian

Caution

1. The bits marked with 0 must always be 0.
2. To initialize an external memory area after a reset, register BEC has to be set. Do not change this register after initialization. Do not access external devices before initialization is finished.

Note

1. Accesses to all internal resources are fixed to little endian format.
2. Different chip select signals can be active for the same memory bank. The priority order defines, which chip select signal is valid.
3. Set the chip select area which is specified as the programmable peripheral I/O area to little endian format.

Initialization Initialize the BEC register as shown in

- *Table 7-16* for µPD70F3426
- *Table 7-17* for µPD70F3427

Table 7-16 Initialization of the µPD70F3426 BEC register

Bits	Set to value	Comment
BEC.BE00	0	Endian format for VSB Flash memory: little endian BEC.BE00 must be left with their default value 0
BEC.BE10	0_B	BEC.BE10 must be left with their default value 0
BEC.BE20	0	Endian format for VSB RAM: little endian BEC.BE20 must be left with their default value 0
BEC.BE30	0_B	BEC.BE30 must be left with their default value 0
BEC.BE40	0_B	BEC.BE40 must be left with their default value 0
BEC.BE50	0_B	BEC.BE50 must be left with their default value 0
BEC.BE60	0_B	BEC.BE60 must be left with their default value 0
BEC.BE70	0_B	BEC.BE70 must be left with their default value 0

Caution For the µPD70F3426 the BEC register must be left with its default value 0000_H. Do not modify this register.

Table 7-17 Initialization of the µPD70F3427 BEC register

Bits	Set to value	Comment
BEC.BE00	x_B	Set as required
BEC.BE10	x_B	Set as required
BEC.BE20	0_B	BEC.BE20 must be left with their default value 0
BEC.BE30	x_B	Set as required
BEC.BE40	x_B	Set as required
BEC.BE50	0_B	BEC.BE50 must be left with their default value 0
BEC.BE60	0_B	BEC.BE60 must be left with their default value 0
BEC.BE70	0_B	BEC.BE70 must be left with their default value 0

7.3.2 Memory controller registers (µPD70F3427 only)

The following registers are part of the Memory Controller. They specify the type of external device that is connected, the number of data wait states, the number of address wait states, the number of idle states, and they control features for page ROM.

(1) BCTn - Bus cycle configuration registers

The 16-bit BCT0 register specifies the external devices that are connected to the microcontroller device. The register enables the operation of the Memory Controller for each chip select signal.

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

Address BCT0: FFFF $F480_H$
BCT1: FFFF $F482_H$

Initial Value 4444_H

BCT0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ME3	1	0	BT30	ME2	1	0	BT20	ME1	1	0	BT10	ME0	1	0	BT00
$\overline{CS3}$				$\overline{CS2}$				$\overline{CS1}$				$\overline{CS0}$			

BCT1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ME7	1	0	BT70	ME6	1	0	BT60	ME5	1	0	BT50	ME4	1	0	BT40
$\overline{CS7}$				$\overline{CS6}$				$\overline{CS5}$				$\overline{CS4}$			

Table 7-18 BCTn register contents

Bit Position	Bit Name	Function
15, 11, 7, 3	MEk	Enables/disables Memory Controller operation for chip select area k. 0: Operation disabled 1: Operation enabled
12, 8, 4, 0	BTk0	Specifies the devices that are connected to chip select area k. 0: SRAM or external I/O connected 1: Page ROM connected

Caution

1. The bits marked with 0 must always be 0.
2. The bits marked with 1 must always be 1.
3. To initialize an external memory area after a reset, registers BCTn have to be set. Do not change this register after initialization. Do not access external devices before initialization is finished.

(2) LBS - Local bus size configuration register

The 16-bit LBS register controls the data bus width for each chip select area.

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

Address FFFF F48E$_H$

Initial Value AAAA$_H$.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LB71	LB70	LB61	LB60	LB51	LB50	LB41	LB40	LB31	LB30	LB21	LB20	LB11	LB10	LB01	LB00
$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$	

Table 7-19 LBS register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>15 to 0</td><td>LBk[1:0]</td><td>Sets the data bus width for each chip select area.
<table>
<tr><th>LBk1</th><th>LBk0</th><th>Data bus size</th></tr>
<tr><td>0</td><td>0</td><td>8 bit</td></tr>
<tr><td>0</td><td>1</td><td>16 bit</td></tr>
<tr><td>1</td><td>0</td><td>32 bit</td></tr>
<tr><td>1</td><td>1</td><td>prohibited</td></tr>
</table>
</td></tr>
</table>

(3) ASC - Address setup wait control register

The 16-bit ASC register controls the number of wait states between address setup and the first access cycle (T1). Each chip select area is controlled separately. A maximum of three address setup wait states is possible.

Address setup wait states can be inserted when accessing
- SRAM
- page ROM

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

Address FFFF F48A$_H$

Initial Value FFFF$_H$: After system setup, by default, three address setup wait states are inserted for each chip select area.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AC71	AC70	AC61	AC60	AC51	AC50	AC41	AC40	AC31	AC30	AC21	AC20	AC11	AC10	AC01	AC00
$\overline{\text{CS7}}$		$\overline{\text{CS6}}$		$\overline{\text{CS5}}$		$\overline{\text{CS4}}$		$\overline{\text{CS3}}$		$\overline{\text{CS2}}$		$\overline{\text{CS1}}$		$\overline{\text{CS0}}$	

Table 7-20 ASC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>15 to 0</td><td>ACk[1:0]</td><td>Sets the number of address setup wait states for each chip select area.
<table>
<tr><th>ACk[1:0]</th><th>Wait states inserted after address setup</th></tr>
<tr><td>00$_B$</td><td>No wait state inserted</td></tr>
<tr><td>01$_B$</td><td>1 wait state</td></tr>
<tr><td>10$_B$</td><td>2 wait states</td></tr>
<tr><td>11$_B$</td><td>3 wait states</td></tr>
</table>
</td></tr>
</table>

Note
1. During address setup wait, the external wait function ($\overline{\text{WAIT}}$ pin) is disabled.
2. For access to internal memory, the setting of register ASC is neglected. No wait states are inserted after address setup.

Caution 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.

(4) DWCn - Data wait control registers

The 16-bit DWCn registers control the number of wait states after the first access cycle (T1). 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 DWC0: FFFF F484$_H$
DWC1: FFFF FFE0$_H$

Initial Value 7777_H: After system setup, by default, seven data wait states are inserted for each chip select area.

DWC0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	DW32	DW31	DW30	0	DW22	DW21	DW20	0	DW12	DW11	DW10	0	DW02	DW01	DW00
$\overline{\text{CS3}}$				$\overline{\text{CS2}}$				$\overline{\text{CS1}}$				$\overline{\text{CS0}}$			

DWC1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	DW72	DW71	DW70	0	DW62	DW61	DW60	0	DW52	DW51	DW50	0	DW42	DW41	DW40
$\overline{\text{CS7}}$				$\overline{\text{CS6}}$				$\overline{\text{CS5}}$				$\overline{\text{CS4}}$			

Table 7-21 DWCn registers contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>15 to 0</td><td>DWk[2:0]</td><td>Sets the number of wait states after the first access cycle (T1) for each chip select area.
<table>
<tr><th>DWk[2:0]</th><th>Number of inserted wait states</th></tr>
<tr><td>000_B</td><td>No wait state inserted</td></tr>
<tr><td>001_B</td><td>1 wait state</td></tr>
<tr><td>010_B</td><td>2 wait states</td></tr>
<tr><td>011_B</td><td>3 wait states</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111_B</td><td>7 wait states</td></tr>
</table>
</td></tr>
</table>

Note

1. For access to internal memory, programmable waits are *not* carried out.
2. During page ROM on-page access, wait control is performed according to PRC register setting.

Caution 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.

---

(5) BCC - Bus cycle control register

The 16-bit BCC register controls the number of idle states inserted after the T2 cycle. Each chip select area is controlled separately. A maximum of three idle states is possible.

Idle states can be inserted when accessing SRAM , external I/O, external ROM, or page ROM.

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

Address FFFF F488$_H$

Initial Value FFFF$_H$. After system reset, three idle states are inserted.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BC71	BC70	BC61	BC60	BC51	BC50	BC41	BC40	BC31	BC30	BC21	BC20	BC11	BC10	BC01	BC00
$\overline{\text{CS7}}$		$\overline{\text{CS6}}$		$\overline{\text{CS5}}$		$\overline{\text{CS4}}$		$\overline{\text{CS3}}$		$\overline{\text{CS2}}$		$\overline{\text{CS1}}$		$\overline{\text{CS0}}$	

Table 7-22 BCC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>15 to 0</td><td>BCk[1:0]</td><td>Sets the number of idle states for each chip select area.
<table>
<tr><th>BCk[1:0]</th><th>Inserted idle states</th></tr>
<tr><td>00_B</td><td>No idle state inserted</td></tr>
<tr><td>01_B</td><td>1 idle state</td></tr>
<tr><td>10_B</td><td>2 idle states</td></tr>
<tr><td>11_B</td><td>3 idle states</td></tr>
</table>
</td></tr>
</table>

Note For access to internal memory, no idle states are inserted.

Caution 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.

(6) RDDLY - Read delay control register

The 8-bit RDDLY register controls the delay of the read strobe $\overline{RD}$ of the external memory interface. It provides the option to delay the rising edge of the $\overline{RD}$ by a half of the bus clock cycle BCLK.

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

Address FFFF FF00$_H$

Initial Value 00$_H$

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

Table 7-23 RDDLY register contents

Bit position	Bit name	Function
0	RDDLYEN	Read strobe control. 0: Rising $\overline{RD}$ edge not delayed 1: Rising $\overline{RD}$ edge delayed by half BCLK clock cycle

---

Caution 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.

---

(7) PRC - Page ROM configuration register

The 16-bit PRC register controls whether a page ROM cycle is on-page or off-page.

The register specifies the address mask. Masked address bits are not considered when deciding between on-page or off-page access. Set the mask according to the number of continuously readable bits.

For page access (cycle is on-page) the register defines the number of inserted data wait cycles.

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

Address FFFF F49A$_H$

Initial Value 7000$_H$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PRW2	PRW1	PRW0	0	0	0	0	0	0	0	0	MA6	MA5	MA4	MA3
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 7-24 PRC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>14 to 12</td><td>PRW[2:0]</td><td>Page ROM on-page wait control.
Sets the number of data waits corresponding to the internal system clock.
<table>
<tr><th>PRW[2:0]</th><th>Inserted data wait states</th></tr>
<tr><td>000_B</td><td>No wait state inserted</td></tr>
<tr><td>001_B</td><td>1 wait state</td></tr>
<tr><td>010_B</td><td>2 wait states</td></tr>
<tr><td>011_B</td><td>3 wait states</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111_B</td><td>7 wait states</td></tr>
</table>
Note: The number of wait states defined in the PRC register is only valid if on-page access is enabled.
If on-page is disabled, the number of wait states is defined by registers DWC0 and DWC1.</td></tr>
<tr><td>3 to 0</td><td>MA[6:3]</td><td>Mask address.
Setting bits MA6 to MA3 masks the corresponding addresses A6 to A3.
<table>
<tr><th rowspan="2">MA6</th><th rowspan="2">MA5</th><th rowspan="2">MA4</th><th rowspan="2">MA3</th><th colspan="3">Number of continuously readable bits</th></tr>
<tr><th>bus width:
8 bits
LBk[1:0]=00_B</th><th>bus width:
16 bits
LBk[1:0]=01_B</th><th>bus width:
32 bits
LBk[1:0]=10_B</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>8 x 8 bits</td><td>4 x 16 bits</td><td>2 x 32 bits</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>16 x 8 bits</td><td>8 x 16 bits</td><td>4 x 32 bits</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>32 x 8 bits</td><td>16 x 16 bits</td><td>8 x 32 bits</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>64 x 8 bits</td><td>32 x 16 bits</td><td>16 x 32 bits</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>128 x 8 bits</td><td>64 x 16 bits</td><td>32 x 32 bits</td></tr>
</table></td></tr>
</table>

Note To initialize an external memory area after a reset, register PRC has to be set if page ROM mode is selected. Do not change this register after initialization. Do not access external page ROM devices before initialization is finished.

Caution 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.

7.4 Page ROM Controller

In page ROM mode the microcontroller reads consecutive data from one page by inserting the wait cycles defined by PRC.PRW[2:0] instead of wait cycles defined in registers DWC0 and DWC1.

The page ROM controller decides whether a page ROM cycle is on-page or off-page. To do so, it buffers the address of the previous cycle and compares it with the address of the current cycle. If the compare result proves that the read access is on-page the read cycle is performed with wait cycles defined by PRC.PRW[2:0].

In the page ROM configuration register (PRC), one or more of the address bits (A3 to A6) are set as masking addresses (no comparison is made for these addresses). The masking address is chosen according to the configuration of the connected page ROM and the number of continuously readable bits.

Wait control for normal access (off-page) and page access (on-page) is specified by different registers: For page access, wait control is performed according to PRC register setting. For normal access, wait control is performed according to DWC0 and DWC1 register settings.

The following figures show the on-page/off-page judgment during page ROM connection for a 16-Mbit page ROM and for different data bus widths.

(1) 8-bit data bus width

The page size or the number of continuously readable bits is 32 x 8 bit. To provide 32 addresses, a 5-bit on-page address is required. Therefore, set PRC.MA[6:3] = 0011_B.

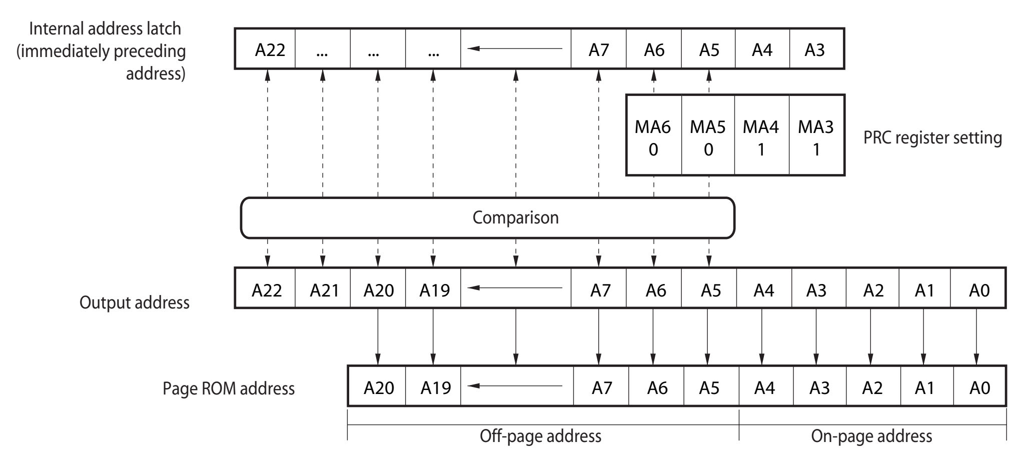


Figure 7-5 16-Mbit page ROM (2 M × 8 bits), page size 32 x 8 bit

(2) 16-bit data bus width

The page size or the number of continuously readable bits is 8 x 16 bit. To provide 8 addresses, a 3-bit on-page address is required. Therefore, set PRC.MA[6:3] = 0001_B.

Note For a 16-bit data bus, bit A0 of the output address is not used.

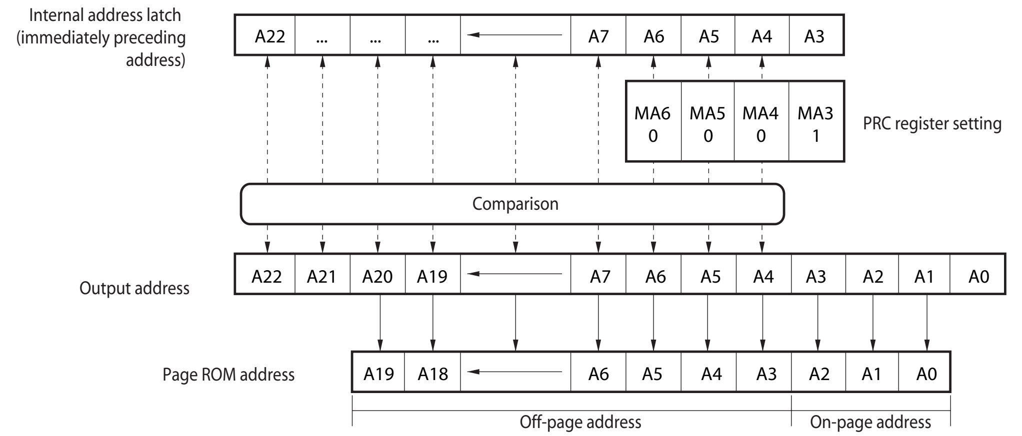


Figure 7-6 16-Mbit page ROM (1 M × 16 bits), page size 8 x 16 bit

(3) 32-bit data bus width

The page size or the number of continuously readable bits is 2 x 32 bit. To provide 2 addresses, a 1-bit on-page address is required. Therefore, set PRC.MA[6:3] = 0000_B.

Note For a 32-bit data bus, bits A0 and A1 of the output address are not used.

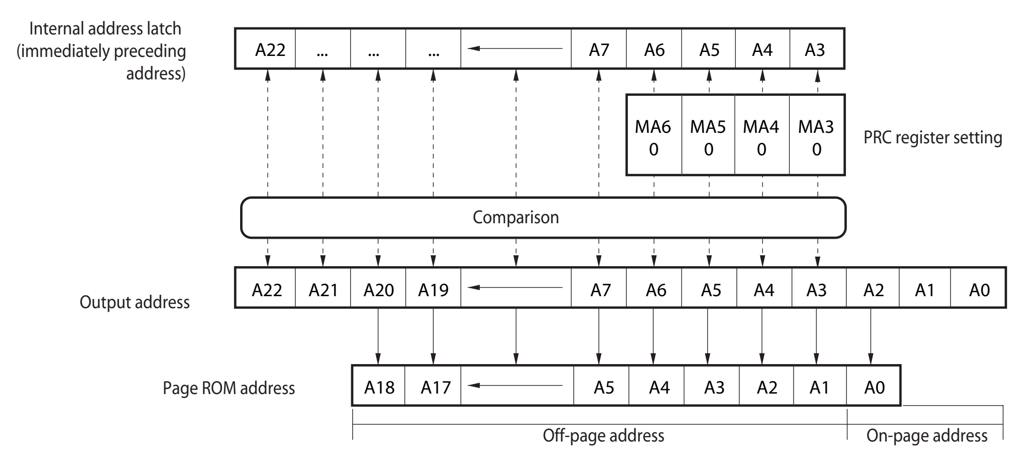


Figure 7-7 16-Mbit page ROM (512 k × 32 bits), page size 2 x 32 bit

7.5 Configuration of Memory Access

The microcontroller device supports interfacing with various memory devices. Therefore, the endian format, wait functions and idle state insertions can be configured.

7.5.1 Endian format

The endian format is specified with the endian configuration register (BEC). It defines the byte order in which word data is stored.

"Big endian" means that the high-order byte of the word is stored in memory at the lowest address, and the low-order byte at the highest address. Therefore, the base address of the word addresses the high-order byte:

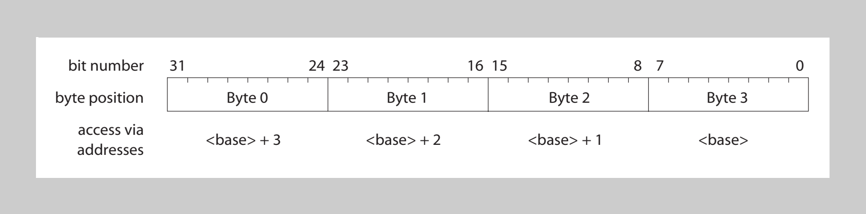


Figure 7-8 Big endian addresses within a word

"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:

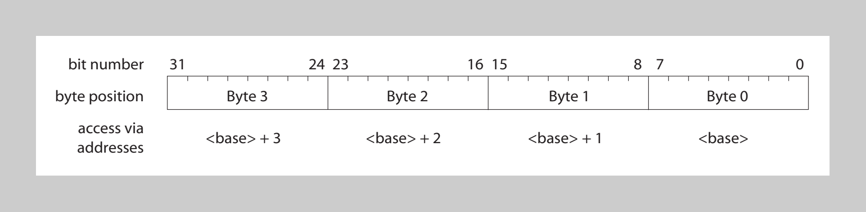


Figure 7-9 Little endian addresses within a word

7.5.2 Wait function

Several wait functions are supported:

(1) Address setup wait

The microcontroller device allows insertion of address setup wait states before the first access cycle (T1 state). The number of address setup wait states can be set with the address setup wait control register ASC for each CS area.

Address setup wait states can be inserted when accessing SRAM or page ROM.

(2) 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 first access cycle (T1 state).

The number of wait states can be specified by data wait control registers DWC0 and DWC1.

For on-page access of a page ROM, wait control is performed according to page ROM configuration register (PRC) setting. The settings of registers DWC0 and DWC1 are neglected.

(3) External wait function

Each read or write operation takes at least two cycles (T1 and T2). 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 rising edge of the clock in the T1 and TW states. Depending on the level of the $\overline{\text{WAIT}}$ signal at sampling timing, a wait state is inserted or not.

(4) 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 7-10* shows that as long as any of the two waits is active, a wait cycle will be performed.

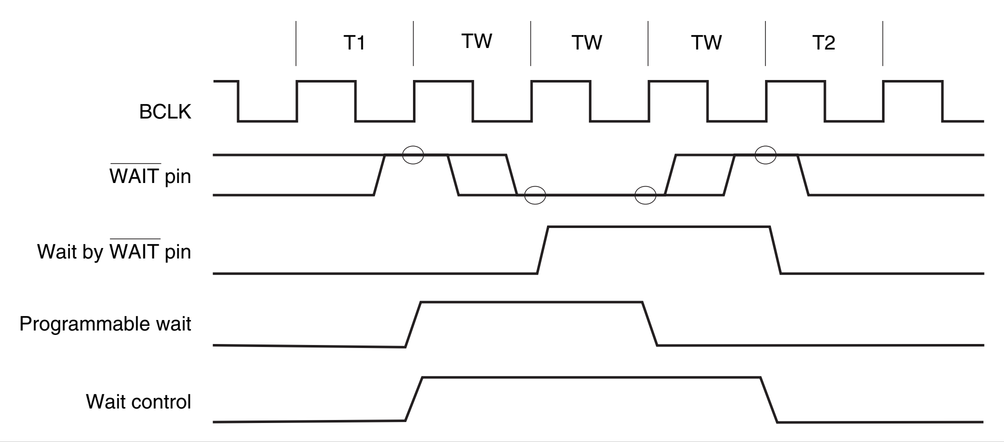


Figure 7-10 **Example of wait insertion**

Note The circles indicate the sampling timing.

7.5.3 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 T2 state. Idle states are inserted to meet the data output float delay time on memory read access for each CS space.

Idle states are used to guarantee the interval until the external data bus is released by memory. The next bus cycle is started after the idle state(s).

Idle states can be inserted after T2 state when accessing SRAM, external I/O, external ROM, or page ROM.

The number of idle states can be specified by program using the bus cycle control register (BCC).

7.6 External Devices Interface Timing

This section presents examples of write and read operations. The states are abbreviated as:

- T1 and T2 states: Basic states for access.
- TW state: Wait state that is inserted according to the DWC0 and DWC1 register settings and according to the $\overline{\text{WAIT}}$ input.
- TASW state: Address setting wait state that is inserted according to the ASC register settings.
- TI state: Idle state that is inserted according to the BCC register settings.

Note For access to page ROM, see *“Page ROM Access Timing” on page 291*.

7.6.1 Writing to external devices

This section shows typical sequences of writing data to external devices.

(1) Write with external wait cycle

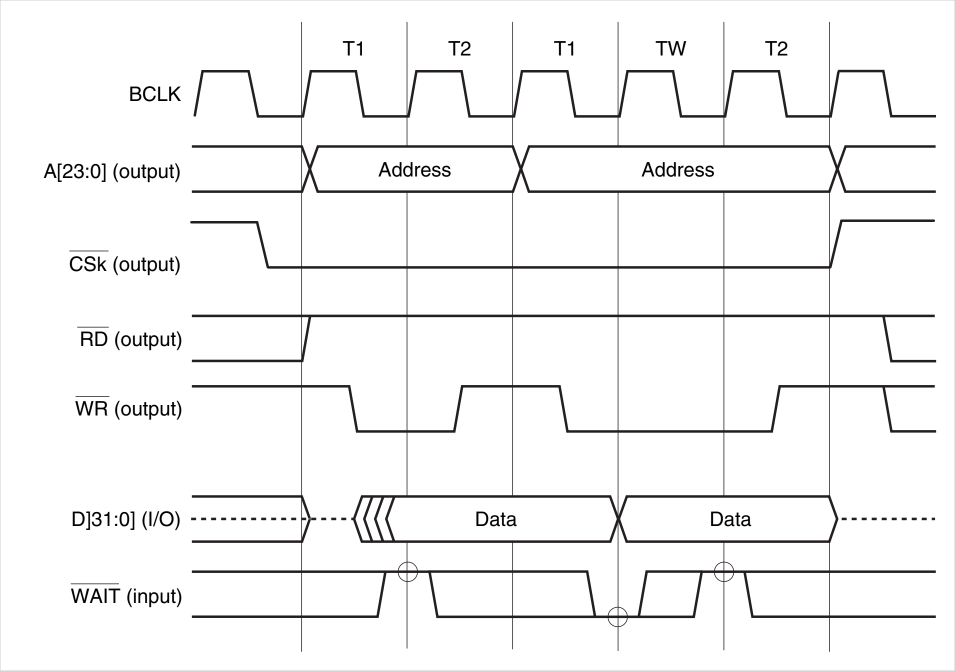


Figure 7-11 Timing: write data

Register settings:

- BCTm.BTk0 = 0 (connected external device is SRAM or external I/O)
- ASC.ACk[1:0] = 00_B (no address setup wait states inserted)
- DWCm.DWk[2:0] = 000_B (no programmable data wait states inserted)
- BCC.BCk[1:0] = 00_B (no idle states inserted)

Note
1. The circles indicate the sampling timing.
2. The broken line indicates the high-impedance state (bus is not driven).

The data has to be stable at the rising edge of the $\overline{\text{WR}}$ signal. For details refer to the Electrical Target Specification.

(2) Write with address setup wait and idle state insertion

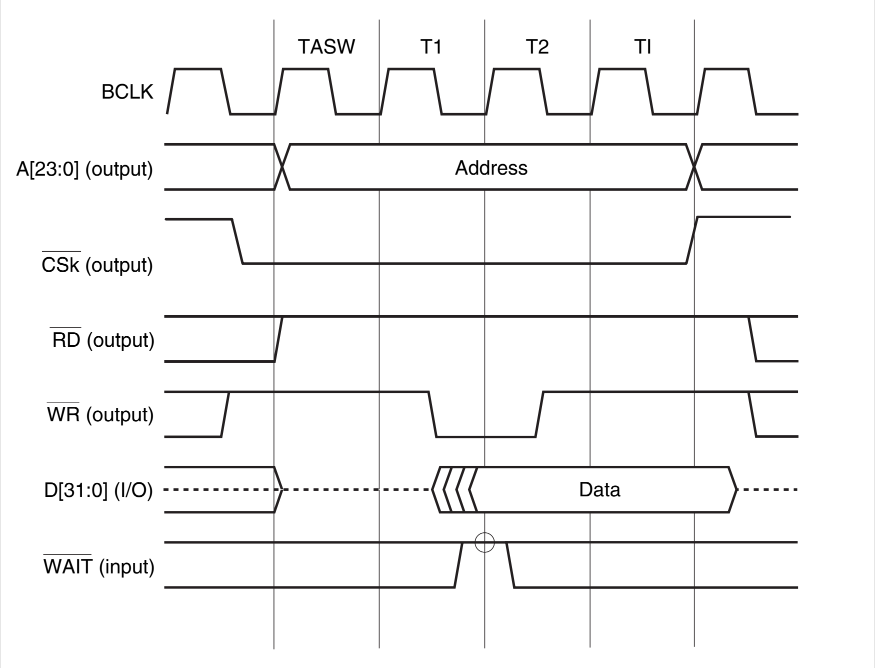


Figure 7-12 Timing: write data with address setup wait and idle state insertion

Register settings:

- BCTm.BTk0 = 0 (connected external device is SRAM or external I/O)
- ASC.ACk[1:0] = 01_B (one address setup wait state inserted)
- DWCm.DWk[2:0] = 000_B (no programmable data wait states inserted)
- BCC.BCk[1:0] = 01_B (one idle state inserted)

Note 1. The circles indicate the sampling timing.

2. The broken line indicates the high-impedance state (bus is not driven).

The data has to be stable at the rising edge of the $\overline{\text{WR}}$ signal. For details refer to the Electrical Target Specification.

7.6.2 Reading from external devices

This section shows typical sequences of reading data from external devices.

(1) Read with external wait cycle

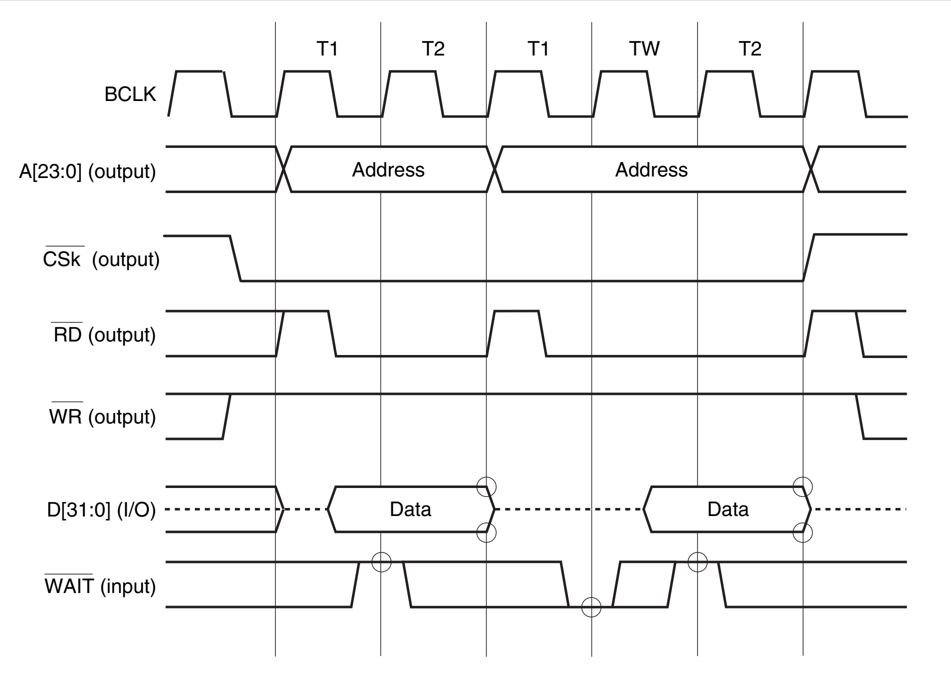


Figure 7-13 Timing: read data

Register settings:

- BCTm.BTk0 = 0 (connected external device is SRAM or external I/O)
- ASC.ACk[1:0] = 00_B (no address setup wait states inserted)
- DWCm.DWk[2:0] = 000_B (no programmable data wait states inserted)
- BCC.BCk[1:0] = 00_B (no idle states inserted)

Note
1. The circles indicate the sampling timing.
2. The broken line indicates the high-impedance state (bus is not driven).

(2) Read with address setup wait and idle state insertion

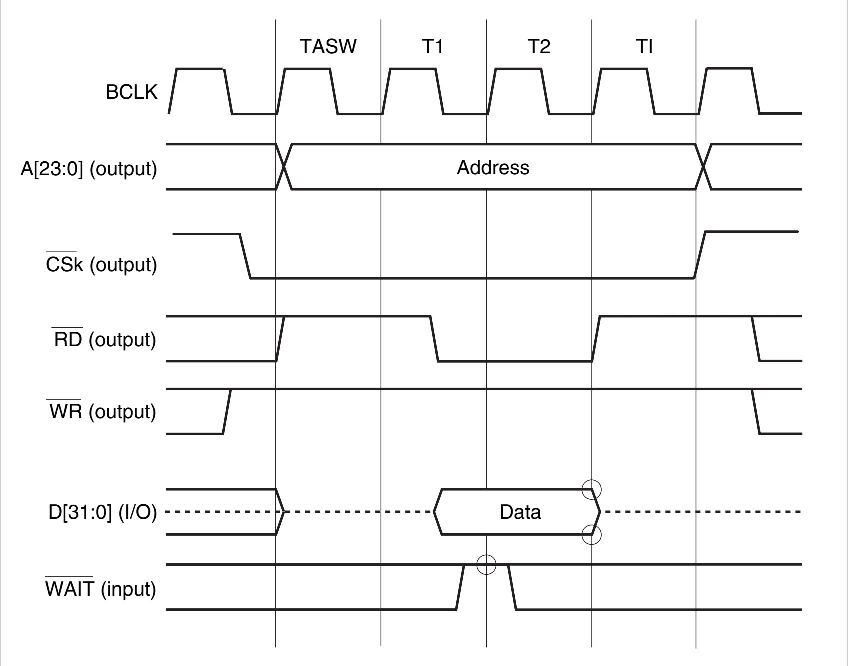


Figure 7-14 Timing: read data with address setup wait and idle state insertion

Register settings:

- BCTm.BTk0 = 0 (connected external device is SRAM or external I/O)
- ASC.ACk[1:0] = 01_B (one address setup wait state inserted)
- DWCm.DWk[2:0] = 000_B (no programmable data wait states inserted)
- BCC.BCk[1:0] = 01_B (one idle state inserted)

Note 1. The circles indicate the sampling timing.

2. The broken line indicates the high-impedance state (bus is not driven).

7.6.3 Read-write operation on external devices

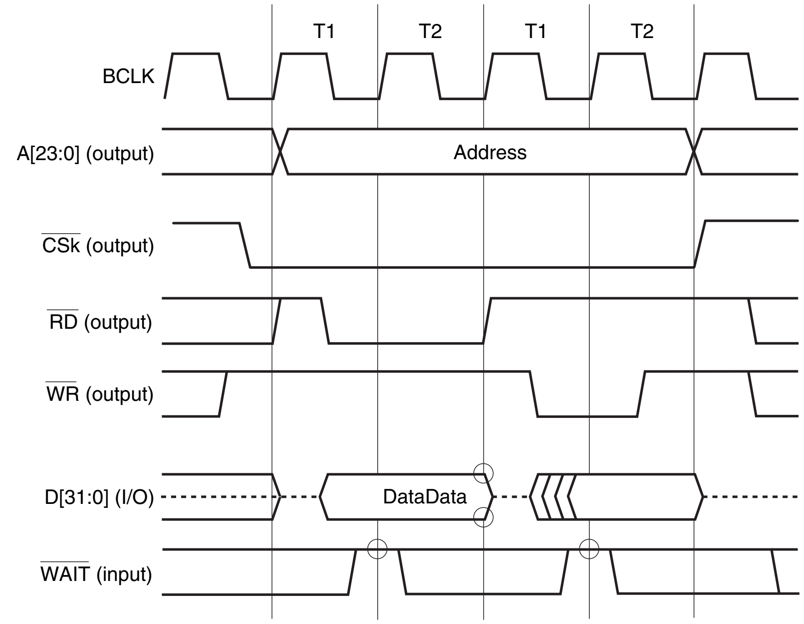


Figure 7-15 **Read-write operation**

Register settings:

- BCTm.BTk0 = 0 (connected external device is SRAM or external I/O)
- ASC.ACk[1:0] = 00_B (no address setup wait states inserted)
- DWCm.DWk[2:0] = 000_B (no programmable data wait states inserted)
- BCC.BCk[1:0] = 00_B (no idle states inserted)

Note
1. The circles indicate the sampling timing.
2. The broken line indicates the high-impedance state (bus is not driven).

The data has to be stable at the rising edge of the $\overline{\text{WR}}$ signal. For details refer to the Electrical Target Specification.

7.6.4 Write-read operation on external devices

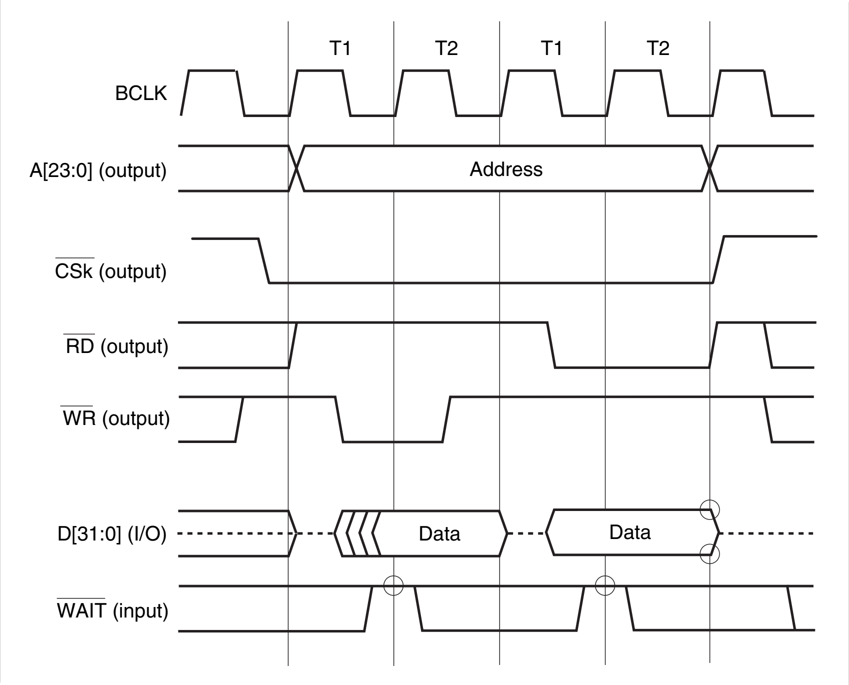


Figure 7-16 Write-read operation

Register settings:

- BCTm.BTk0 = 0 (connected external device is SRAM or external I/O)
- ASC.ACk[1:0] = 00_B (no address setup wait states inserted)
- DWCm.DWk[2:0] = 000_B (no programmable data wait states inserted)
- BCC.BCk[1:0] = 00_B (no idle states inserted)

Note

1. The circles indicate the sampling timing.
2. The broken line indicates the high-impedance state (bus is not driven).

The data has to be stable at the rising edge of the $\overline{\text{WR}}$ signal. For details refer to the Electrical Target Specification.

7.7 Page ROM Access Timing

This section presents examples of read operations on page ROM. The states are abbreviated as:

- T1 and T2 states: Basic states for access.
- TW state: Wait state that is inserted according to the DWC0 and DWC1 register settings and according to the $\overline{\text{WAIT}}$ input.
- TOW state: Wait state that is inserted according to the PRC.PRW[2:0] settings and according to the WAIT input.
- TO1 and TO2: On-page states
- TASW state: Address setting wait state that is inserted according to the ASC register settings.
- TI state: Idle state that is inserted according to the BCC register settings.

7.7.1 Half word/word access with 8-bit bus or word access with 16-bit bus

(1) Read operation

Note that during on-page access, less data wait states are inserted than during off-page access.

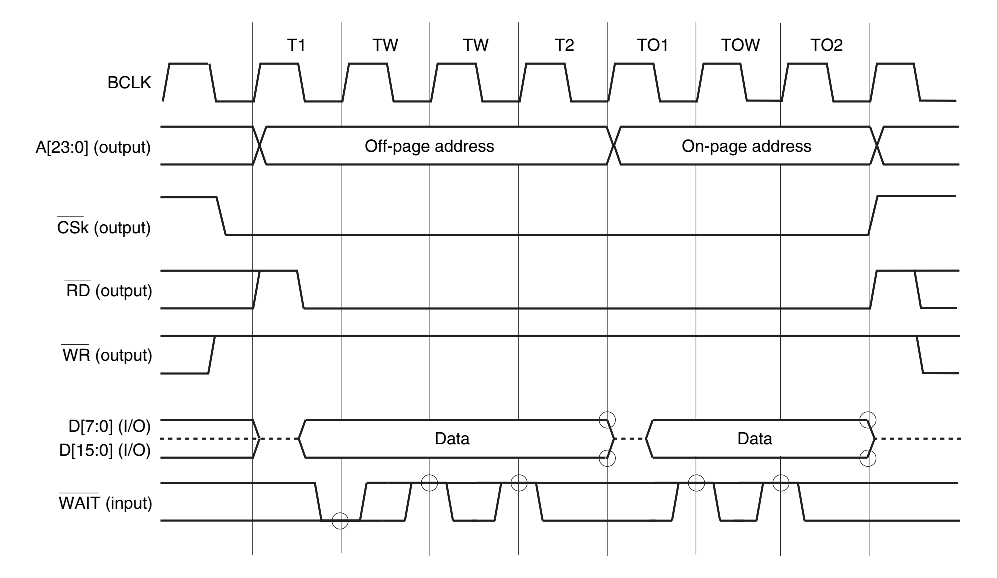


Figure 7-17 Reading page ROM

- Register settings:
- BCTm.BTk0 = 1 (connected external device is page ROM)
- ASC.ACk[1:0] = 00_B (no address setup wait states inserted)
- DWCm.DWk[2:0] = 010_B (two programmable data wait states for off-page access inserted)
- PRC.PRW[2:0] = 001_B (one programmable data wait state for on-page access inserted)
- BCC.BCk[1:0] = 00_B (no idle states inserted)

Note
1. The circles indicate the sampling timing.
2. The broken line indicates the high-impedance state (bus is not driven).

(2) Read operation with address setup wait states and idle state insertion

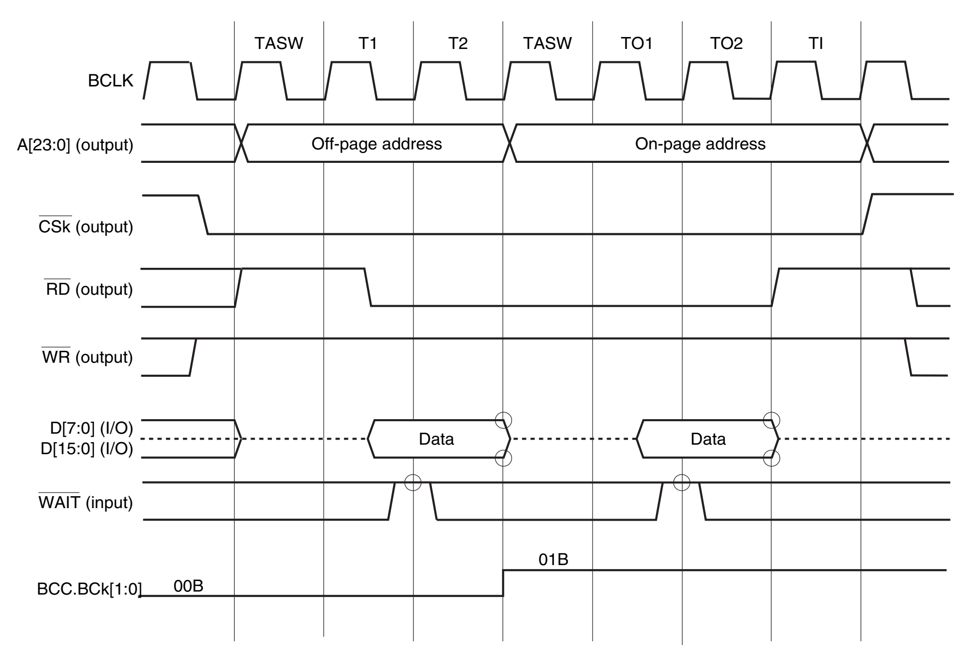


Figure 7-18 Reading page ROM with address setup wait states and idle state insertion

Register settings:

- BCTm.BTk0 = 1 (connected external device is page ROM)
- ASC.ACk[1:0] = 01_B (one address setup wait state inserted)
- DWCm.DWk[2:0] = 000_B (no programmable data wait states for off-page access inserted)
- PRC.PRW[2:0] = 000_B (no programmable data wait states for on-page access inserted)
- BCC.BCk[1:0] : see *Figure 7-18*

Note
1. The circles indicate the sampling timing.
2. The broken line indicates the high-impedance state (bus is not driven).

7.7.2 Byte access with 8-bit bus or byte/half word access with 16-bit bus

(1) Read operation

Note that during on-page access, less data wait states are inserted than during off-page access.

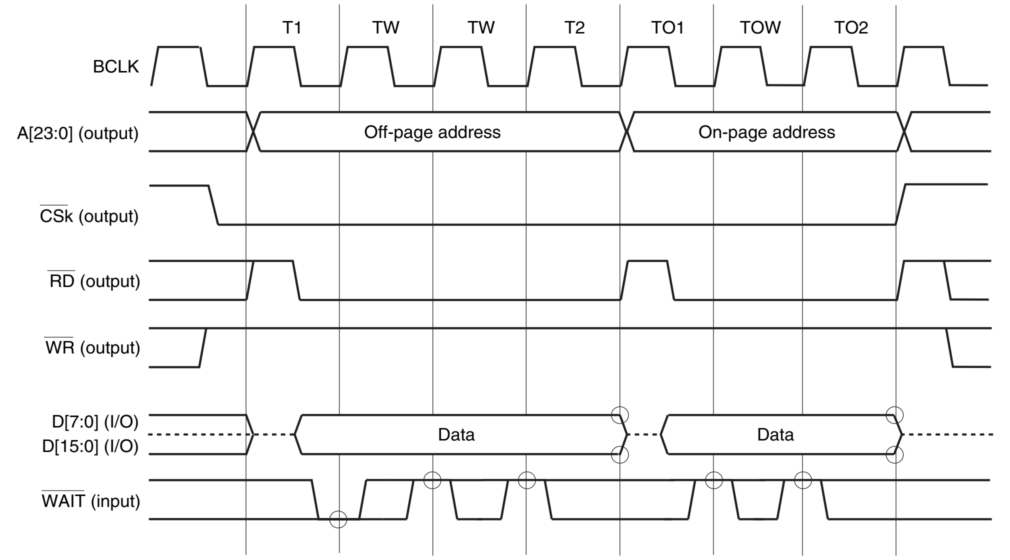


Figure 7-19 Reading page ROM

Register settings:

- BCTm.BTk0 = 1 (connected external device is page ROM)
- ASC.ACk[1:0] = 00_B (no address setup wait states inserted)
- DWCm.DWk[2:0] = 010_B (two programmable data wait states for off-page access inserted)
- PRC.PRW[2:0] = 001_B (one programmable data wait state for on-page access inserted)
- BCC.BCk[1:0] = 00_B (no idle states inserted)

Note
1. The circles indicate the sampling timing.
2. The broken line indicates the high-impedance state (bus is not driven).

(2) Read operation with address setup wait states and idle state insertion

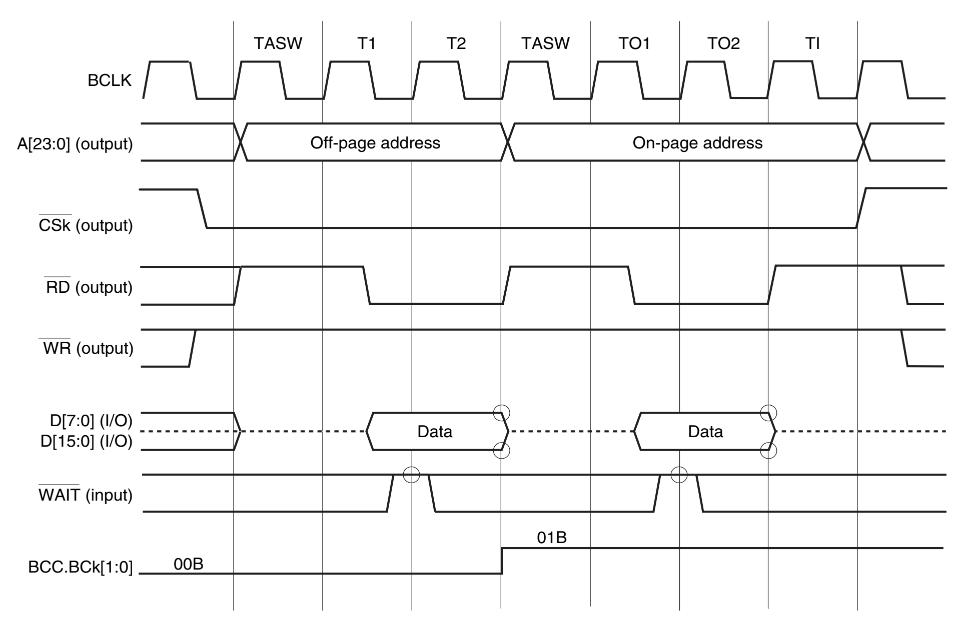


Figure 7-20 Reading page ROM with address setup wait states and idle state insertion

Register settings:

- BCTm.BTk0 = 1 (connected external device is page ROM)
- ASC.ACk[1:0] = 01_B (one address setup wait state inserted)
- DWCm.DWk[2:0] = 000_B (no programmable data wait states for off-page access inserted)
- PRC.PRW[2:0] = 000_B (no programmable data wait states for on-page access inserted)
- BCC.BCk[1:0] : see *Figure 7-20*

Note **1.** The circles indicate the sampling timing.

2. The broken line indicates the high-impedance state (bus is not driven).

7.8 Data Access Order

7.8.1 Access to 8-bit data busses

This section shows how byte, half word and word accesses are performed for an 8-bit data bus.

(1) Byte access (8 bits)

(a) Little endian

Figure 7-21 Left: Access to even address (2n)
Right: Access to odd address (2n + 1)

(b) Big endian

Figure 7-22 Left: Access to even address (2n)
Right: Access to odd address (2n + 1)

(2) Halfword access (16 bits)

(a) Little endian

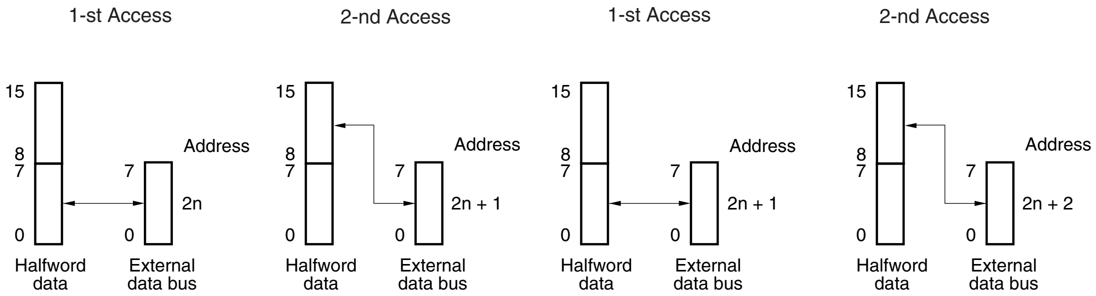


Figure 7-23 Left: Access to even address (2n)
Right: Access to odd address (2n + 1)

(b) Big endian

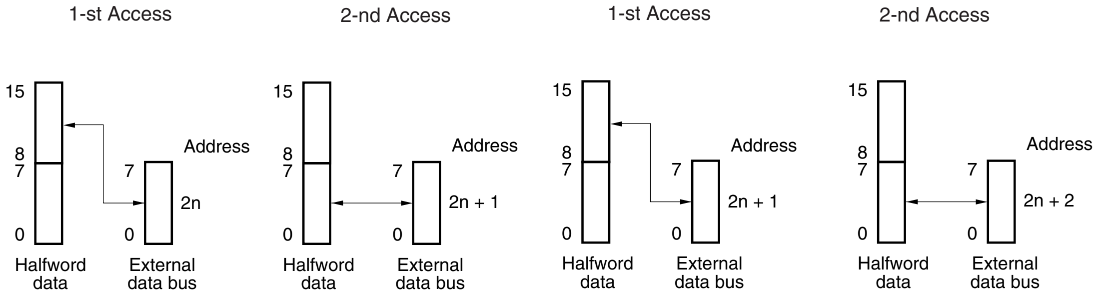


Figure 7-24 Left: Access to even address (2n)
Right: Access to odd address (2n + 1)

(3) Word access (32 bits)

(a) Little endian

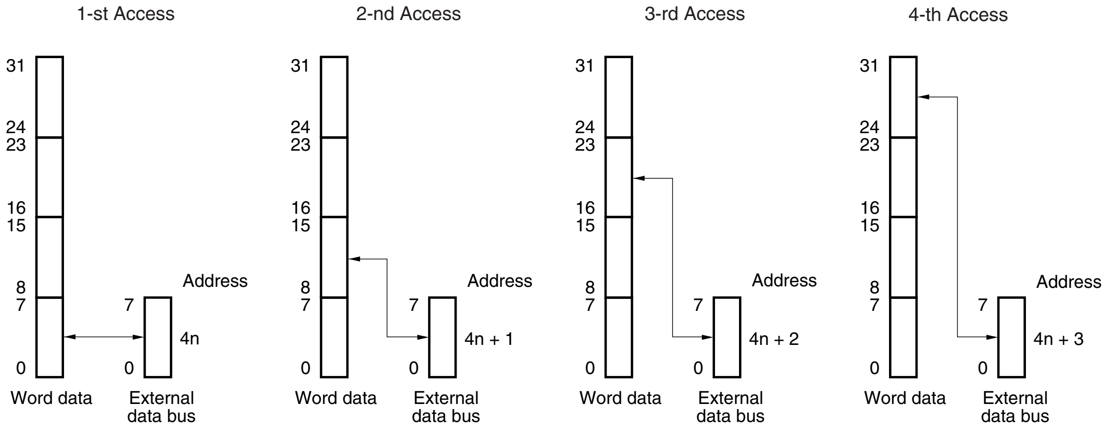


Figure 7-25 Access to address 4n

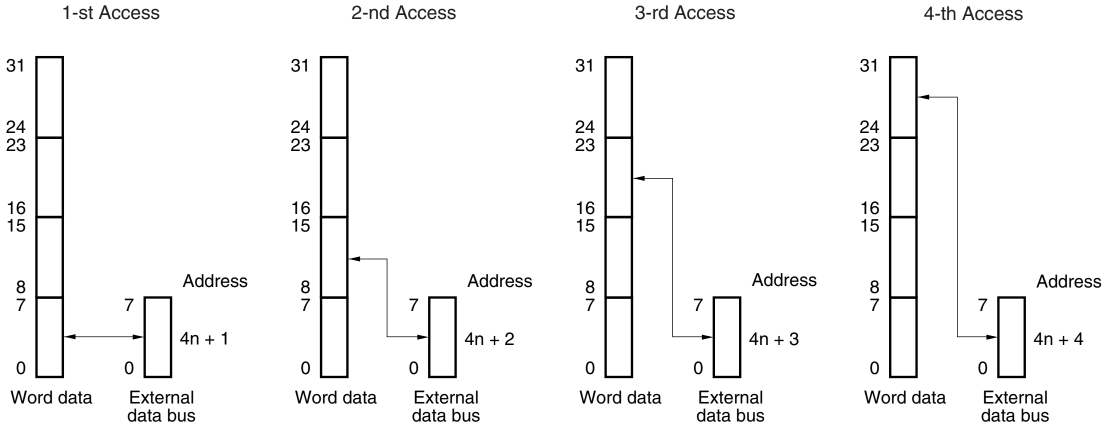


Figure 7-26 Access to address 4n + 1

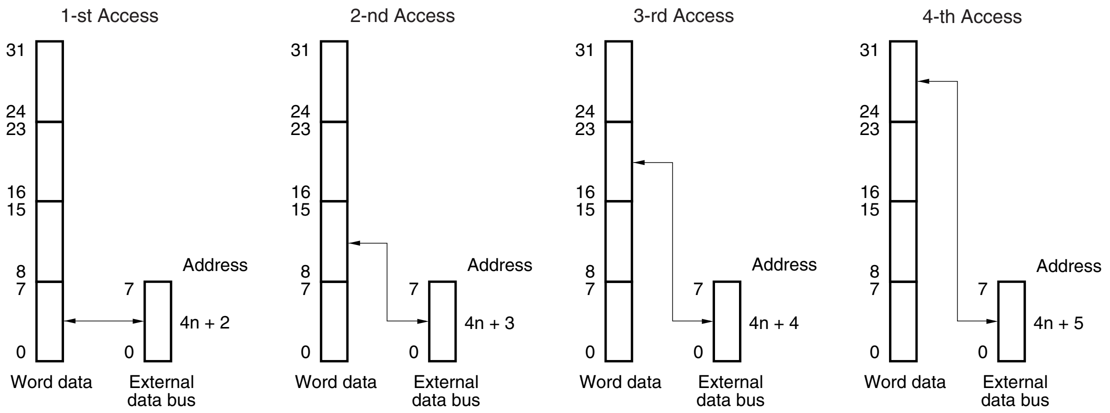


Figure 7-27 Access to address 4n + 2

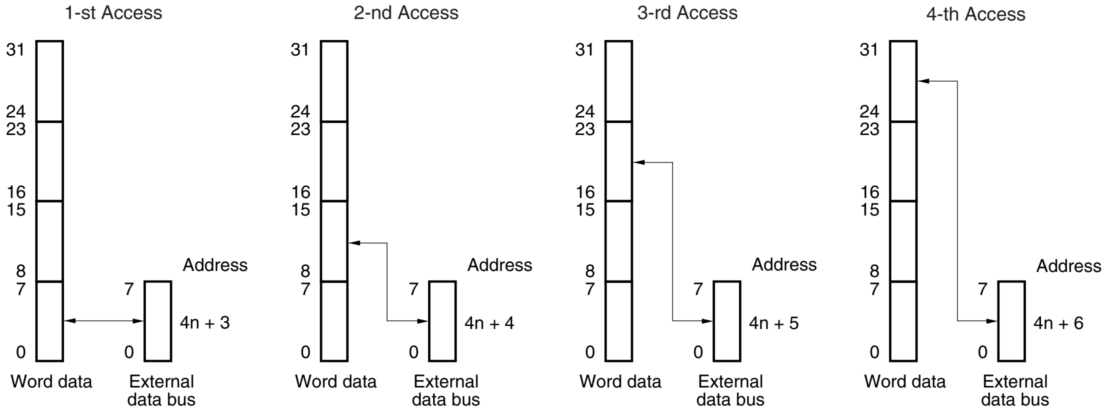


Figure 7-28 Access to address 4n + 3

(b) Big endian

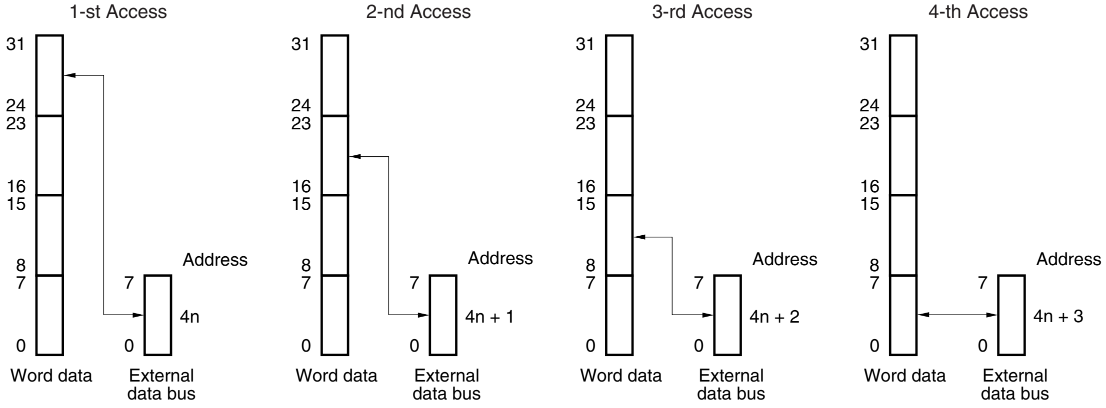


Figure 7-29 Access to address 4n

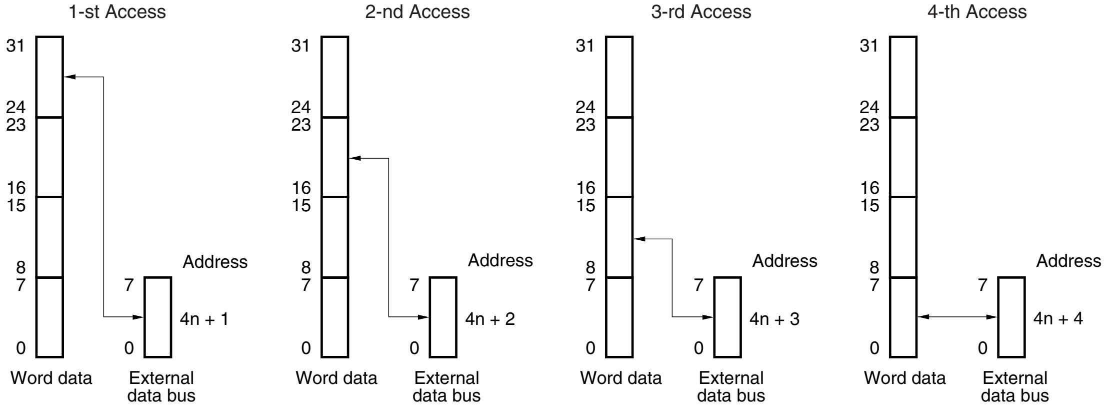


Figure 7-30 Access to address 4n + 1

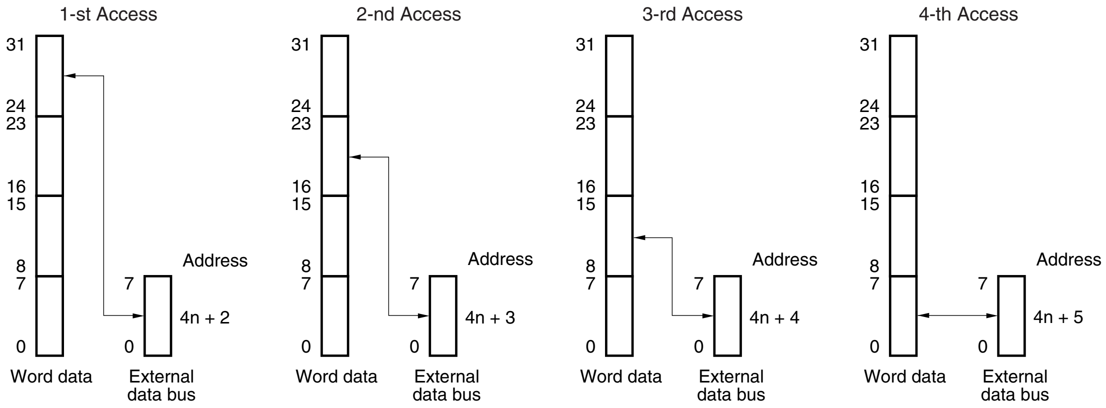


Figure 7-31 Access to address 4n + 2

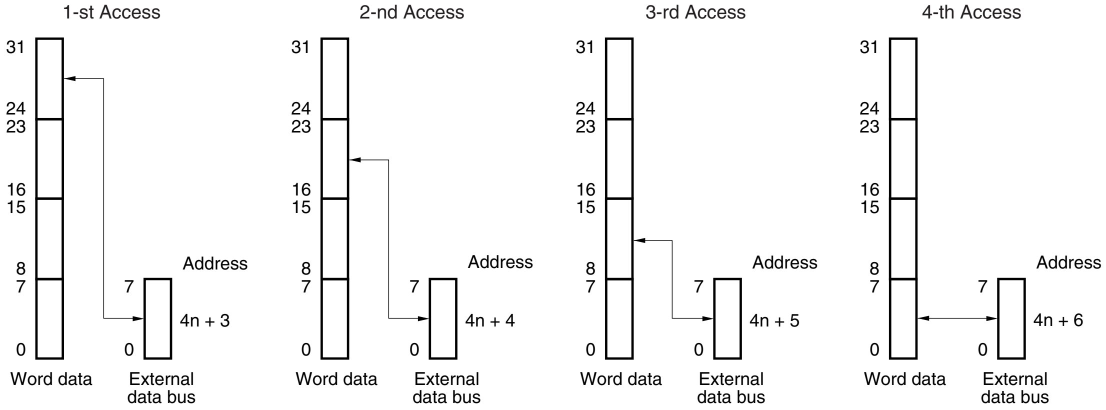


Figure 7-32 Access to address 4n + 3

7.8.2 Access to 16-bit data busses

This section shows how byte, half word and word accesses are performed for a 16 bit data bus.

Access all data in order starting from the lower order side.

(1) Byte access (8 bits)

(a) Little endian

Figure 7-33 Left: Access to even address (2n)
Right: Access odd address (2n + 1)

(b) Big endian

Figure 7-34 Left: Access to even address (2n)
Right: Access to odd address (2n + 1)

(2) Halfword access (16 bits)

(a) Little endian

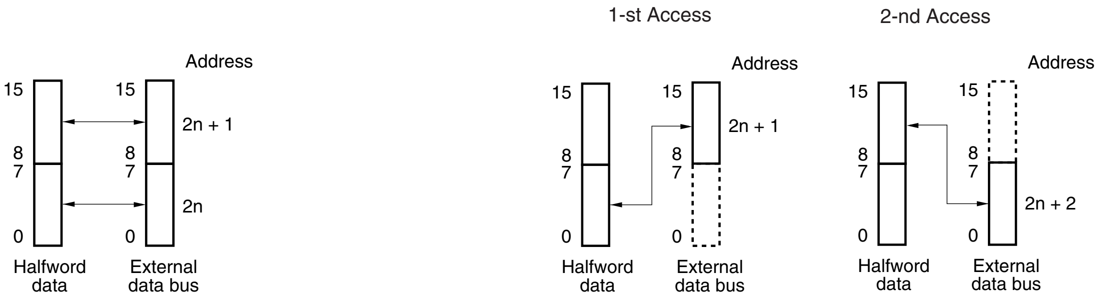


Figure 7-35 Left: Access to even address (2n)
Right: Access to odd address (2n + 1)

(b) Big endian

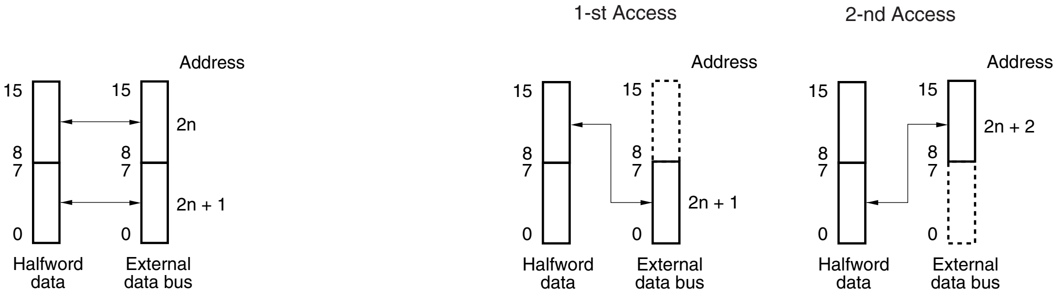


Figure 7-36 Left: Access to even address (2n)
Right: Access to odd address (2n + 1)

(3) Word access (32 bits)

(a) Little endian

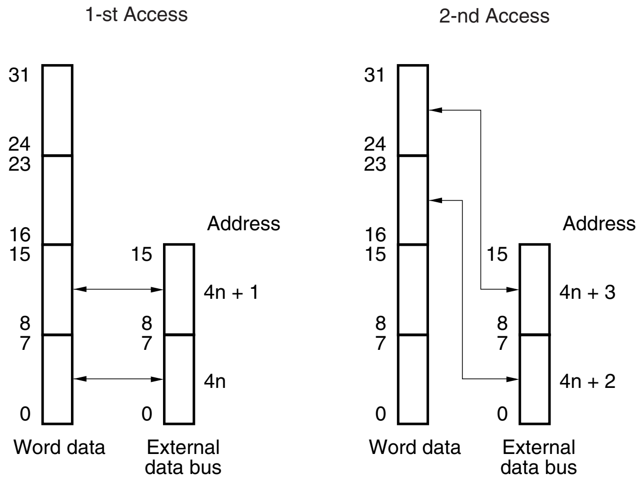


Figure 7-37 Access to address 4n

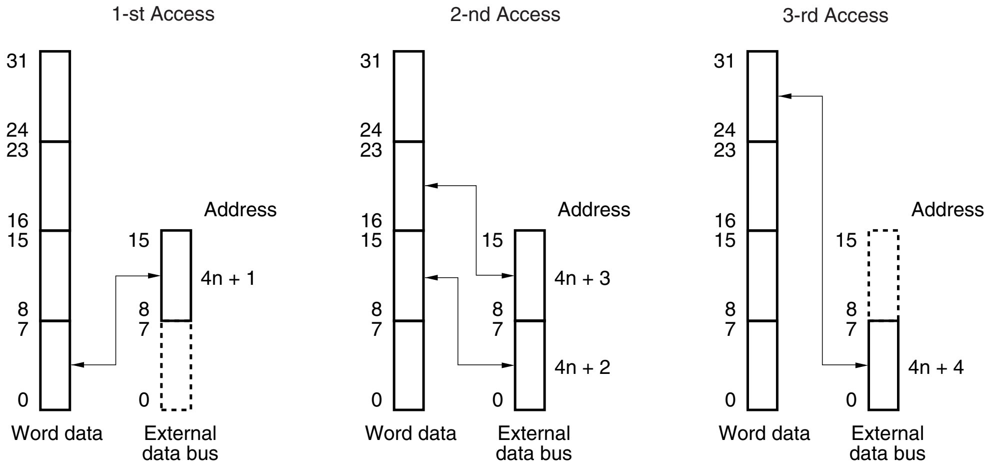


Figure 7-38 Access to address 4n + 1

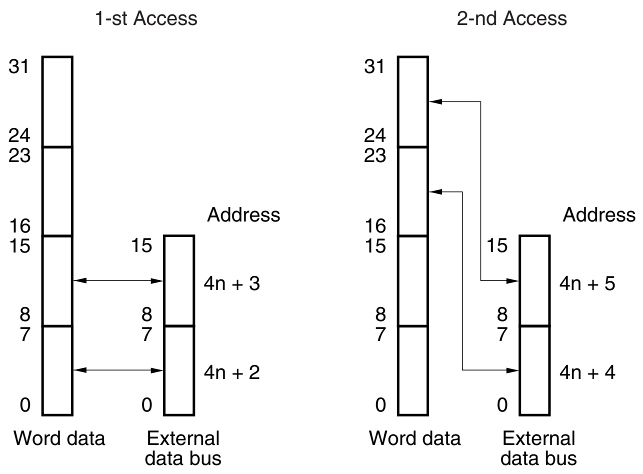


Figure 7-39 Access to address 4n + 2

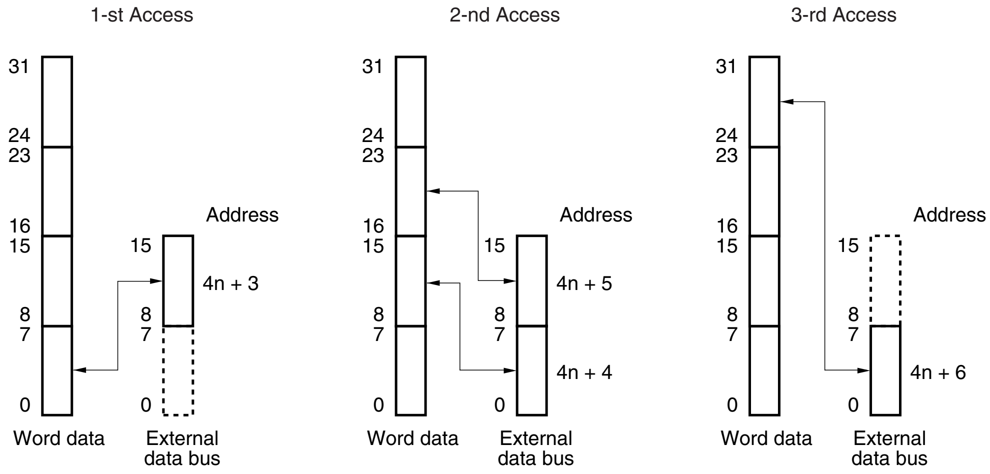


Figure 7-40 Access to address 4n + 3

(b) Big endian

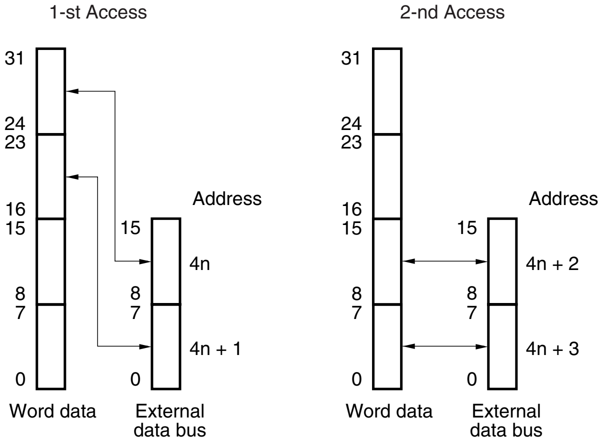


Figure 7-41 Access to address 4n

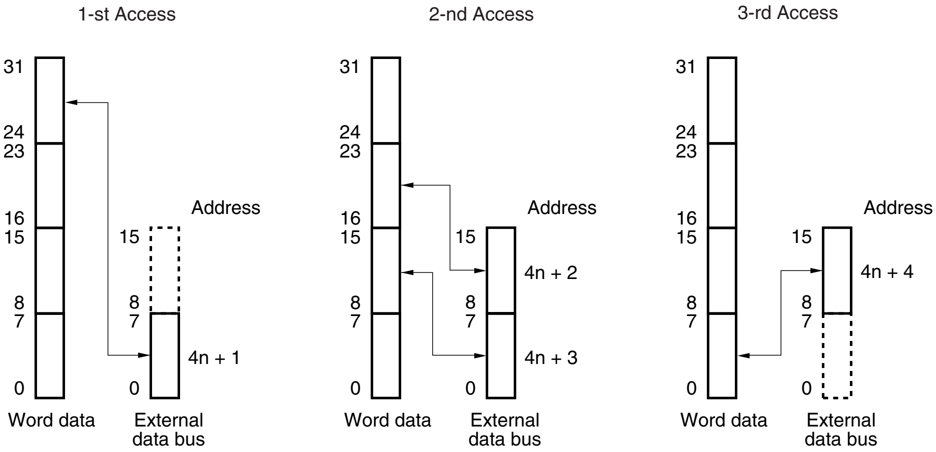


Figure 7-42 Access to address 4n + 1

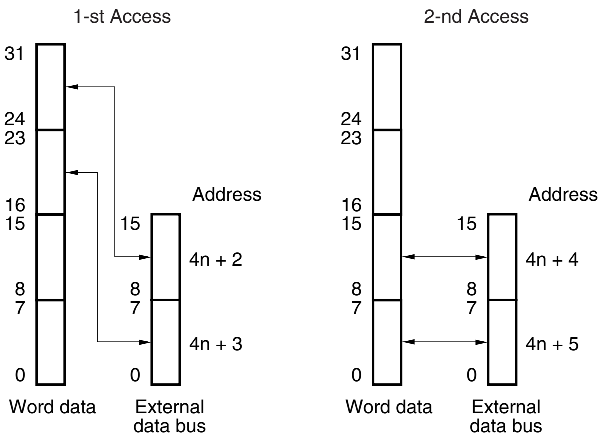


Figure 7-43 Access to address 4n + 2

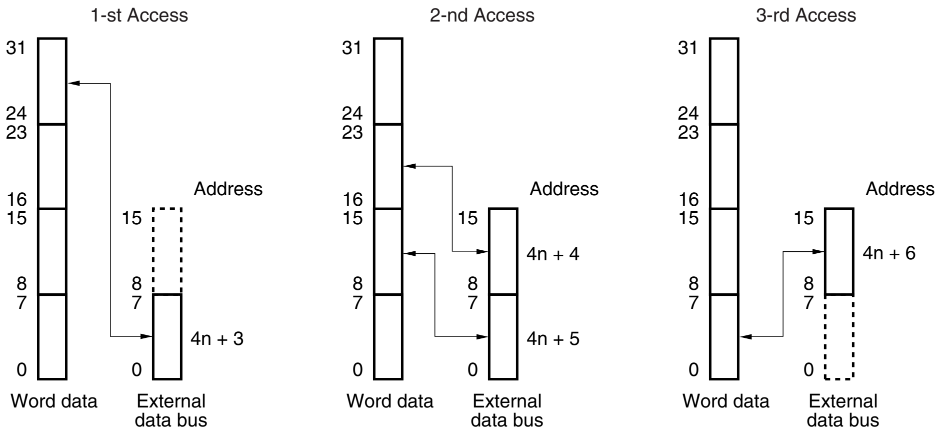


Figure 7-44 Access to address 4n + 3

Chapter 8 DMA Controller (DMAC)

The microcontroller includes a direct memory access (DMA) controller (DMAC) that executes and controls DMA transfers.

Note Throughout this chapter, the individual channels of the DMA Controller are identified by “n”.

The DMAC controls data transfer between memory and I/O or among I/Os, based on DMA requests issued by the on-chip peripheral I/O, or software triggers.

8.1 Features

- Four independent DMA channels
- Transfer units: 8, 16 and 32 bits
- Maximum transfer count: 65536 (2^{16})
- Two transfer modes independently selectable for each DMA channel
  - Single transfer mode
  - Block transfer mode
- Transfer requests
  - Requests by dedicated peripheral interrupts of
    - µPD70(F)3420, µPD70(F)3421, µPD70F3423: CSIB0, CSIB1, UARTA0, UARTA1, IIC0, IIC1, TMG0, TMP0, TMP1, TMZ0, TMZ1, TMZ2, ADC
    - µPD70F3424, µPD70F3425: CSIB0…CSIB2, UARTA0, UARTA1, IIC0, IIC1, TMG0, TMP0, TMP1, TMZ0, TMZ1, TMZ2, LCDIF, ADC
  - Requests by software trigger
- Transfer objects

Source \ Destination	Internal RAM	Peripherals
Internal RAM	–	√
Peripherals	√	√

- DMA transfer completion flag
- Automatic restart function
- Forcible DMA termination by NMI

8.2 Peripheral and CPU Clock Settings

In order to ensure safe capture of DMA trigger signals from the involved peripheral functions, a certain minimum relation between the operation clock of the concerned peripheral function and the CPU system has to be regarded.

In the following table the minimum CPU system clock frequency f_{VBCLK} is given for all peripheral functions operation clocks.

Table 8-1 Peripheral functions and CPU system clocks for DMA transfers (1/2)

Peripheral	Clock controller settings	SPCLKn, PCLKn configuration		Input clock [MHz]	Minimum f_{VBCLK} [MHz]
		Peripheral clock	Source		
ADC	SCC = 00_H	SPCLK0	MainOsc	4	6.00
	SCC = 01_H	SPCLK0	PLL/2	16	24.00
	SCC = 03_H SCPS.SPSPS[2:0] = 001_B SSCG: 32 MHz	SPCLK0	f_{SSCGPS}	16	24.00
	SCC = 03_H SCPS.SPSPS[2:0] = 011_B SSCG: 48 MHz	SPCLK0	f_{SSCGPS}	12	18.00
UARTA	CKC.PERIC = 0	PCLK1	MainOsc	4	6.00
	CKC.PERIC = 1		PLL/4	8	12.00
		PCLK2	MainOsc	4	6.00
		PCLK3	MainOsc/2	2	3.00
		PCLK4	MainOsc/4	1	1.5
		PCLK5	MainOsc/8	0.5	0.75
		PCLK6	MainOsc/16	0.25	0.38
		PCLK7	MainOsc/32	0.125	0.19
		PCLK8	MainOsc/64	0.0625	0.09
CSIB	CKC.PERIC = 0	PCLK1	MainOsc	4	6.00
	CKC.PERIC = 1		PLL/4	8	12.00
		PCLK2	MainOsc	4	6.00
		PCLK3	MainOsc/2	2	3.00
		PCLK4	MainOsc/4	1	1.50
		PCLK5	MainOsc/8	0.5	0.75
		PCLK6	MainOsc/16	0.250	0.38
	SCC = 00_H	SPCLK1 via Baud Rate Generator	MainOsc	max. 4 min. 0.002	6.00 0.003
	SCC = 01_H		PLL/4	max. 8 min. 0.002	12.00 0.003
	SCC = 03_H SCPS.SPSPS[2:0] = 001_B SSCG: 32 MHz		$f_{SSCGPS}/2$	max. 8 min. 0.002	12.00 0.003
	SCC = 03_H SCPS.SPSPS[2:0] = 011_B SSCG: 48 MHz		$f_{SSCGPS}/2$	max. 8 min. 0.002	12.00 0.003

Table 8-1 Peripheral functions and CPU system clocks for DMA transfers (2/2)

Peripheral	Clock controller settings		SPCLKn, PCLKn configuration		Input clock [MHz]	Minimum f_{VBCLK} [MHz]
			Peripheral clock	Source		
IIC	ICC = 00_H		IICLK	MainOsc	4	6.00
	ICC = 72_H			PLL / 4.5	7.11	10.67
TMZ	all settings		PCLK1	MainOsc	4	6.00
TMP	CKC.PERIC =	0	PCLK0	MainOsc	4	6.00
		1		PLL/2	16	24.00
TMG	SCC = 00_H		SPCLK0	MainOsc	4	6.00
	SCC = 01_H			PLL/2	16	24.00
	SCC = 03_H SCPS.SPSPS[2:0] = 001_B SSCG: 32 MHz			f_{SSCGPS}	16	24.00
	SCC = 03_H SCPS.SPSPS[2:0] = 011_B SSCG: 48 MHz			f_{SSCGPS}	12	18.00
LCDIF	SCC = 00_H		SPCLK0	MainOsc	4	6.00
			SPCLK1	MainOsc	4	6.00
			SPCLK2	MainOsc	4	6.00
			SPCLK5	MainOsc/	0.5	0.75
	SCC = 01_H		SPCLK0	PLL/2	16	24.00
			SPCLK1	PLL/4	8	12.00
			SPCLK2	MainOsc	4	6.00
			SPCLK5	MainOsc/	0.5	0.75
	SCC = 03_H SCPS.SPSPS[2:0] = 001_B SSCG: 32 MHz		SPCLK0	f_{SSCGPS}	16	24.00
			SPCLK1	$f_{SSCGPS}/2$	8	12.00
			SPCLK2	$f_{SSCGPS}/4$	4	6.00
			SPCLK5	$f_{SSCGPS}/32$	0.5	0.75
	SCC = 03_H SCPS.SPSPS[2:0] = 011_B SSCG: 48 MHz		SPCLK0	f_{SSCGPS}	12	18.00
			SPCLK1	$f_{SSCGPS}/2$	6	9
			SPCLK2	$f_{SSCGPS}/4$	3	4.50
			SPCLK5	$f_{SSCGPS}/32$	0.375	0.56

8.3 DMAC Registers

8.3.1 DMA Source address registers

These registers are used to set the DMA source addresses (28 bits each) for DMA channel n. They are divided into two 16-bit registers, DSAHn and DSALn.

Since these registers are configured as 2-stage FIFO buffer registers, a new source address for DMA transfer can be specified during DMA transfer (refer to *“Automatic Restart Function” on page 323*).

Caution DMA transfers of misaligned 16-bit/32-bit data is not supported.

(1) DSAHn - DMA source address registers Hn

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAH0	IR	0	0	0	SA26	SA26	SA25	SA24	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	FFFFF082H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAH1	IR	0	0	0	SA26	SA26	SA25	SA24	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	FFFFF08AH	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAH2	IR	0	0	0	SA26	SA26	SA25	SA24	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	FFFFF092H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAH3	IR	0	0	0	SA26	SA26	SA25	SA24	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	FFFFF09AH	undef.

Bit position	Bit name	Function
15	IR	Specifies the DMA source address. 0: External memory or On-chip peripheral I/O 1: Internal RAM
11 to 0	SA27 to SA16	Sets the DMA source addresses (A27 to A16). During DMA transfer, it stores the next DMA transfer source address.

(2) DSALn - DMA source address registers Ln

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAL0	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	FFFFF080H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAL1	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	FFFFF088H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAL2	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	FFFFF090H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAL3	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	FFFFF098H	undef.

Bit position	Bit name	Function
15 to 0	SA15 to SA0	Sets the DMA source address (A15 to A0). During DMA transfer, it stores the next DMA transfer source address.

8.3.2 DMA destination address registers

These registers are used to set the DMA destination address (28 bits each) for DMA channel n. They are divided into two 16-bit registers, DDAHn and DDALn.

Since these registers are configured as 2-stage FIFO buffer registers, a new destination address for DMA transfer can be specified during DMA transfer (refer to *“Automatic Restart Function” on page 323*).

Caution DMA transfers of misaligned 16-bit/32-bit data is not supported.

(1) DDAHn - DMA destination address registers Hn

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAH0	IR	0	0	0	DA27	DA26	DA25	DA24	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16	FFFFF086H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAH1	IR	0	0	0	DA27	DA26	DA25	DA24	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16	FFFFF08EH	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAH2	IR	0	0	0	DA27	DA26	DA25	DA24	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16	FFFFF096H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAH3	IR	0	0	0	DA27	DA26	DA25	DA24	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16	FFFFF09EH	undef.

Bit position	Bit name	Function
15	IR	Specifies the DMA destination address. 0: External memory or On-chip peripheral I/O 1: Internal RAM
11 to 0	DA27 to DA16	Sets the DMA destination addresses (A27 to A16). During DMA transfer, it stores the next DMA transfer destination address.

(2) DDALn - DMA destination address registers Ln

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAL0	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	FFFFF084H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAL1	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	FFFFF08CH	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAL2	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	FFFFF094H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAL3	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	FFFFF09CH	undef.

Bit position	Bit name	Function
15 to 0	DA15 to DA0	Sets the DMA destination address (A15 to A0). During DMA transfer, it stores the next DMA transfer destination address.

8.3.3 DBCn - DMA transfer count registers

These 16-bit registers are used to set the transfer counts for DMA channels n. They store the remaining transfer counts during DMA transfer.

Since these registers are configured as 2-stage FIFO buffer registers, a new DMA transfer count for DMA transfer can be specified during DMA transfer (refer to *“Automatic Restart Function” on page 323*).

During DMA transfer these registers are decremented by 1 for each transfer that is performed. DMA transfer is terminated when an underflow occurs (from 0 to FFFFH). On terminal count these registers are rewritten with the value that was set to the DBCn master register before.

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DBC0	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8	BC7	BC6	BC5	BC4	BC3	B2C	BC1	BC0	FFFFF0C0H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DBC1	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8	BC7	BC6	BC5	BC4	BC3	B2C	BC1	BC0	FFFFF0C2H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DBC2	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8	BC7	BC6	BC5	BC4	BC3	B2C	BC1	BC0	FFFFF0C4H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DBC3	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8	BC7	BC6	BC5	BC4	BC3	B2C	BC1	BC0	FFFFF0C6H	undef.

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>15 to 0</td><td>BC15 to BC0</td><td>Sets the transfer count. It stores the remaining transfer count during DMA transfer.
<table>
<tr><th>DBCn</th><th>States</th></tr>
<tr><td>0000H</td><td>Transfer count 1 or remaining transfer count</td></tr>
<tr><td>0001H</td><td>Transfer count 2 or remaining transfer count</td></tr>
<tr><td>:</td><td>:</td></tr>
<tr><td>FFFFH</td><td>Transfer count 65,536 (2^{16}) or remaining transfer count</td></tr>
</table></td></tr>
</table>

8.3.4 DADCn - DMA addressing control registers

These 16-bit registers are used to control the DMA transfer modes for DMA channel n.

They can be read/written in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DADC0	DS1	DS0	0	0	0	0	0	0	SAD1	SAD0	DAD1	DAD0	TM1	TM0	0	0	FFFFF0D0H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DADC1	DS1	DS0	0	0	0	0	0	0	SAD1	SAD0	DAD1	DAD0	TM1	TM0	0	0	FFFFF0D2H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DADC2	DS1	DS0	0	0	0	0	0	0	SAD1	SAD0	DAD1	DAD0	TM1	TM0	0	0	FFFFF0D4H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DADC3	DS1	DS0	0	0	0	0	0	0	SAD1	SAD0	DAD1	DAD0	TM1	TM0	0	0	FFFFF0D6H	0000H

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>15, 14</td><td>DS1, DS0</td><td>Sets the transfer data size for DMA transfer.
<table>
<tr><th>DS1</th><th>DS0</th><th>Transfer data size</th></tr>
<tr><td>0</td><td>0</td><td>8 bits</td></tr>
<tr><td>0</td><td>1</td><td>16 bits</td></tr>
<tr><td>1</td><td>0</td><td>32 bits</td></tr>
<tr><td>1</td><td>1</td><td>Setting prohibited</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>SAD1, SAD0</td><td>Sets the count direction of the source address for DMA channel n.
<table>
<tr><th>SAD1</th><th>SAD0</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>DAD1, DAD0</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>

Bit position	Bit name	Function
3, 2	TM1, TM0	Sets the transfer mode during DMA transfer. <table><tr><th>TM1</th><th>TM0</th><th>Transfer mode</th></tr><tr><td>0</td><td>0</td><td>Single transfer mode</td></tr><tr><td>0</td><td>1</td><td>Setting prohibited</td></tr><tr><td>1</td><td>0</td><td>Setting prohibited</td></tr><tr><td>1</td><td>1</td><td>Block transfer mode</td></tr></table>

Caution These registers cannot be accessed during DMA operation.

8.3.5 DCHCn - DMA channel control registers

These 8-bit registers are used to control the DMA transfer operating mode for DMA channel n.

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.)

	7	6	5	4	3	2	1	0	Address	Initial value
DCHC0	TC0	0	0	0	MLE0	INIT0	STG0	EN0	FFFFF0E0H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
DCHC1	TC1	0	0	0	MLE1	INIT1	STG1	EN1	FFFFF0E2H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
DCHC2	TC2	0	0	0	MLE	INIT	STG	EN2	FFFFF0E4H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
DCHC3	TC3	0	0	0	MLE	INIT	STG	EN3	FFFFF0E6H	00H

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 had not ended. 1: DMA transfer had ended.
3	MLEn	When the Multi Link Enable bit MLE is set to 1 at terminal count output, the ENn bit is not cleared to 0 and the DMA transfer enable state is retained (refer to *“Automatic Restart Function” on page 323*). Moreover, the next DMA transfer request can be accepted even when the TCn bit is not read, that means it is not cleared. When this bit is cleared to 0 at terminal count output, the ENn bit is cleared to 0 and the DMA transfer disable state is entered. At the next DMA request, the setting of the ENn bit to 1 and the reading of the TCn bit are required.
2	INITn	When this bit is set to 1, DMA transfer is forcibly terminated.
1	STGn	If this bit is set to 1 in the DMA transfer enable state (TCn bit = 0, ENn bit = 1), 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 If MLEn=0, this bit is cleared to 0 when DMA transfer ends. If MLEn=1, this bit is not cleared and the next DMA transfer is automatically restarted (refer to *“Automatic Restart Function” on page 323*). This bit is also cleared to 0 when DMA transfer is forcibly terminated by means of setting the INITn bit to 1 or by NMI input.

8.3.6 DRST - DMA restart register

The ENn bit of this register and the ENn bit of the DCHCn register are linked to each other. This provides a fast way to check the status of all DMA channels.

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

	7	6	5	4	3	2	1	0	Address	Initial value
DRST	0	0	0	0	EN3	EN2	EN1	EN0	FFFFF0F2H	00H

Bit position	Bit name	Function
3 to 0	EN3 to EN0	Specifies whether DMA transfer through DMA channel n is to be enabled or disabled. This bit is cleared to 0 when DMA transfer is completed in accordance with the terminal count output. It is also cleared to 0 when DMA transfer is forcibly terminated by setting the INITn bit to 1 or by NMI input. 0: DMA transfer disabled 1: DMA transfer enabled

8.3.7 DTFRn - DMA trigger source select register

The 8-bit DMA trigger source selection registers are used to control the DMA transfer triggers for the individual DMA channels. These triggers initiate DMA transfer requests received from built-in peripheral hardware.

Interrupt signals are used as DMA transfer requests.

These registers support read/write in 8-bit units or bit-wise.

Addresses DTFR0: FFFF FE00H

DTFR1: FFFF FE02H

DTFR2: FFFF FE04H

DTFR3: FFFF FE06H

	7	6	5	4	3	2	1	0
DTFRn	DRQn	DOFLn	DMACTn	0[a]	0[a]	IFCn2	IFCn1	IFCn0
Reset value	0	0	0	0	0	0	0	0
R/W	R/W[Note]	R/W[Note]	R/W	R/W	R/W	R/W	R/W	R/W

a) The default value “0” of this bit must not be changed!

Note DRQn and DOFLn are set by hardware.
DRQn and DOFLn can be *reset* by software. Setting these bits by software is not possible. A “0” must be written to the respective bit location to reset these bits.

The bits DTFRn.IFCn[2:0] select the interrupts to be used as DMA trigger sources according to the following table

		n	0	1	2	3
IFCn2	**IFCn1**	**IFCn0**	**Channel 0**	**Channel 1**	**Channel 2**	**Channel 3**
0	0	0	INTCB1R	INTCB2R[a]	INTCB1R	INTCB2R[a]
0	0	1	INTCB1T	INTCB2T[a]	INTCB1T	INTCB2T[a]
0	1	0	INTCB0R	INTCB0T	INTCB0R	INTCB0T
0	1	1	INTUA0R	INTUA0R	INTUA0T	INTUA0T
1	0	0	INTUA1R	INTUA1R	INTUA1T	INTUA1T
1	0	1	INTTZ0UV	INTTZ0UV	INTTZ1UV	INTTZ2UV
1	1	0	INTIIC0	INTLCD[a]	INTIIC1	INTLCD[a]
1	1	1	INTTP0CC1	INTTP1CC1	INTTG0CC1	INTAD

a) µPD70F3425, µPD70F3424 only

Caution If the DMA trigger source is changed by modifying DTFRn.IFCn[2:0] bits while DMA channel n is active, a DMA request may be set accidentally.

Proceed in any of the two ways when changing the DMA trigger source:

1. Disable the DMA channel n by DCHCn.ENn = 0 before changing the DMA trigger source DTFRn.IFCn[2:0].

2. Set the DMA request bit DTFRn.DRQn = 0 in parallel to changing DTFRn.IFCn[2:0], i.e. within the same write operation. Thus DTFRn must be written in 8-bit access mode. Do not change DTFRn.IFCn[2:0] with single-bit instructions.

The following list details the functions of the individual DMA trigger sources referenced in the above table.

- INTCB2R...INTCB0R
  The receive interrupts of the Clocked Serial Interfaces CSIB2…CSIB0 are used as DMA trigger sources. In case of a receive overflow condition no DMA trigger will be issued. The receive error interrupt of the respective CSIB INTCBnRE should be enabled to inform the application software about the overflow condition.
- INTCB2T…INTCB0T
  The transmit interrupts of the Clocked Serial Interfaces CSIB2…CSIB0 are used as DMA trigger sources.
- INTUA1R, INTUA0R
  The receive interrupts of the Asynchronous Serial Interfaces UARTA1 or UARTA0 are used as DMA trigger sources.

  In case of a receive overflow, or a framing or parity error condition, no DMA trigger will be issued. The receive error interrupt INTUAnRE of the respective UARTn should be enabled to inform the application software about the error condition. These interrupts are also generated upon reception of an SBF in LIN mode.
- INTUA1T, INTUA0T
  The transmit interrupts of the Asynchronous Serial Interfaces UARTA1 or UARTA0 are used as DMA trigger sources.
- INTLCD
  The interrupt signal of the LCD Bus Interface macro is used to trigger the DMA transfer.
- INTIIC0, INTIIC1
  The interrupts of the I^{2}C Interfaces IIC0, IIC1 are used to trigger the respective DMA channel.

DRQn	DMA request
0	No DMA transfer request is pending for channel n
1	DMA transfer request is pending for channel n

DOFLn	DMA request overflow
0	DMA transfer request overflow did not occur for channel n
1	DMA transfer request overflow occurred for channel n

DMACTn	DMA active count
0	DMACTn=0 must be set if internal RAM is not specified as source or destination
1	DMACTn=1 must be set if internal RAM is specified as source or destination

Set DMACTn according to the following table:

Source \ Destination	Internal RAM	Peripherals
Internal RAM	–	1
Peripherals	1	0

8.4 Automatic Restart Function

The DMA source address registers (DSAHn, DSALn), DMA destination address registers (DDAHn, DDALn), and DMA transfer count register (DBCn) are buffer registers with a 2-stage FIFO structure, named master and slave register.

The setup data of the slave registers is always used for the current DMA transfer, while the master registers may hold a new setup to be used automatically after the first DMA transfer has completed.

When the terminal count DCHCn.TCn=1 is issued, the slave registers are automatically rewritten with the values of the master registers.

Therefore, during DMA transfer, transfer is automatically started when a new DMA transfer setting is made for these registers and the MLEn bit of the DCHCn register is set (however, the DMA transfer end interrupt is issued even if DMA transfer is automatically started).

This mode is called multi link mode and is configured by DCHCn.MLEn=1.

If DMA channel n is disabled (DCHCn.ENn=0), writing to DSAH/Ln, DDAH/Ln, DBCn stores the data to the master and slave registers.

Writing the next DMA transfer setup data to the master registers only - and to keep the first setup data in the slave registers - is possible after

- the DMA channel n has been enabled (DCHCn.ENn=1) *and*
- the first DMA trigger interrupt for channel n has occurred.

The new setup data will become effective after

- the previous DMA transfer has completed (DCHC.TCn=1, INTDMAn) *and*
- the next following DMA trigger interrupt for channel n has occurred.

Note that the terminal count flag DCHC.TCn does not need to be cleared in multi link mode (DCHC.MLEn = 1) for starting up the next DMA transfer automatically.

Figure 8-1 shows the configuration of the buffer register.

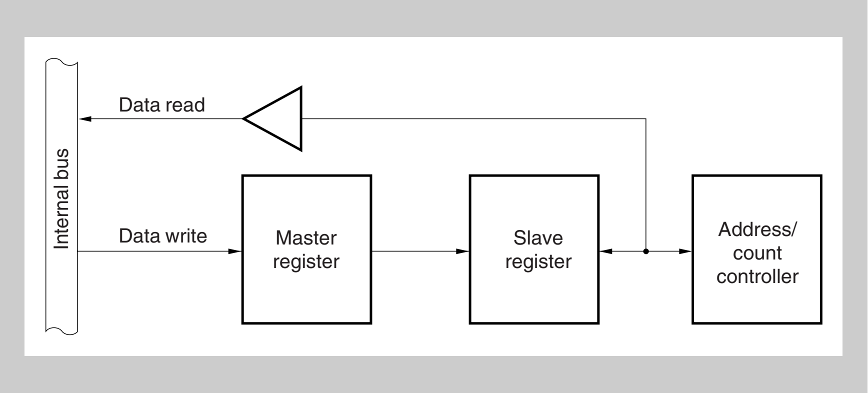


Figure 8-1 Buffer register configuration

8.5 Transfer Type

All DMA transfers of this microcontroller are two-cycle transfers.

In two-cycle transfer, data transfer is performed in two cycles: a read cycle (source to DMAC) and a write cycle (DMAC to destination).

In the first cycle, the source address is output and reading is performed from the source to the DMAC. In the second cycle, the transfer destination address is output and writing is performed from the DMAC to the transfer destination.

8.6 Transfer Object

The following transfer objects can be specified as source and destination:

Table 8-2 Transfer objects

Source \ Destination	Internal RAM	Peripherals
Internal RAM	–	√
Peripherals	√	√

8.7 DMA Channel Priorities

The DMA channel priorities are fixed as follows.

DMA channel 0 > DMA channel 1 > … > DMA channel n

In the single-step transfer mode, the DMA Controller releases the buses after each byte/half-word/word transfer. If a higher priority DMA transfer request is issued while the bus is released, the higher priority DMA transfer request is acknowledged.

In the block transfer mode, the channel used for transfer is never switched.

8.8 DMA Transfer Start Factors

There are two types of DMA transfer start factors, as shown below.

(1) Request from on-chip peripheral I/O

If the ENn and the TCn bits of the DCHCn register are set as shown below, and an interrupt request is issued from the on-chip peripheral I/O that is set in the DTFRn register, the DMA transfer starts.

- ENn bit = 1
- TCn bit = 0

(2) Request from software

If the STGn, the ENn and the TCn bits of the DCHCn register are set as follows, the DMA transfer starts.

- STGn bit = 1
- ENn bit = 1
- TCn bit = 0

8.9 Forcible Interruption

DMA transfer can be forcibly interrupted by NMI input during DMA transfer. At such a time, the DMAC clears the ENn bit of the DCHCn register of all channels and the DMA transfer disabled state is entered. An NMI request can then be acknowledged after the DMA transfer executed during NMI input is terminated.

In block transfer mode, the DMA transfer request is held in the DMAC. If the ENn bit is set back to "1", the DMA transfer is resumed from the point where it was interrupted.

In the single transfer mode, if the ENn bit is set back to "1", the next DMA transfer request is acknowledged and DMA transfer is resumed.

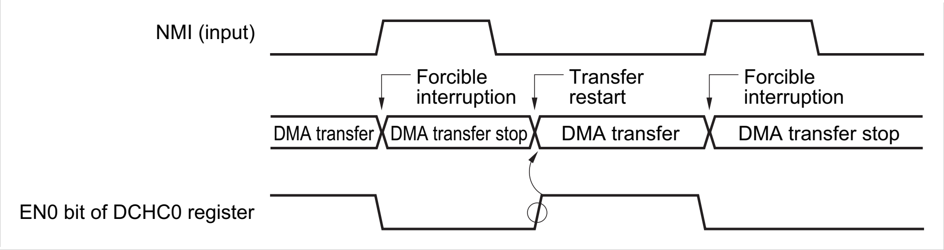


Figure 8-2 Example of forcible interruption of DMA transfer

Caution The resumed DMA transfer after NMI interruption cannot be executed with new settings. New settings for a DMA transfer can be validated either after the end of the current transfer or after the transfer has been forcibly terminated by setting the INITn bit of the DCHCn register.

8.10 Forcible Termination

In addition to the forcible interruption operation by means of the NMI input, DMA transfer can be forcibly terminated by the INITn bit of the DCHCn register. The following is an example of the operation of a forcible termination.

Figure 8-3 shows a block transfer of channel 3 which begins during the DMA block transfer of DMA channel 2. The block transfer of DMA channel 2 is forcibly terminated by setting the INIT2 bit of its DCHC2 control register.

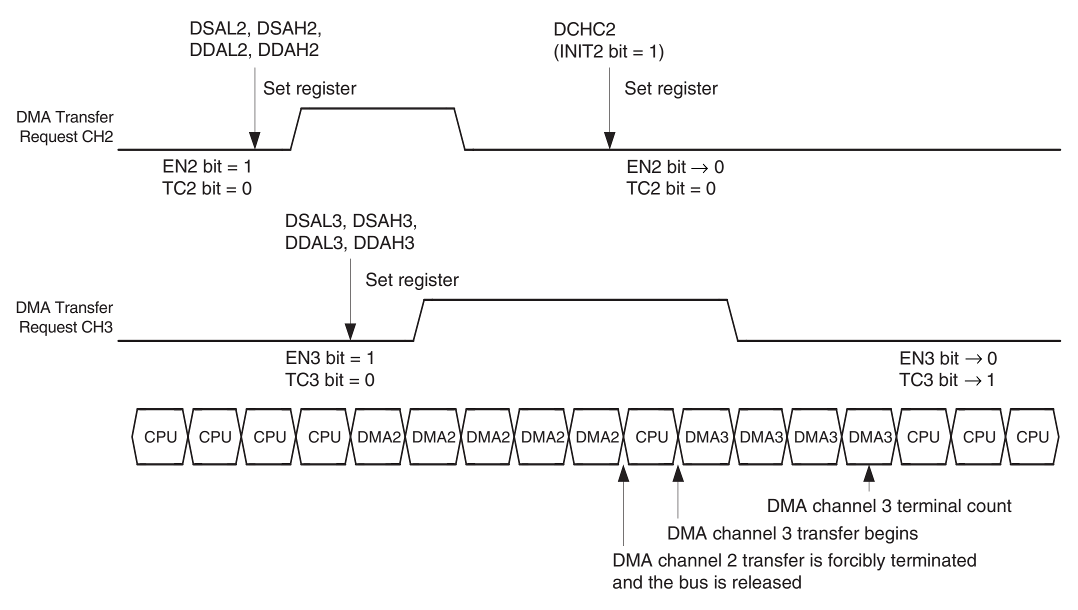


Figure 8-3 DMA transfer forcible termination example 1

Note The next condition can be set even during DMA transfer because the DSAn, DDAn, and DBCn registers are buffered registers. However, the setting to the DADCn register is invalid (refer to *“Automatic Restart Function” on page 323* and *“DADCn - DMA addressing control registers” on page 317*).

Figure 8-4 shows a forcible termination of a block transfer operation of DMA channel 1. A transfer containing a new configuration is executed.

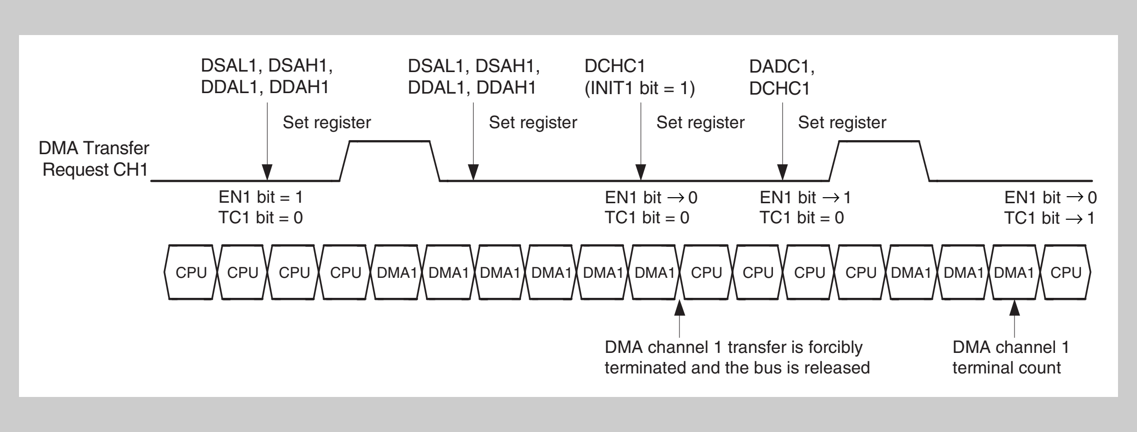


Figure 8-4 **DMA transfer forcible termination example 2**

Note Since the DSALn, DSAHn, DDALn, DDAHn and DBCn registers are buffered registers, the next transfer condition can be set even during a DMA transfer. However, a setting in the DADCn register is ignored (refer to *“Automatic Restart Function” on page 323*)

8.11 DMA Transfer Completion

When DMA transfer ends and the TCn bit of the DCHCn register is set, a DMA transfer end interrupt (INTDMAn) is issued to the Interrupt Controller (INTC).

8.12 Transfer Mode

8.12.1 Single transfer mode

In single transfer mode, the DMAC releases the bus after each byte/halfword/word 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. However, if a lower priority DMA transfer request is generated within one clock after the end of a single transfer, even if the previous higher priority DMA transfer request signal stays active, this request is not prioritized and the next DMA transfer after the bus is released for the CPU is a transfer based on the newly generated, lower priority DMA transfer request.

Figure 8-5 shows a DMAC transfer in single transfer mode. In this example the DMA channel 3 is used for a single transfer.

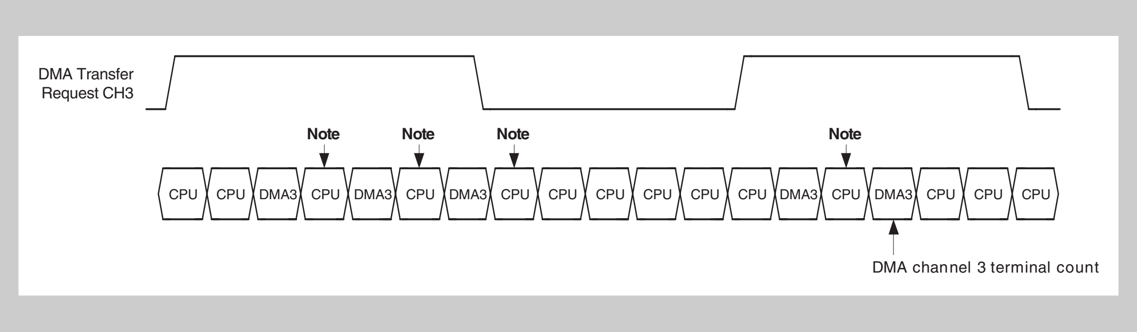


Figure 8-5 Single transfer example 1

Note The bus is always released

Figure 8-6 shows DMAC transfers in single transfer mode in which a higher priority DMA transfer request is generated. DMA channels 0 to 2 are used for a block transfer and channel 3 is used for a single transfer.

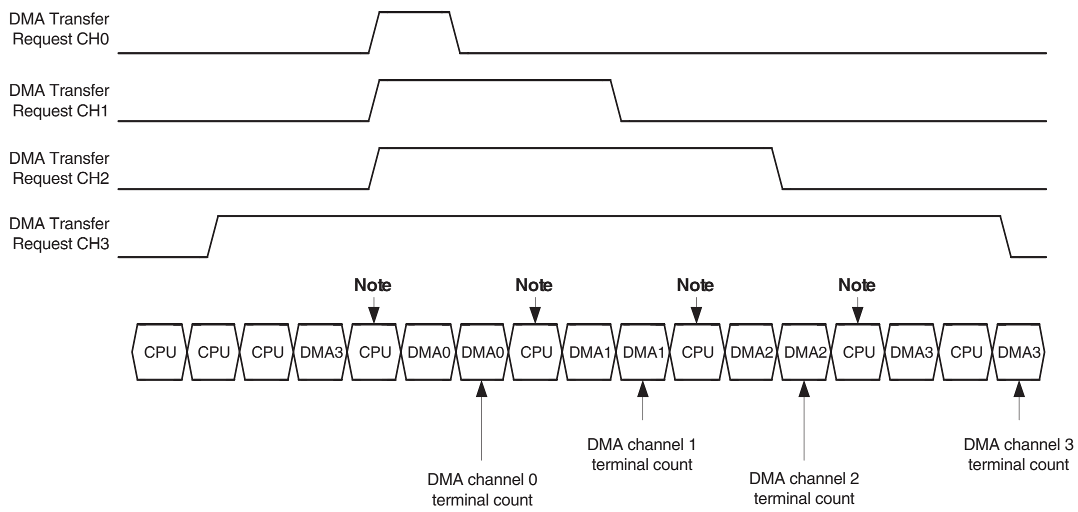


Figure 8-6 **Single transfer example 2**

Note The bus is always released

Figure 8-7 shows a DMA transfer example in single transfer mode in which a lower priority DMA transfer request is generated within one clock after the end of a single transfer. DMA channels 0 and 3 are used for the single transfer example. When two DMA transfer request signals are activated at the same time, the two DMA transfers are performed alternately.

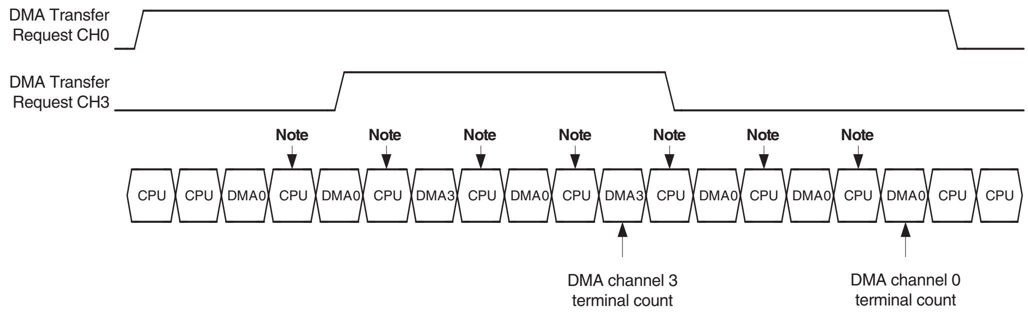


Figure 8-7 **Single transfer example 3**

Note The bus is always released

Figure 8-8 shows a single transfer mode example in which two or more lower priority DMA transfer requests are generated within one clock after the end of a single transfer. DMA channels 0, 2 and 3 are used for this single transfer example. When three or more DMA transfer request signals are activated at the same time always the two highest priority DMA transfers are performed alternately.

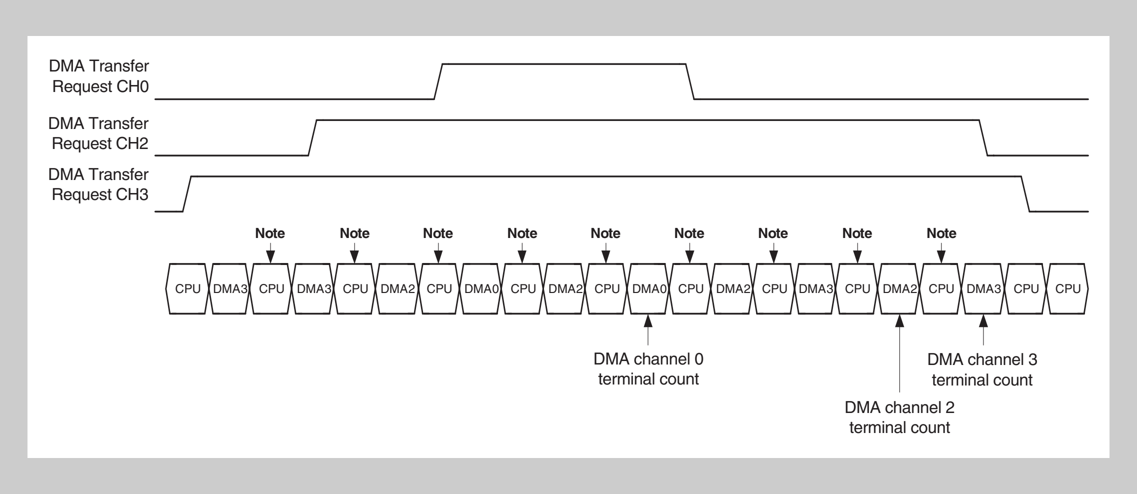


Figure 8-8 Single transfer example 4

Note The bus is always released

8.12.2 Block transfer mode

In the block transfer mode, once transfer begins, the DMAC continues the transfer operation without releasing the bus until a terminal count occurs. No other DMA requests are acknowledged during block transfer.

After the block transfer ends and the DMAC releases the bus and another DMA transfer can be acknowledged.

Figure 8-9 shows a block transfer mode example. It is a block transfer mode example in which a higher priority DMA transfer request is generated. DMA channels 2 and 3 are used for the block transfer example.

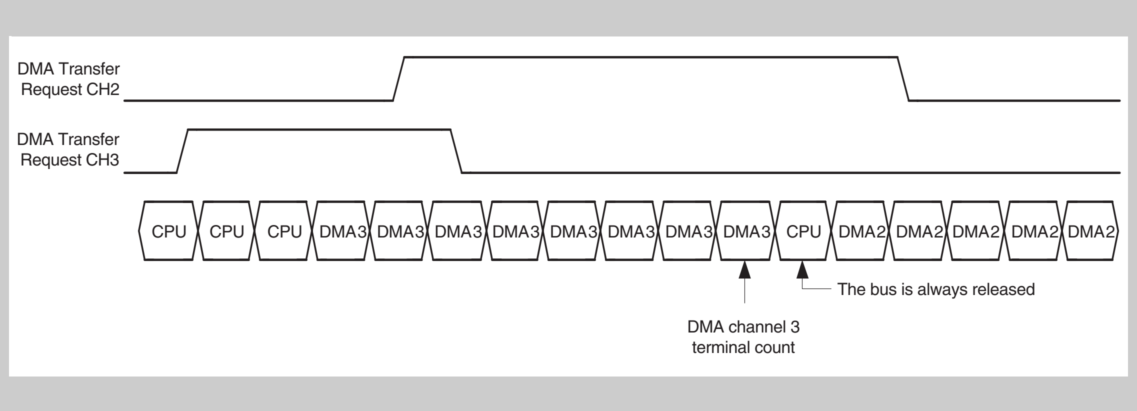


Figure 8-9 Block transfer example

Chapter 9 ROM Correction Function (ROMC)

This microcontroller features following ROM correction facilities:

- “DBTRAP” ROM correction:
  - 1x 8 channels for VFB flash memory and ROM
  - 1 x 8 channels for VSB flash memory (for µPD70F3426 only)

  The individual channels of each “DBTRAP” ROM correction are identified by “m” (m = 0 to 7)

Caution During self-programming make sure to disable all ROM correction facilities, as enabled ROM corrections may conflict with the internal firmware.

9.1 Overview

The ROM Correction Function is used to replace part of the internal ROM or flash memory with user defined data.

By using this function, program bugs found in the internal ROM and flash memory can be corrected.

The “DBTRAP” ROM correction unit substitutes an instruction fetched from ROM or flash memory by the DBTRAP instruction. Thus a DBTRAP exception is excited and program execution branches to the DBTRAP vector 0000 0060_H.

Note that the “DBTRAP” ROM correction unit is utilized by the N-Wire on-chip debug unit. Therefore ROM corrections by DBTRAP are not available, when N-Wire on-chip debugging is performed.

9.2 “DBTRAP” ROM Correction Unit

- 1x 8 channels for VFB flash memory and ROM
- The individual channels of the “DBTRAP” ROM correction unit are identified by “m” (m = 0 to 7)
- Programmable correction address for each channel
- “DBTRAP” exception processing upon correction address match
- Enable/Disable of each channel individually by software

Caution The “DBTRAP” ROM correction unit is also used by the N-Wire on-chip debug unit. Thus ROM correction will not be performed on these correction channels when the microcontroller is operating in N-Wire debug mode.

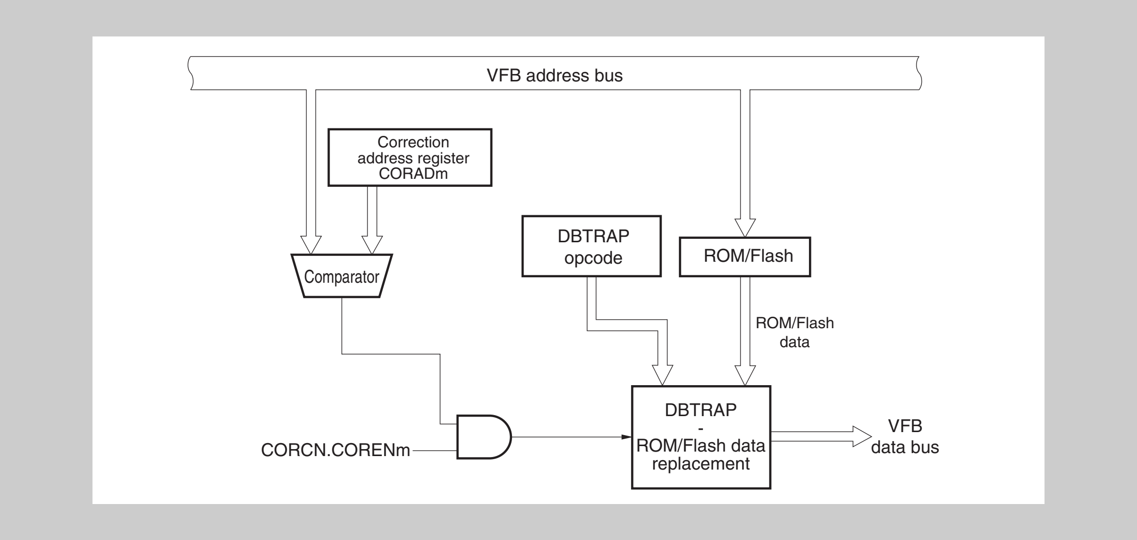


Figure 9-1 **“DBTRAP” ROM correction block diagram**

9.2.1 “DBTRAP” ROM correction operation

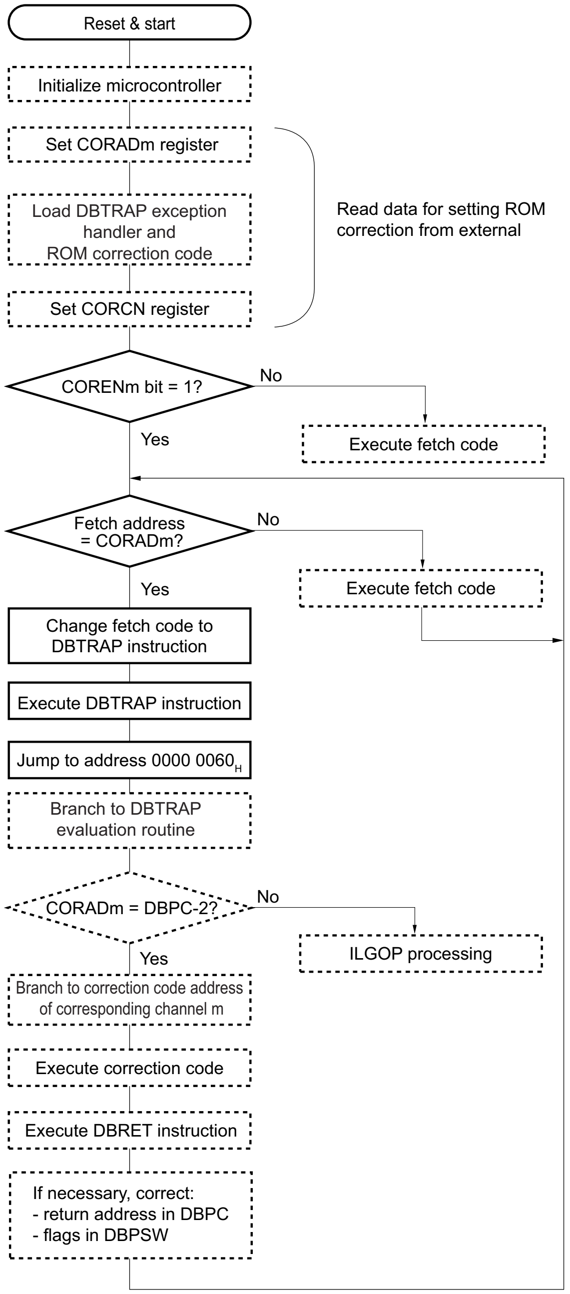
The “DBTRAP” ROM correction unit compares the address on the V850 fetch bus (VFB) with the contents of the programmable correction address registers CORADm. If an address matches, the DBTRAP instruction opcode is put on the V850 fetch bus instead of the ROM contents. If no address matches, the ROM contents is passed on the fetch bus as normal.

The DBTRAP exception branches to the DBTRAP/ILGOP exception handler address 0000 0060_H, which comprises the user’s ROM correction instructions.

Since the ROM correction routines for all correction channels are invoked at the DBTRAP exception handler address 0000 0060_H, the exception handler has to evaluate first the right correction routine to be executed. This is done by reading the DBPC register, which holds the address next to the correction address of CORADm, which has caused the DBTRAP exception. If non of CORADm matches DBPC - 2, DBTRAP was generated by an illegal opcode detection event ILGOP. For further details concerning DBTRAP/ILGOP handling refer to *“Exception Trap“ on page 222.*

Figure 9-2 outlines a typical program flow for using the “DBTRAP” ROM correction.

1. If the address CORADm to be corrected and the fetch address of the internal ROM memory match, the instruction code fetched from ROM is replaced by the DBTRAP instruction.
2. When the DBTRAP instruction is executed, execution branches to address 0000 0060_H.
3. The DBTRAP evaluation routine identifies the cause of the DBTRAP exception and launches either the appropriate ROM correction routine or the ILGOP handler.
4. In case several consecutive ROM instruction are replaced by ROM correction code the return address in DBPC must be corrected. It may also be required to correct some flags in the DBPSW register.
5. Return processing is started by the DBRET instruction.



Notes: ⬚ : Processing by user program (software)

▭ : Processing by ROM correction (hardware)

Figure 9-2 ROM correction operation and program flow

9.2.2 “DBTRAP” ROM correction registers

(1) CORCN - VFB flash/ROM “DBTRAP” ROM correction control register

This register enables or disables the VFB flash/ROM ROM correction of each channel.

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

Address FFFF F880$_H$

Initial Value 0000$_H$

7	6	5	4	3	2	1	0
COREN7	COREN6	COREN5	COREN4	COREN3	COREN2	COREN1	COREN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 9-1 CORCN register contents

Bit Position	Bit Name	Function
7 to 0	CORENm	ROM correction channel 0: ROM correction for channel m disabled 1: ROM correction for channel m enabled

Note ROM correction of channel n should only be enabled after the correction address CORADm has been set.

(2) COR2CN - VSB flash “DBTRAP” ROM correction control register (µPD70F3427 only)

This register enables or disables VSB flash memory ROM correction of each channel.

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

Address FFFF F9D0$_H$

Initial Value 0000$_H$

7	6	5	4	3	2	1	0
COR2EN7	COR2EN6	COR2EN5	COR2EN4	COR2EN3	COR2EN2	COR2EN1	COR2EN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 9-2 COR2CN register contents

Bit Position	Bit Name	Function
7 to 0	COR2ENm	ROM correction channel 0: ROM correction for channel m disabled 1: ROM correction for channel m enabled

Note ROM correction of channel n should only be enabled after the correction address COR2ADm has been set.

(3) CORADm - VFB flash/ROM “DBTRAP” ROM Correction address register

These registers hold the address where the VFB flash/ROM correction should be performed.

Access These registers can be read/written in 32-bit (CORADm) and 16-bit units (CORADmL for bits 15 to 0, CORADmH for bits 31 to 16).

Address

CORAD0, CORAD0L:	FFFF F840$_H$	CORAD0H:	FFFF F842$_H$
CORAD1, CORAD1L:	FFFF F844$_H$	CORAD1H:	FFFF F846$_H$
CORAD2, CORAD2L:	FFFF F848$_H$	CORAD2H:	FFFF F84A$_H$
CORAD3, CORAD3L:	FFFF F84C$_H$	CORAD3H:	FFFF F84E$_H$
CORAD4, CORAD4L:	FFFF F850$_H$	CORAD4H:	FFFF F852$_H$
CORAD5, CORAD5L:	FFFF F854$_H$	CORAD5H:	FFFF F856$_H$
CORAD6, CORAD6L:	FFFF F858$_H$	CORAD6H:	FFFF F85A$_H$
CORAD7, CORAD7L:	FFFF F85C$_H$	CORAD7H:	FFFF F85E$_H$

Initial Value 0000 0000$_H$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CORADm[15:0]															0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	CORADm[19:16]			

R/W

Table 9-3 CORADm register contents

Bit Position	Bit Name	Function
19 to 0	CORADm [19:0]	Lower 16 bit of the ROM correction address of channel m. Bit 0 and bits 31 to 20 are fixed to 0, writing to these bits is ignored.

Caution The ROM correction address CORADm[19:0] must not exceed the upper address of the internal ROM respectively flash memory.
If the internal ROM/flash memory size is less than 1 MB the appropriate number of upper address bits of CORADm[19:0] must be set to 0.

Note CORADm shall only be changed when the corresponding channel is disabled (CORCN.CORENm = 0).

(4) COR2ADm - VSB flash “DBTRAP” ROM Correction address register (µPD70F3427 only)

These registers hold the address where the VSB flash memory correction should be performed.

Access These registers can be read/written in 32-bit (COR2ADm) and 16-bit units (COR2ADmL for bits 15 to 0, COR2ADmH for bits 31 to 16).

Address

COR2AD0, COR2AD0L:	FFFF F8A0$_H$	COR2AD0H:	FFFF F8A2$_H$
COR2AD1, COR2AD1L:	FFFF F8A4$_H$	COR2AD1H:	FFFF F8A6$_H$
COR2AD2, COR2AD2L:	FFFF F8A8$_H$	COR2AD2H:	FFFF F8AA$_H$
COR2AD3, COR2AD3L:	FFFF F8AC$_H$	COR2AD3H:	FFFF F8AE$_H$
COR2AD4, COR2AD4L:	FFFF F8B0$_H$	COR2AD4H:	FFFF F8B2$_H$
COR2AD5, COR2AD5L:	FFFF F8B4$_H$	COR2AD5H:	FFFF F8B6$_H$
COR2AD6, COR2AD6L:	FFFF F8B8$_H$	COR2AD6H:	FFFF F8BA$_H$
COR2AD7, COR2AD7L:	FFFF F8BC$_H$	COR2AD7H:	FFFF F8BE$_H$

Initial Value 0000 0000$_H$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COR2ADm[15:0]															0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	COR2ADm[19:16]			
R/W															

Table 9-4 COR2ADm register contents

Bit Position	Bit Name	Function
19 to 0	COR2ADm [19:0]	Lower 16 bit of the ROM correction address of channel m. Bit 0 and bits 31 to 20 are fixed to 0, writing to these bits is ignored.

Caution The ROM correction address COR2ADm[19:0] must not exceed the upper address of the internal VSB flash memory.

Note COR2ADm shall only be changed when the corresponding channel is disabled (COR2CN.COR2ENm = 0).

Chapter 10 Code Protection and Security

10.1 Overview

The microcontroller supports various methods for protecting the program code in the flash memory from undesired access, such as illegal read-out or illegal reprogramming.

Some interfaces offer in general access to the internal flash memory: N-Wire debug interface, external flash programmer interface, self-programming facilities and test interfaces.

In the following the security relevant items are listed. The features to protect the internal flash memory data from being read by unauthorized persons are described.

For more information on the flash memory, see *“Flash Memory“ on page 229*.

The following sections give an overview about supported code protection methods.

10.2 Boot ROM

Undesired access to the flash memory via the boot ROM is not possible.

10.3 N-Wire Debug Interface

In general, illegal read-out of the flash memory contents is possible via the N-Wire debug interface. 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 10-1*

Table 10-1 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 N-Wire debug interface can only be used if the user enters the correct ID code.
	ID code is all ones[c]	Level 0: No protection N-Wire debug interface can be used.

a) Codes are not compared

b) Once the N-Wire debug interface has been set as “use-prohibited”, it cannot be used until the flash memory is re-programmed.

c) This is the default state after the flash memory has been erased.

Note

1. After you have set protection levels 1 or 2, set the “block erase disable flag” in the flash extra area. Otherwise, an unauthorized person could erase the block that contains the ID code or the “N-Wire use enable flag”, respectively, and thus suspend the protection.
2. If an unauthorized user tries to find out the 10-byte ID by comparing all possible ID codes, this will take up to 3.83 x 10^8 years at 100 MHz.

For more details refer to *“Security function“ on page 879*.

10.4 Flash Writer and Self-Programming Protection

In general, illegal read-out and re-programming of the flash memory contents is possible via the flash writer interface and the self-programming feature. For protection of the flash memory, the following flags provide various protection levels.

The flags can be set by flash programmers. For a description of flash memory programming see *“Flash Memory“ on page 229*.

(1) Program protection flag (Program protection function)

Set this flag to disable the programming function via flash writer interface. This flag does not affect the self-programming interface.

The flag is valid for the whole flash memory.

(2) Chip erase protection flag (Chip erase protection function)

Set this flag to disable the chip erase function via flash writer interface. This flag does not affect the self-programming interface.

(3) Block erase protection flag (Block erase protection function)

Set this flag to disable the feature to erase single blocks via flash writer interface. This flag does not affect the self-programming interface.

This flag does not affect the chip erase function.

The flag is valid for the whole flash memory.

(4) Read-out protection flag (Read-out protection function)

Set this flag to disable the feature that allows reading back the flash memory via flash writer interface. This flag does not affect the self-programming interface.

This flag is valid for the whole flash memory.

10.4.1 Variable reset vector

The reset vector, determining the start of the user’s program is stored in an “extra area” of the flash memory. This vector is configurable via an external flash programmer and by self-programming.

Chapter 11 16-bit Timer/Event Counter P (TMP)

Timer P (TMP) is a 16-bit timer/event counter.

The V850E/Dx3 microcontrollers have following instances of the 16-bit timer/event counter TMP:

TMP	**All devices**
Instances	4
Names	TMP0 to TMP3

Throughout this chapter, the individual instances of Timer P are identified by “n”, for example TMPn, or TPnCTL0 for the TMPn control register 0.

11.1 Overview

An outline of TMPn is shown below.

- Clock selection: 8 ways
- Capture/trigger input pins: 2
- External event count input pins: 1
- External trigger input pins: 1
- Timer/counters: 1
- Capture/compare registers: 2
- Capture/compare match interrupt request signals: 2
- Timer output pins: 2

11.2 Functions

TMPn has the following functions.

- Interval timer
- External event counter
- External trigger pulse output
- One-shot pulse output
- PWM output
- Free-running timer
- Pulse width measurement
- TMP0 and TMP1 can be used for triggering the DMA Controller.
- All TMPn can be optionally stopped when a breakpoint is hit during debugging (refer to *“On-Chip Debug Unit“ on page 877*).

11.3 Configuration

TMPn includes the following hardware.

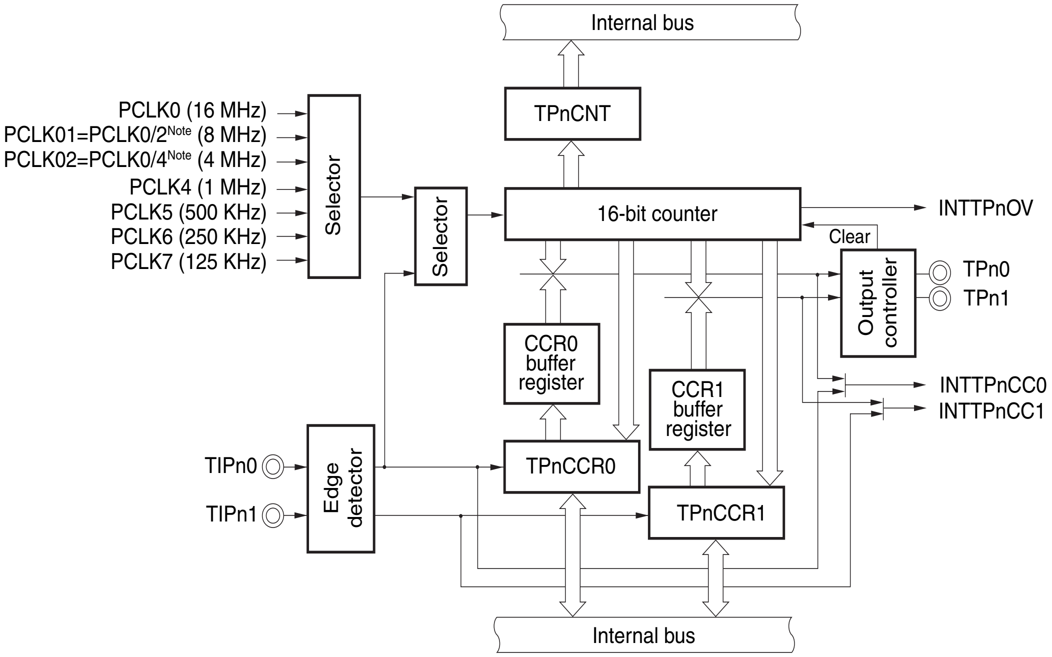


Figure 11-1 Block diagram of TMPn

The second (PCLK01) and the third (PCLK02) clock selector input is not supplied from the clock generator, but derived from the first selector input PCLK0 inside the timer P.
In case the PLL is disabled the PCLKx clocks are supplied from the main oscillator, i.e.:

- PCLK0 = 4 MHz
- PCLK01 = PCLK0/2 = 2 MHz
- PCLK02 = PCLK0/4 = 1 MHz

For information about PCLKx, please refer to *“Clock Generator“ on page 129*.

(1) 16-bit counter

This 16-bit counter can count internal clocks or external events.

The count value of this counter can be read by using the TPnCNT register.

When the TPnCTL0.TPnCE bit = 0, the value of the 16-bit counter is FFFFH. If the TPnCNT register is read at this time, 0000H is read.

Reset input clears the TPnCE bit to 0. Therefore, the 16-bit counter is set to FFFFH.

(2) CCR0 buffer register

This is a 16-bit compare register that compares the count value of the 16-bit counter.

When the TPnCCR0 register is used as a compare register, the value written to the TPnCCR0 register is transferred to the CCR0 buffer register. When the count value of the 16-bit counter matches the value of the CCR0 buffer register, a compare match interrupt request signal (INTTPnCC0) is generated.

The CCR0 buffer register cannot be read or written directly.

The CCR0 buffer register is cleared to 0000H after reset, as the TPnCCR0 register is cleared to 0000H.

(3) CCR1 buffer register

This is a 16-bit compare register that compares the count value of the 16-bit counter.

When the TPnCCR1 register is used as a compare register, the value written to the TPnCCR1 register is transferred to the CCR1 buffer register. When the count value of the 16-bit counter matches the value of the CCR1 buffer register, a compare match interrupt request signal (INTTPnCC1) is generated.

The CCR1 buffer register cannot be read or written directly.

The CCR1 buffer register is cleared to 0000H after reset, as the TPnCCR1 register is cleared to 0000H.

(4) Edge detector

This circuit detects the valid edges input to the TIPn0 and TIPn1 pins. No edge, rising edge, falling edge, or both the rising and falling edges can be selected as the valid edge by using the TPnIOC1 and TPnIOC2 registers.

(5) Output controller

This circuit controls the output of the TOPn0 and TOPn1 pins. The output controller is controlled by the TPnIOC0 register.

(6) Selector

This selector selects the count clock for the 16-bit counter. Eight types of internal clocks or an external event can be selected as the count clock.

11.4 TMP Registers

The TMPn are controlled and operated by means of the following registers:

Table 11-1 TMPn registers overview

Register name	Shortcut	Address
TMPn control registers 0	TPnCTL0	<base>
TMPn control registers 1	TPnCTL1	<base> + 1_H
TMPn I/O control register 0	TPnIOC0	<base> + 2_H
TMPn I/O control register 1	TPnIOC1	<base> + 3_H
TMPn I/O control register 2	TPnIOC2	<base> + 4_H
TMPn option registers 0	TPnOPT0	<base> + 5_H
TMPn capture/compare registers 0	TPnCCR0	<base> + 6_H
TMPn capture/compare registers 1	TPnCCR1	<base> + 8_H
TMPn counter read buffer register	TPnCNT	<base> + A_H

Table 11-2 CSIBn register base address

Timer	Base address
TMP0	FFFF $F660_H$
TMP1	FFFF $F670_H$
TMP2	FFFF $F680_H$
TMP3	FFFF $F690_H$

(1) TPnCTL0 - TMPn control register 0

The TPnCTL0 register is an 8-bit register that controls the operation of TMPn.

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

Address <base>

Initial Value 00_H. This register is initialized by any reset.
The same value can always be written to the TPnCTL0 register by software.

7	6	5	4	3	2	1	0
TPnCE	0	0	0	0	TPnCKS2	TPnCKS1	TPnCKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 11-3 TPnCTL0 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>TPnCE</td><td>TMPn operation disable/enable:
0: TMPn operation disabled (TMPn reset asynchroneously: reset of TPn0PT0.TPnOVF bit, 16-bit counter, timer output (TOPn0, TOPn1 pins)
1: TMPn operation enabled (TMPn operation starts)</td></tr>
<tr><td>2 to 0</td><td>TPnCKS[2:0]</td><td>Internal count clock selection:
<table>
<tr><th>TPnCKS2</th><th>TPnCKS1</th><th>TPnCKS0</th><th>Internal count clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>PCLK0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>PCLK01 = PCLK0/2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>PCLK02 = PCLK0/4</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Prohibited</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>PCLK4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>PCLK5</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>PCLK6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>PCLK7</td></tr>
</table></td></tr>
</table>

Caution

1. Set the TPnCKS2 to TPnCKS0 bits when the TPnCE bit = 0.
2. When the value of the TPnCE bit is changed from 0 to 1, the TPnCKS2 to TPnCKS0 bits can be set simultaneously.
3. Be sure to clear bits 3 to 6 to 0.

Note For information about PCLKx, please refer to *“Clock Generator“ on page 129*.

(2) TPnCTL1 - TMPn control register 1

The TPnCTL1 register is an 8-bit register that controls the operation of TMPn.

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

Address <base> + 1_H

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

7	6	5	4	3	2	1	0
0	TPnEST	TPnEEE	0	0	TPnMD2	TPnMD1	TPnMD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 11-4 TPnCTL1 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>6</td><td>TPnEST</td><td>Software trigger control.
0: –
1: Generate a valid signal for external trigger input.
• In one-shot pulse output mode:
A one-shot pulse is output with writing 1 to the TPnEST bit as the trigger.
• In external trigger pulse output mode:
A PWM waveform is output with writing 1 to the TPnEST bit as the trigger.</td></tr>
<tr><td>5</td><td>TPnEEE</td><td>Count clock selection:
0: Disable operation with external event count input.
(Perform counting with the count clock selected by the TPnCTL0.TPnCK0 to TPnCK2 bits.)
1: Enable operation with external event count input.
(Perform counting at the valid edge of the external event count input signal.)
The TPnEEE bit selects whether counting is performed with the internal count clock or the valid edge of the external event count input.</td></tr>
<tr><td>2 to 0</td><td>TPnMD[2:0]</td><td>Timer mode selection:
<table>
<tr><th>TPnMD2</th><th>TPnMD1</th><th>TPnMD0</th><th>Timer mode</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Interval timer</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>External event count</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>External trigger pulse output</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>One-shot pulse output</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>PWM output</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Free-runnning timer</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Pulse width measurement</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Setting prohibited</td></tr>
</table>
</td></tr>
</table>

Caution

1. The TPnEST bit is valid only in the external trigger pulse output mode or one-shot pulse output mode. In any other mode, writing 1 to this bit is ignored.
2. External event count input is selected in the external event count mode regardless of the value of the TPnEEE bit.

3. Set the TPnEEE and TPnMD2 to TPnMD0 bits when the TPnCTL0.TPnCE bit = 0. (The same value can be written when the TPnCE bit = 1.) The operation is not guaranteed when rewriting is performed with the TPnCE bit = 1. If rewriting was mistakenly performed, clear the TPnCE bit to 0 and then set the bits again.
4. Be sure to clear bits 3, 4, and 7 to 0.

(3) TPnIOC0 - TMPn I/O control register 0

The TPnIOC0 register is an 8-bit register that controls the timer output (TOPn0, TOPn1 pins).

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

Address <base> + 2_H

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

7	6	5	4	3	2	1	0
0	0	0	0	TPnOL1	TPnOE1	TPnOL0	TPnOE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 11-5 TPnIOC0 register contents

Bit position	Bit name	Function
3	TPnOL1	TOPn1 pin output level setting: 0: TOPn1 pin output inversion disabled 1: TOPn1 pin output inversion enabled
2	TPnOE1	TOPn1 pin output setting: 0: Timer output disable – when TPnOL1 = 0: low level is output from TOPn1 pin – when TPnOL1 = 1: high level is output from TOPn1 pin 1: Timer output enable (A square wave is output from TOPn1 pin.)
1	TPnOL0	TOPn0 pin output level setting: 0: TOPn0 pin output inversion disabled 1: TOPn0 pin output inversion enabled
0	TPnOE0	TOPn0 pin output setting: 0: Timer output disable – when TPnOL0 = 0: low level is output from TOPn0 pin – when TPnOL0 = 1: high level is output from TOPn0 pin 1: Timer output enable (A square wave is output from TOPn0 pin.)

Caution

1. Rewrite the TPnOL1, TPnOE1, TPnOL0, and TPnOE0 bits when the TPnCTL0.TPnCE bit = 0. (The same value can be written when the TPnCE bit = 1.) If rewriting was mistakenly performed, clear the TPnCE bit to 0 and then set the bits again.
2. Even if the TPnOLm bit is manipulated when the TPnCE and TPnOEm bits are 0, the TOPnm pin output level varies (m = 0, 1).

(4) TPnIOC1 - TMPn I/O control register 1

The TPnIOC1 register is an 8-bit register that controls the valid edge of the capture trigger input signals (TIPn0, TIPn1 pins).

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

Address <base> + 3_H

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

7	6	5	4	3	2	1	0
0	0	0	0	TPnIS3	TPnIS2	TPnIS1	TPnIS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 11-6 TPnIOC1 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3 to 2</td><td>TPnIS[3:2]</td><td>Capture trigger input signal (TIPn1 pin) valied edge setting:
<table>
<tr><th>TPnIS3</th><th>TPnIS2</th><th>Capture trigger valid edge of TIPn1</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection (capture operation invalid)</td></tr>
<tr><td>0</td><td>1</td><td>Detection of rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Detection of falling edge</td></tr>
<tr><td>1</td><td>1</td><td>Detection of both edges</td></tr>
</table></td></tr>
<tr><td>1 to 0</td><td>TPnIS[1:0]</td><td>Capture trigger input signal (TIPn0 pin) valied edge setting:
<table>
<tr><th>TPnIS1</th><th>TPnIS0</th><th>Capture trigger valid edge of TIPn0</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection (capture operation invalid)</td></tr>
<tr><td>0</td><td>1</td><td>Detection of rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Detection of falling edge</td></tr>
<tr><td>1</td><td>1</td><td>Detection of both edges</td></tr>
</table></td></tr>
</table>

Caution

1. Rewrite the TPnIS3 to TPnIS0 bits when the TPnCTL0.TPnCE bit = 0. (The same value can be written when the TPnCE bit = 1.) If rewriting was mistakenly performed, clear the TPnCE bit to 0 and then set the bits again.
2. The TPnIS3 to TPnIS0 bits are valid only in the free-running timer mode and the pulse width measurement mode. In all other modes, a capture operation is not possible.

(5) TPnIOC2 - TMPn I/O control register 2

The TPnIOC2 register is an 8-bit register that controls the valid edge of the external event count input signal (TIPn0 pin) and external trigger input signal (TIPn0 pin).

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

Address <base> + 4_H

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

7	6	5	4	3	2	1	0
0	0	0	0	TPnEES1	TPnEES0	TPnETS1	TPnETS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 11-7 TPnIOC2 register contents

Bit position	Bit name	Function
3 to 2	TPnEES[1:0]	External event count input signal (TIPn0 pin) valid edge setting: TPnEES1 / TPnEES0 / External event count valid edge of TIPn0 0 / 0 / No edge detection (external event invalid) 0 / 1 / Detection of rising edge 1 / 0 / Detection of falling edge 1 / 1 / Detection of both edges
1 to 0	TPnETS[1:0]	Capture trigger input signal (TIPn0 pin) valid edge setting: TPnETS1 / TPnETS0 / External trigger input valid edge of TIPn0 0 / 0 / No edge detection (external trigger invalid) 0 / 1 / Detection of rising edge 1 / 0 / Detection of falling edge 1 / 1 / Detection of both edges

Caution

1. Rewrite the TPnEES1, TPnEES0, TPnETS1, and TPnETS0 bits when the TPnCTL0.TPnCE bit = 0. (The same value can be written when the TPnCE bit = 1.) If rewriting was mistakenly performed, clear the TPnCE bit to 0 and then set the bits again.
2. The TPnEES1 and TPnEES0 bits are valid only when the TPnCTL1.TPnEEE bit = 1 or when the external event count mode (TPnCTL1.TPnMD2 to TPnCTL1.TPnMD0 bits = 001) has been set.

(6) TPnOPT0 - TMPn option register 0

The TPnOPT0 register is an 8-bit register used to set the capture/compare operation and detect an overflow.

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

Address <base> + 5_H

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

7	6	5	4	3	2	1	0
0	0	TPnCCS1	TPnCCS10	0	0	0	TPnOVF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 11-8 TPnOPT0 register contents

Bit position	Bit name	Function
5	TPnCCS1	TPnCCR1 register capture/compare selection: 0: compare register selected 1: capture register selected The TPnCCS1 bit setting is valid only in the free-running timer mode.
4	TPnCCS0	TPnCCR0 register capture/compare selection: 0: compare register selected 1: capture register selected The TPnCCS0 bit setting is valid only in the free-running timer mode.
0	TPnOVF	TMPn overflow detection flag: Set (1): Overflow occurred Reset (0): TPnOVF bit 0 written or TPnCTL0.TPnCE bit = 0 • The TPnOVF bit is reset when the 16-bit counter count value overflows from FFFFH to 0000H in the free-running timer mode or the pulse width measurement mode. • An interrupt request signal (INTTPnOV) is generated at the same time that the TPnOVF bit is set to 1. The INTTPnOV signal is not generated in modes other than the free-running timer mode and the pulse width measurement mode. • The TPnOVF bit is not cleared even when the TPnOVF bit or the TPnOPT0 register are read when the TPnOVF bit = 1. • The TPnOVF bit can be both read and written, but the TPnOVF bit cannot be set to 1 by software. Writing 1 has no influence on the operation of TMPn.

Caution

1. Rewrite the TPnCCS1 and TPnCCS0 bits when the TPnCE bit = 0. (The same value can be written when the TPnCE bit = 1.) If rewriting was mistakenly performed, clear the TPnCE bit to 0 and then set the bits again.
2. Be sure to clear bits 1 to 3, 6, and 7 to 0.

(7) TPnCCR0 - TMPn capture/compare register 0

The TPnCCR0 register can be used as a capture register or a compare register depending on the mode.

This register can be used as a capture register or a compare register only in the free-running timer mode, depending on the setting of the TPnOPT0.TPnCCS0 bit. In the pulse width measurement mode, the TPnCCR0 register can be used only as a capture register. In any other mode, this register can be used only as a compare register.

The TPnCCR0 register can be read or written during operation.

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

Address <base> + 6_H

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

<table>
<tr><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 colspan="16">CCR0 value</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

(a) Function as compare register

The TPnCCR0 register can be rewritten even when the TPnCTL0.TPnCE bit = 1.

The set value of the TPnCCR0 register is transferred to the CCR0 buffer register. When the value of the 16-bit counter matches the value of the CCR0 buffer register, a compare match interrupt request signal (INTTPnCC0) is generated. If TOPn0 pin output is enabled at this time, the output of the TOPn0 pin is inverted.

When the TPnCCR0 register is used as a cycle register in the interval timer mode, external event count mode, external trigger pulse output mode, one-shot pulse output mode, or PWM output mode, the value of the 16-bit counter is cleared (0000H) if its count value matches the value of the CCR0 buffer register.

(b) Function as capture register

When the TPnCCR0 register is used as a capture register in the free-running timer mode, the count value of the 16-bit counter is stored in the TPnCCR0 register if the valid edge of the capture trigger input pin (TIPn0 pin) is detected. In the pulse-width measurement mode, the count value of the 16-bit counter is stored in the TPnCCR0 register and the 16-bit counter is cleared (0000H) if the valid edge of the capture trigger input pin (TIPn0) is detected.

Even if the capture operation and reading the TPnCCR0 register conflict, the correct value of the TPnCCR0 register can be read.

The following table shows the functions of the capture/compare register in each mode, and how to write data to the compare register.

Table 11-9 Function of capture/compare register in each mode and how to write compare register

Operation mode	Capture/compare register	How to write compare register
Interval timer	Compare register	Anytime write
External event counter	Compare register	Anytime write
External trigger pulse output	Compare register	Batch write
One-shot pulse output	Compare register	Anytime write
PWM output	Compare register	Batch write
Free-running timer	Capture/compare register	Anytime write
Pulse width measurement	Capture register	-

(8) TPnCCR1 - TMPn capture/compare register 1

The TPnCCR1 register can be used as a capture register or a compare register depending on the mode.

This register can be used as a capture register or a compare register only in the free-running timer mode, depending on the setting of the TPnOPT0.TPnCCS1 bit. In the pulse width measurement mode, the TPnCCR1 register can be used only as a capture register. In any other mode, this register can be used only as a compare register.

The TPnCCR1 register can be read or written during operation.

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

Address <base> + 8_H

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

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CCR1 value</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

(a) Function as compare register

The TPnCCR1 register can be rewritten even when the TPnCTL0.TPnCE bit = 1.

The set value of the TPnCCR1 register is transferred to the CCR1 buffer register. When the value of the 16-bit counter matches the value of the CCR1 buffer register, a compare match interrupt request signal (INTTPnCC1) is generated. If TOPn1 pin output is enabled at this time, the output of the TOPn1 pin is inverted.

(b) Function as capture register

When the TPnCCR1 register is used as a capture register in the free-running timer mode, the count value of the 16-bit counter is stored in the TPnCCR1 register if the valid edge of the capture trigger input pin (TIPn1 pin) is detected. In the pulse-width measurement mode, the count value of the 16-bit counter is stored in the TPnCCR1 register and the 16-bit counter is cleared (0000H) if the valid edge of the capture trigger input pin (TIPn1) is detected.

Even if the capture operation and reading the TPnCCR1 register conflict, the correct value of the TPnCCR1 register can be read.

The following table shows the functions of the capture/compare register in each mode, and how to write data to the compare register.

Table 11-10 Function of capture/compare register in each mode and how to write compare register

Operationmode	Capture/compare register	How to write compare register
Interval timer	Compare register	Anytime write
External event counter	Compare register	Anytime write
External trigger pulse output	Compare register	Batch write
One-shot pulse output	Compare register	Anytime write
PWM output	Compare register	Batch write
Free-running timer	Capture/compare register	Anytime write
Pulse width measurement	Capture register	-

(9) TPnCNT - TMPn counter read buffer register

The TPnCNT register is a read buffer register that can read the count value of the 16-bit counter.

If this register is read when the TPnCTL0.TPnCE bit = 1, the count value of the 16-bit timer can be read.

The value of the TPnCNT register is cleared to 0000_H when the TPnCE bit = 0. If the TPnCNT register is read at this time, the value of the 16-bit counter ($FFFF_H$) is not read, but 0000_H is read.

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

Address <base> + A_H

Initial Value 0000_H. This register is initialized by any reset, as the TPnCE bit is cleared to 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPnCNT value															
R/W															

11.5 Operation

TMPn can perform the following operations.

Operation	TPnCTL1.TPnEST Bit (Software Trigger Bit)	TIPn0 Pin (Ext. Trigger Input)	Capture/ Compare Register Setting	Compare Register Write
Interval timer mode	Invalid	Invalid	Compare only	Anytime write
External event count mode[Note 1]	Invalid	Invalid	Compare only	Anytime write
External trigger pulse output mode[Note 2]	Valid	Valid	Compare only	Batch write
One-shot pulse output mode[Note 2]	Valid	Valid	Compare only	Anytime write
PWM output mode	Invalid	Invalid	Compare only	Batch write
Free-running timer mode	Invalid	Invalid	Switching enabled	Anytime write
Pulse width measurement mode[Note 2]	Invalid	Invalid	Capture only	Not applicable

Note 1. To use the external event count mode, specify that the valid edge of the TIPn0 pin capture trigger input is not detected (by clearing the TPnIOC1.TPnIS1 and TPnIOC1.TPnIS0 bits to “00”).

2. When using the external trigger pulse output mode, one-shot pulse output mode, and pulse width measurement mode, select the internal clock as the count clock (by clearing the TPnCTL1.TPnEEE bit to 0).

11.5.1 Interval timer mode (TPnMD2 to TPnMD0 = 000)

In the interval timer mode, an interrupt request signal (INTTPnCC0) is generated at the specified interval if the TPnCTL0.TPnCE bit is set to 1. A square wave whose half cycle is equal to the interval can be output from the TOPn0 pin.

Usually, the TPnCCR1 register is not used in the interval timer mode.

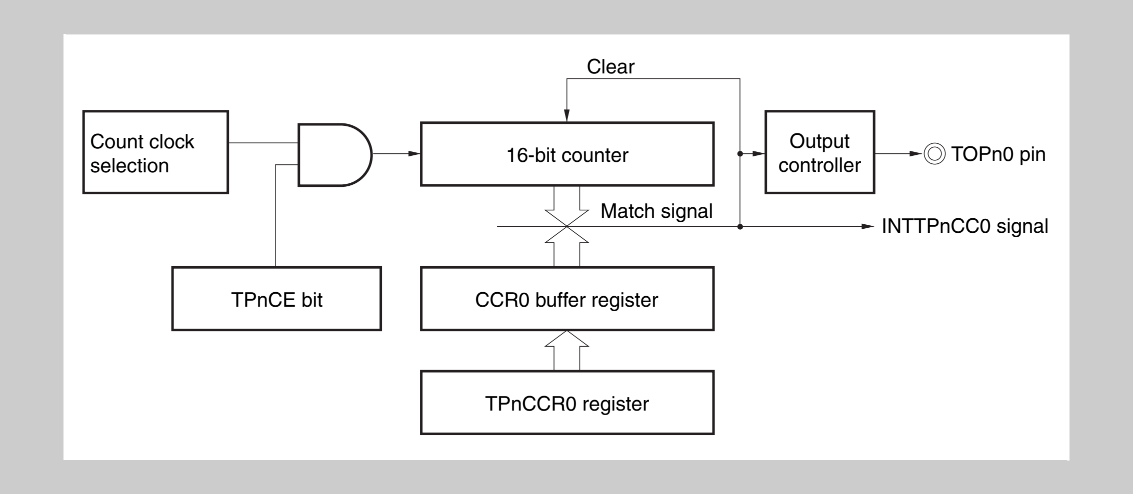


Figure 11-2 Configuration of interval timer

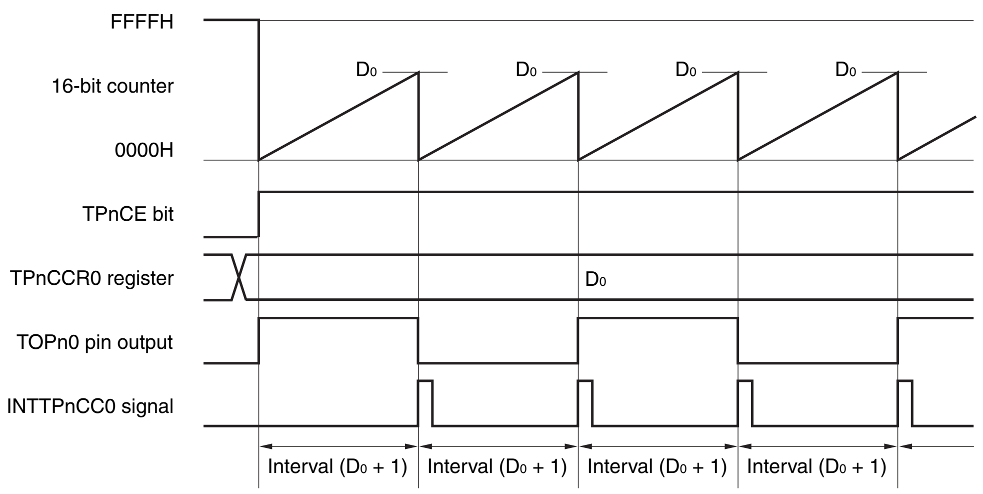


Figure 11-3 Basic timing of operation in interval timer mode

When the TPnCE bit is set to 1, the value of the 16-bit counter is cleared from FFFFH to 0000H in synchronization with the count clock, and the counter starts counting. At this time, the output of the TOPn0 pin is inverted. Additionally, the set value of the TPnCCR0 register is transferred to the CCR0 buffer register.

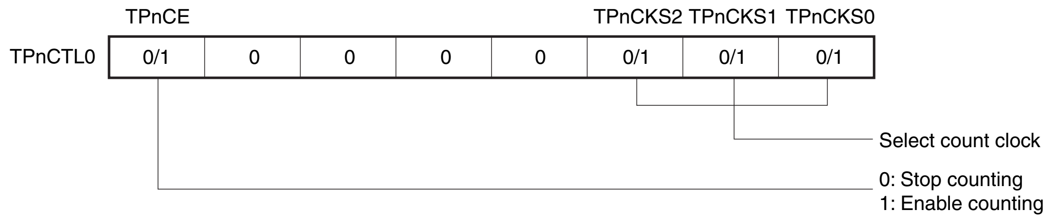
When the count value of the 16-bit counter matches the value of the CCR0 buffer register, the 16-bit counter is cleared to 0000H, the output of the TOPn0 pin is inverted, and a compare match interrupt request signal (INTTPnCC0) is generated.

The interval can be calculated by the following expression.

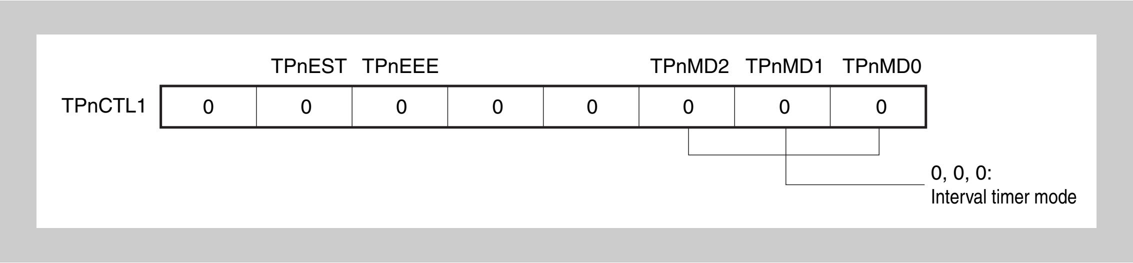
Interval = (Set value of TPnCCR0 register + 1) × Count clock cycle

(1) Register setting for interval timer mode operation

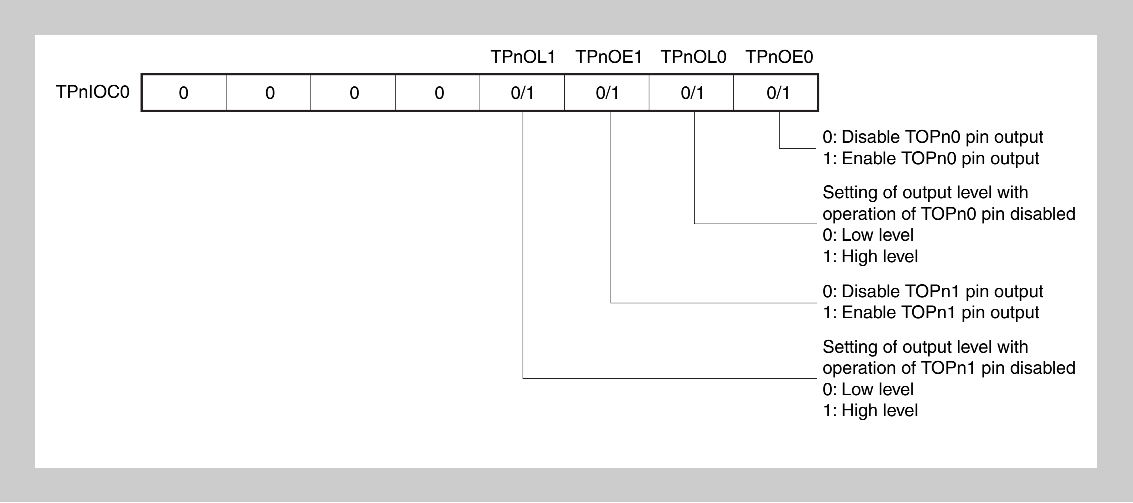
(a) TMPn control register 0 (TPnCTL0)



(b) TMPn control register 1 (TPnCTL1)



(c) TMPn I/O control register 0 (TPnIOC0)



(d) TMPn counter read buffer register (TPnCNT)

By reading the TPnCNT register, the count value of the 16-bit counter can be read.

(e) TMPn capture/compare register 0 (TPnCCR0)

If the TPnCCR0 register is set to D_0, the interval is as follows.

Interval = (D_0 + 1) × Count clock cycle

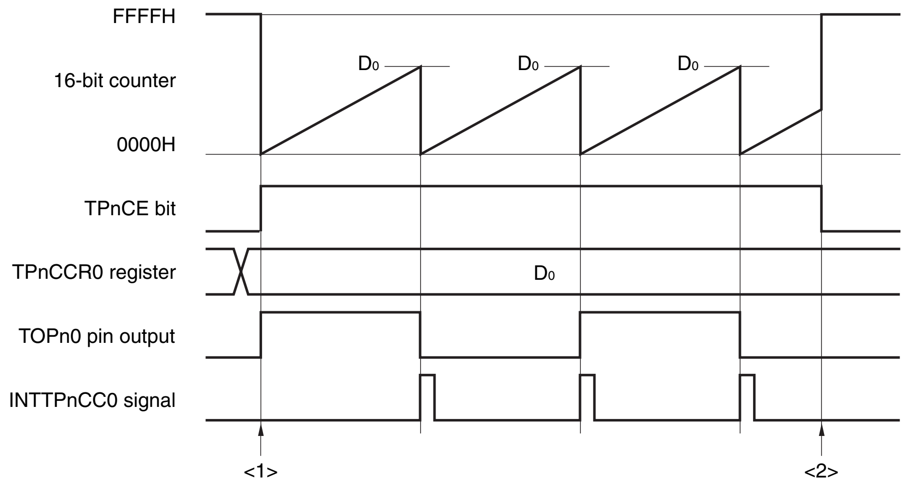
(f) TMPn capture/compare register 1 (TPnCCR1)

Usually, the TPnCCR1 register is not used in the interval timer mode. However, the set value of the TPnCCR1 register is transferred to the CCR1 buffer register. A compare match interrupt request signal (INTTPnCC1) is generated when the count value of the 16-bit counter matches the value of the CCR1 buffer register.

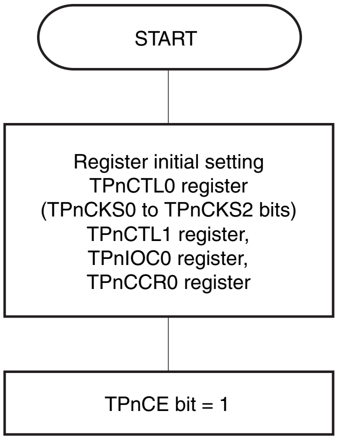
Therefore, mask the interrupt request by using the corresponding interrupt mask flag (TPnCCMK1).

Note TMPn I/O control register 1 (TPnIOC1), TMPn I/O control register 2 (TPnIOC2), and TMPn option register 0 (TPnOPT0) are not used in the interval timer mode.

(2) Interval timer mode operation flow



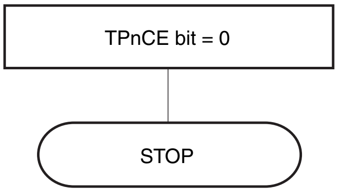
<1> Count operation start flow



Initial setting of these registers is performed before setting the TPnCE bit to 1.

The TPnCKS0 to TPnCKS2 bits can be set at the same time when counting has been started (TPnCE bit = 1).

<2> Count operation stop flow



The counter is initialized and counting is stopped by clearing the TPnCE bit to 0.

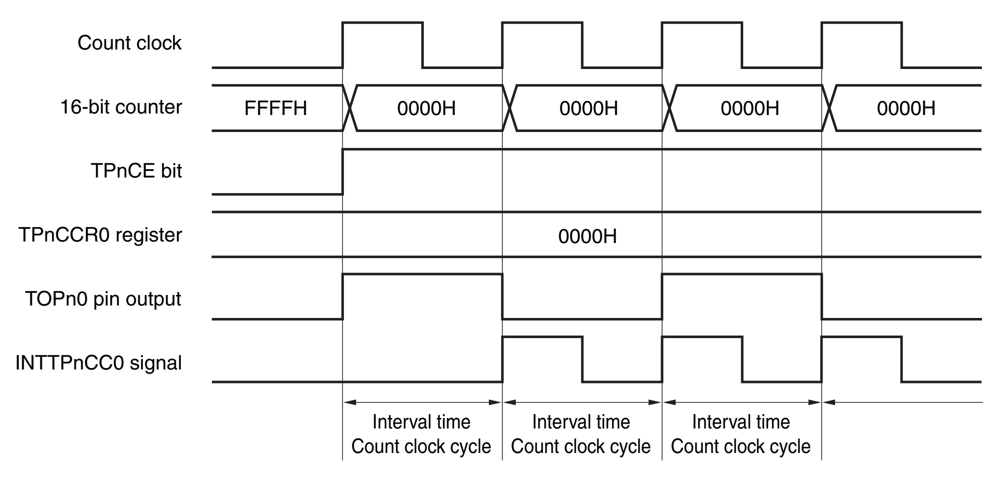
Figure 11-4 Software processing flow in interval timer mode

(3) Interval timer mode operation timing

(a) Operation if TPnCCR0 register is set to 0000H

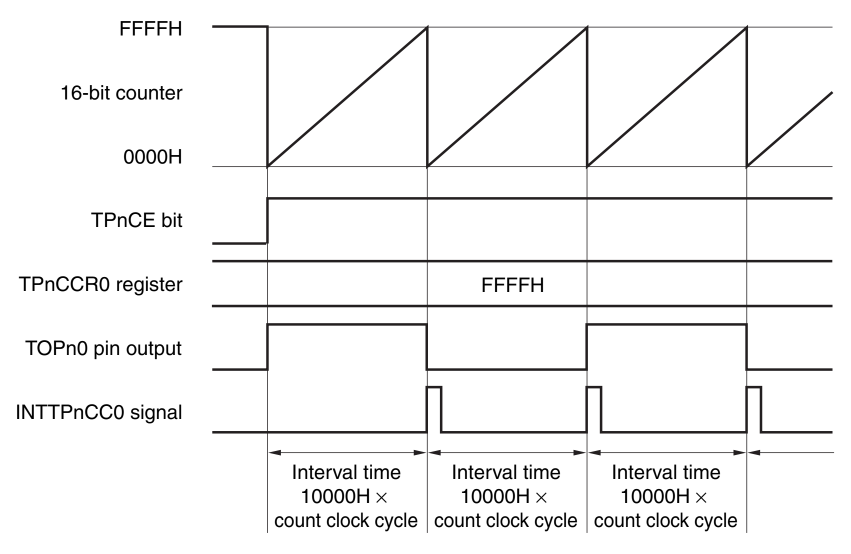
If the TPnCCR0 register is set to 0000H, the INTTPnCC0 signal is generated at each count clock, and the output of the TOPn0 pin is inverted.

The value of the 16-bit counter is always 0000H.



(b) Operation if TPnCCR0 register is set to FFFFH

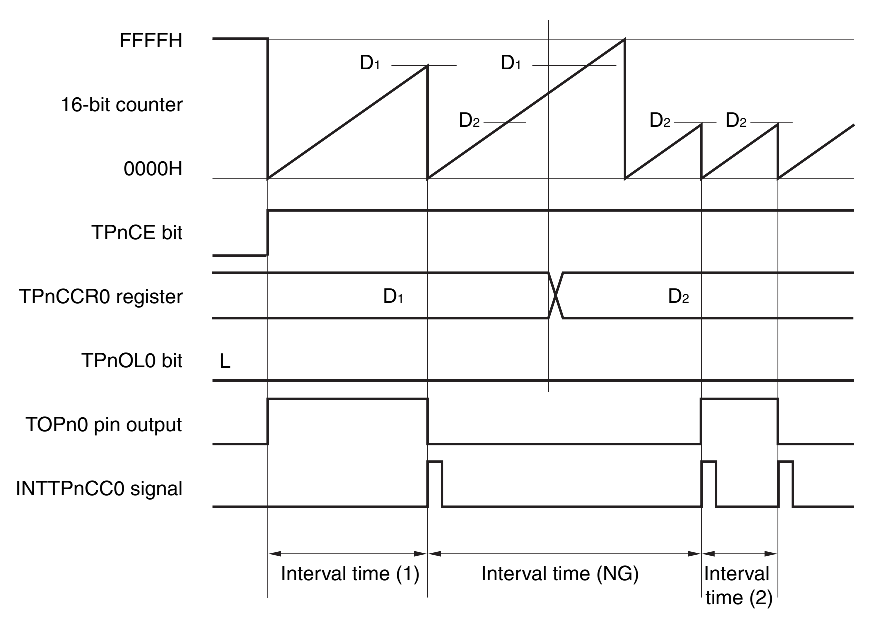
If the TPnCCR0 register is set to FFFFH, the 16-bit counter counts up to FFFFH. The counter is cleared to 0000H in synchronization with the next count-up timing. The INTTPnCC0 signal is generated and the output of the TOPn0 pin is inverted. At this time, an overflow interrupt request signal (INTTPnOV) is not generated, nor is the overflow flag (TPnOPT0.TPnOVF bit) set to 1.



(c) Notes on rewriting TPnCCR0 register

To change the value of the TPnCCR0 register to a smaller value, stop counting once and then change the set value.

If the value of the TPnCCR0 register is rewritten to a smaller value during counting, the 16-bit counter may overflow.



Note
1. Interval time (1): (D_1 + 1) × Count clock cycle
2. Interval time (NG): (10000H + D_2 + 1) × Count clock cycle
3. Interval time (2): (D_2 + 1) × Count clock cycle

If the value of the TPnCCR0 register is changed from D_1 to D_2 while the count value is greater than D_2 but less than D_1, the count value is transferred to the CCR0 buffer register as soon as the TPnCCR0 register has been rewritten. Consequently, the value of the 16-bit counter that is compared is D_2.

Because the count value has already exceeded D_2, however, the 16-bit counter counts up to FFFFH, overflows, and then counts up again from 0000H. When the count value matches D_2, the INTTPnCC0 signal is generated and the output of the TOPn0 pin is inverted.

Therefore, the INTTPnCC0 signal may not be generated at the interval time "(D_1 + 1) × Count clock cycle" or "(D_2 + 1) × Count clock cycle" originally expected, but may be generated at an interval of "(10000H + D_2 + 1) × Count clock period".

(d) Operation of TPnCCR1 register

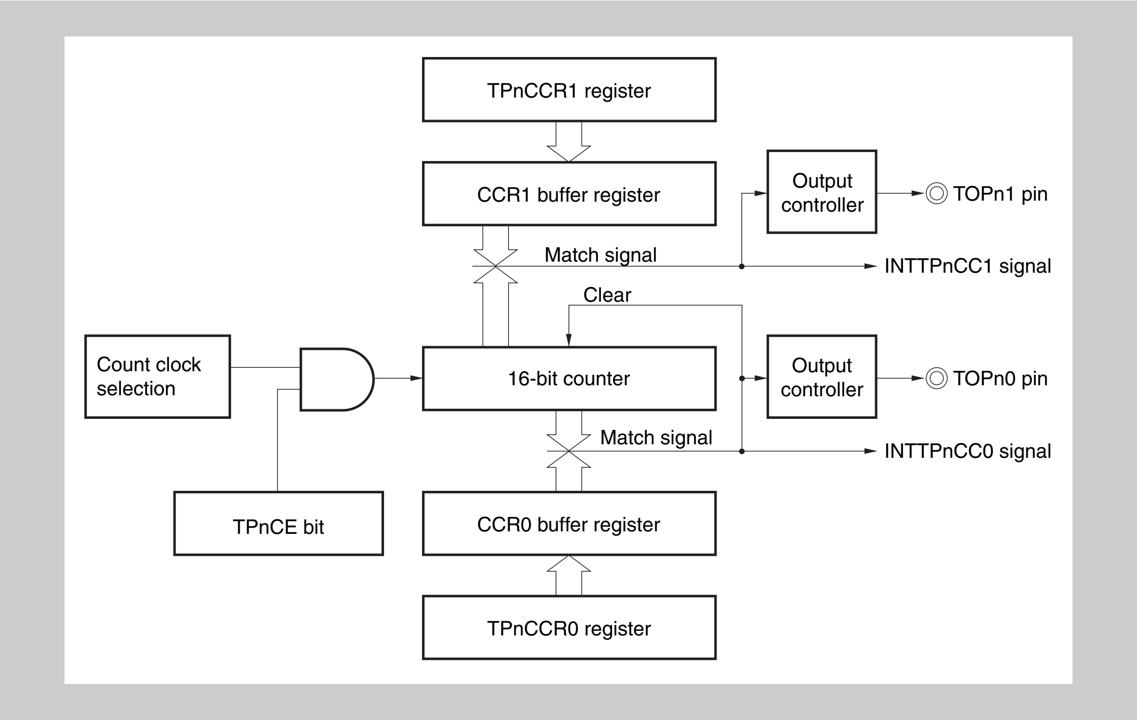


Figure 11-5 Configuration of TPnCCR1 register

If the set value of the TPnCCR1 register is less than the set value of the TPnCCR0 register, the INTTPnCC1 signal is generated once per cycle. At the same time, the output of the TOPn1 pin is inverted.

The TOPn1 pin outputs a square wave with the same cycle as that output by the TOPn0 pin.

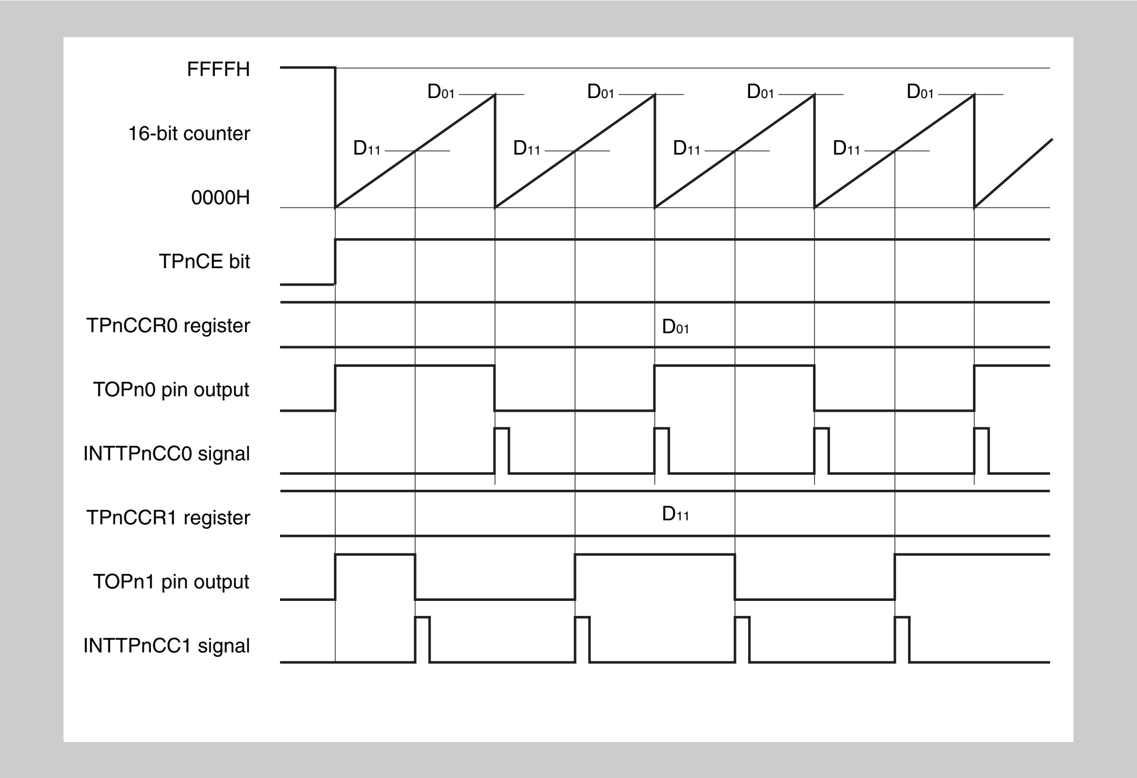


Figure 11-6 Timing chart when $D_{01} \geq D_{11}$

If the set value of the TPnCCR1 register is greater than the set value of the TPnCCR0 register, the count value of the 16-bit counter does not match the value of the TPnCCR1 register. Consequently, the INTTPnCC1 signal is not generated, nor is the output of the TOPn1 pin changed.

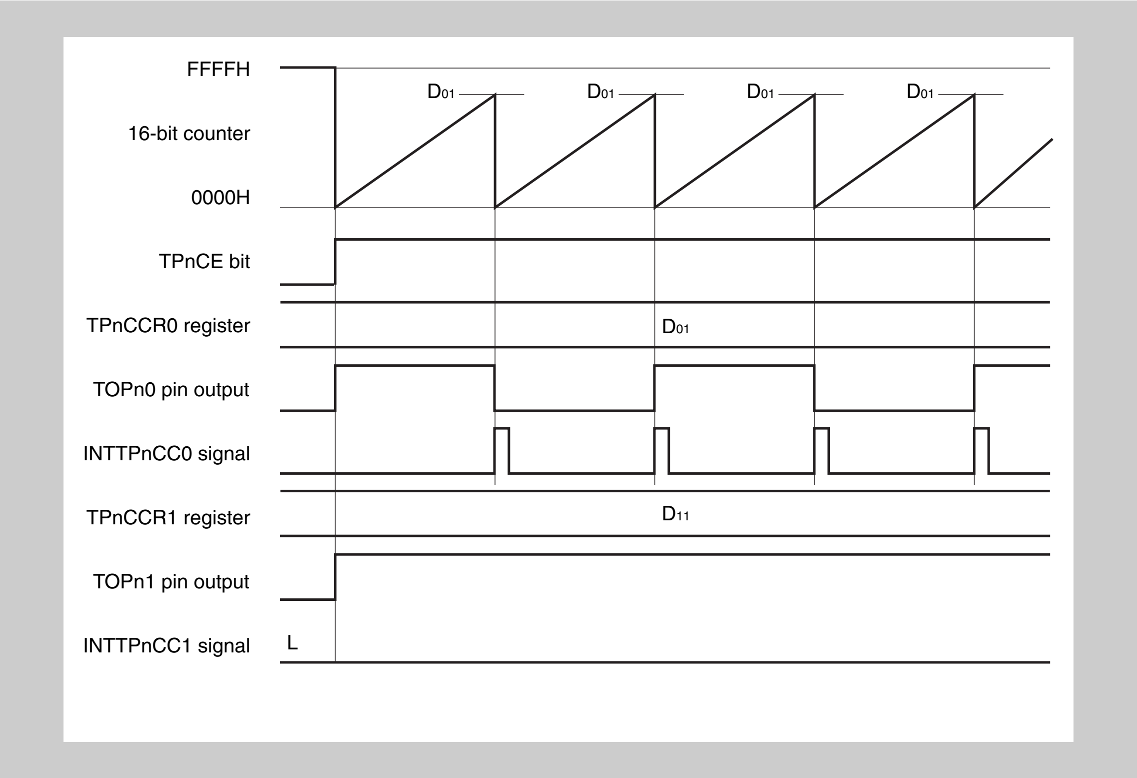


Figure 11-7 Timing chart when $D_{01} < D_{11}$

11.5.2 External event count mode (TPnMD2 to TPnMD0 = 001)

In the external event count mode, the valid edge of the external event count input is counted when the TPnCTL0.TPnCE bit is set to 1, and an interrupt request signal (INTTPnCC0) is generated each time the specified number of edges have been counted. The TOPn0 pin cannot be used.

Usually, the TPnCCR1 register is not used in the external event count mode.

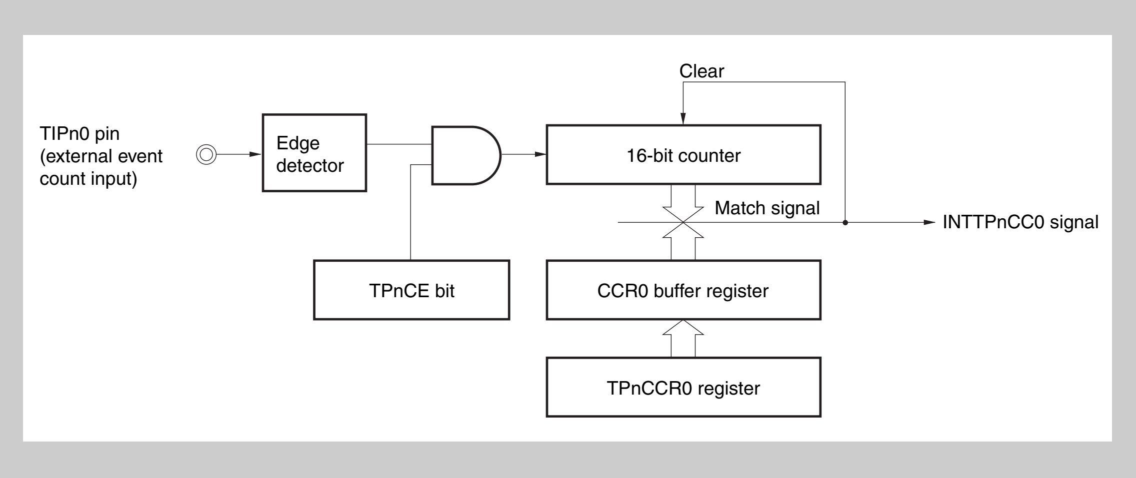


Figure 11-8 Configuration in external event count mode

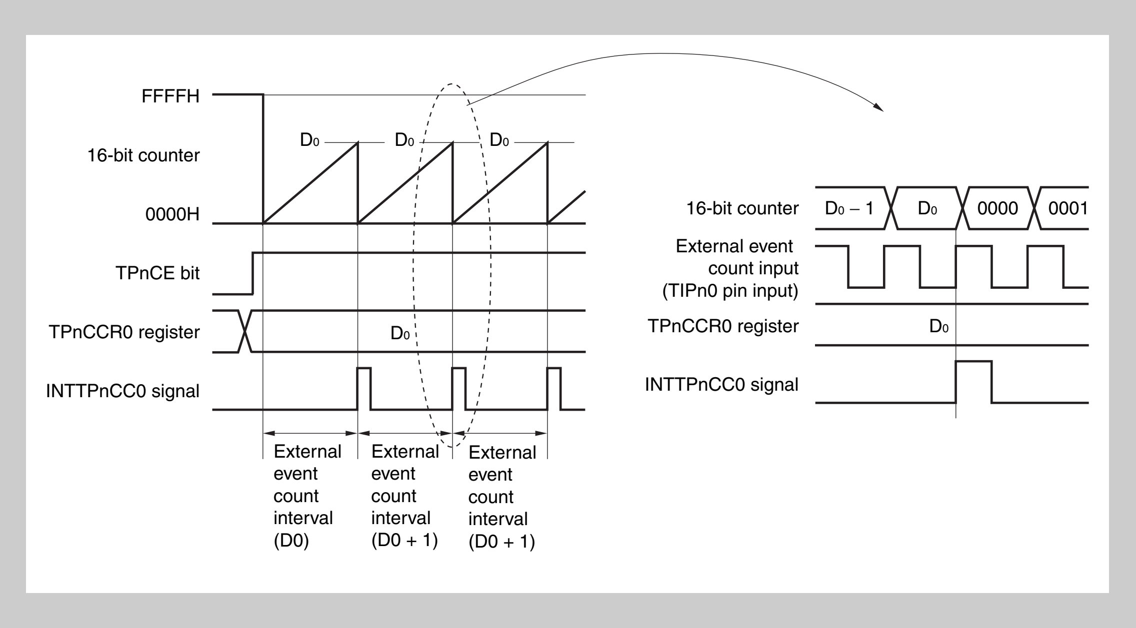


Figure 11-9 Basic timing in external event count mode

Caution This figure shows the basic timing when the rising edge is specified as the valid edge of the external event count input.

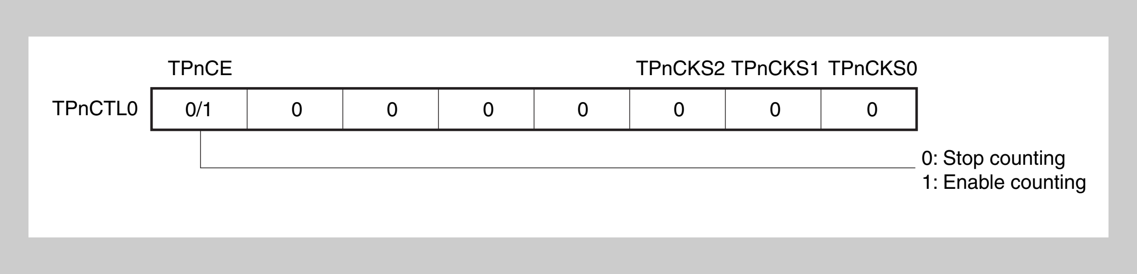
When the TPnCE bit is set to 1, the value of the 16-bit counter is cleared from FFFFH to 0000H. The counter counts each time the valid edge of external event count input is detected. Additionally, the set value of the TPnCCR0 register is transferred to the CCR0 buffer register.

When the count value of the 16-bit counter matches the value of the CCR0 buffer register, the 16-bit counter is cleared to 0000H, and a compare match interrupt request signal (INTTPnCC0) is generated.

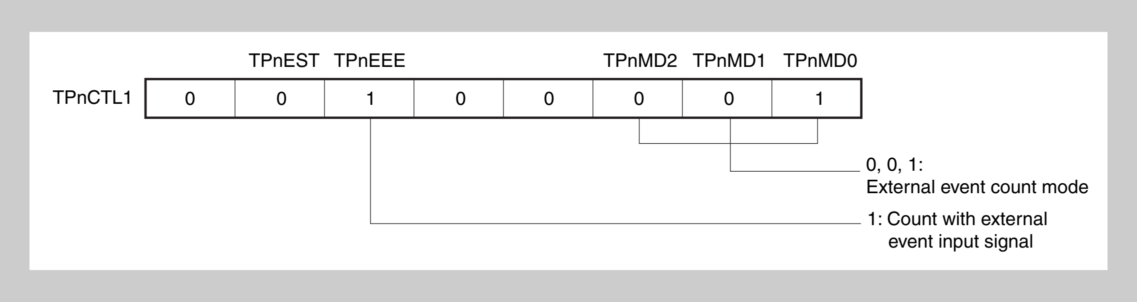
The INTTPnCC0 signal is generated each time the valid edge of the external event count input has been detected (set value of TPnCCR0 register + 1) times.

(1) Register setting for operation in external event count mode

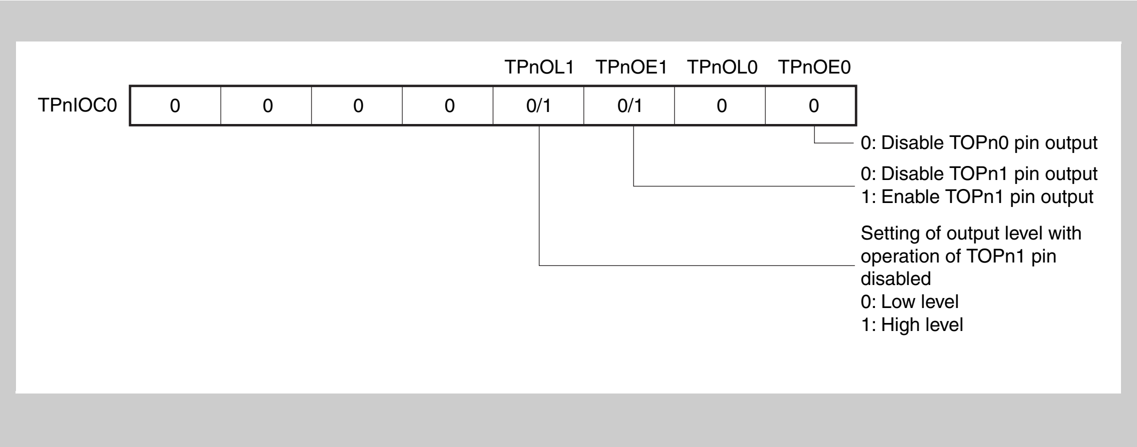
(a) TMPn control register 0 (TPnCTL0)



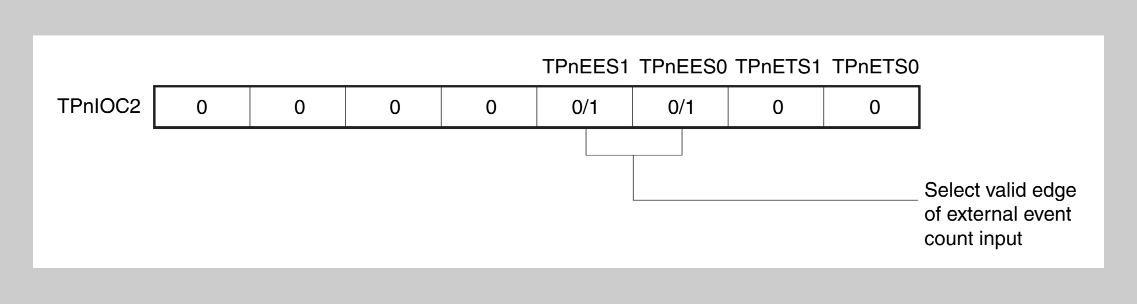
(b) TMPn control register 1 (TPnCTL1)



(c) TMPn I/O control register 0 (TPnIOC0)



(d) TMPn I/O control register 2 (TPnIOC2)



(e) TMPn counter read buffer register (TPnCNT)

The count value of the 16-bit counter can be read by reading the TPnCNT register.

(f) TMPn capture/compare register 0 (TPnCCR0)

If D_0 is set to the TPnCCR0 register, the counter is cleared and a compare match interrupt request signal (INTTPnCC0) is generated when the number of external event counts reaches (D_0 + 1).

(g) TMPn capture/compare register 1 (TPnCCR1)

Usually, the TPnCCR1 register is not used in the external event count mode. However, the set value of the TPnCCR1 register is transferred to the CCR1 buffer register. When the count value of the 16-bit counter matches the value of the CCR1 buffer register, a compare match interrupt request signal (INTTPnCC1) is generated.

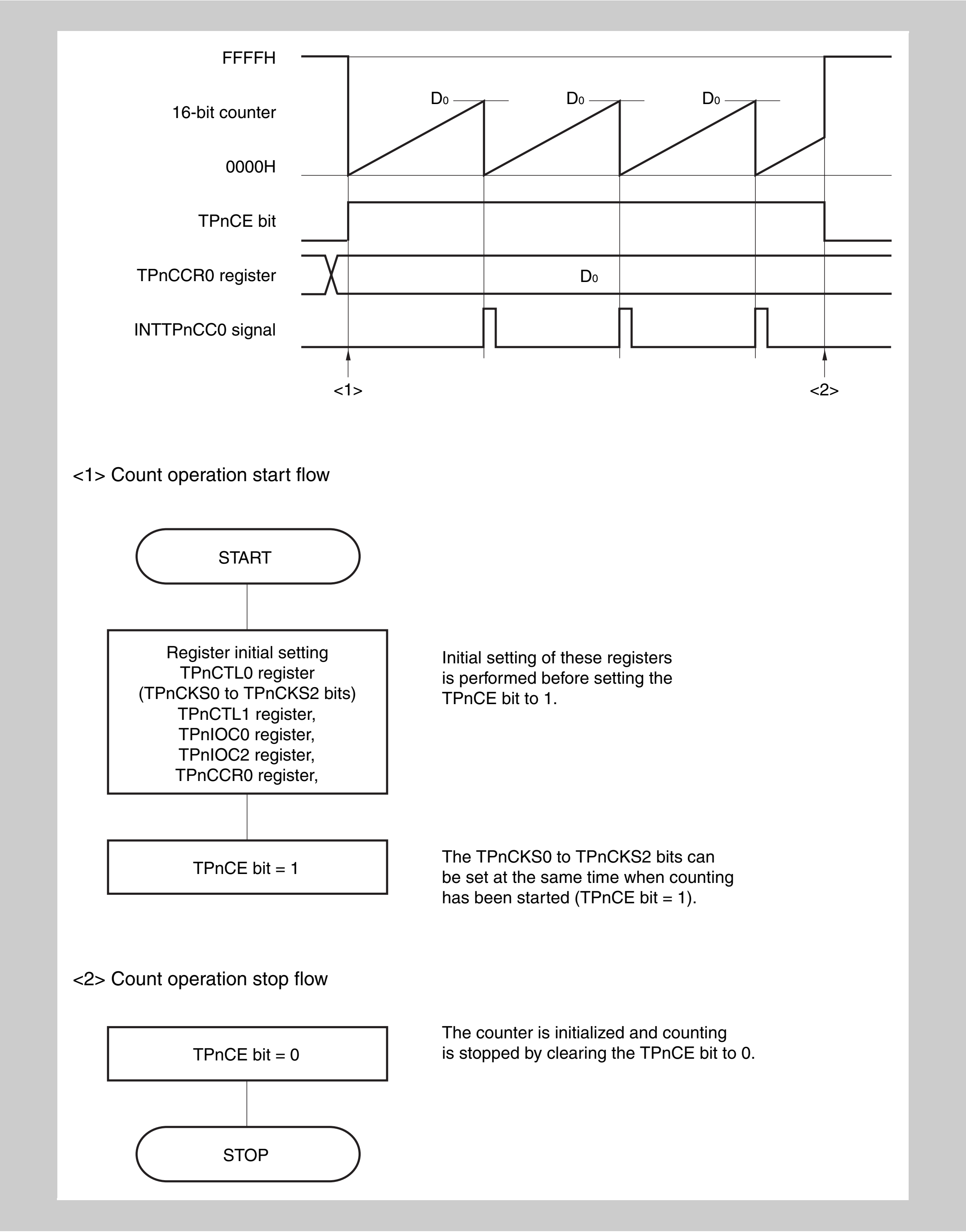
Therefore, mask the interrupt signal by using the interrupt mask flag (TPnCCMK1).

Note TMPn I/O control register 1 (TPnIOC1) and TMPn option register 0 (TPnOPT0) are not used in the external event count mode.

Caution When the compare register TPnCCR0 (TPnCCR1) is set to 0000_H and the external event counter mode is started the first interrupt INTTPnCC0 (INTTPnCC1) occurs upon the first timer overflow (TPnCNT: $FFFF_H \rightarrow 0000_H$), but not with the first external count event.

Afterwards the following interrupts INTTPnCC0 (INTTPnCC1) are generated as specified, i.e. with each external count event.

(2) External event count mode operation flow



<1> Count operation start flow

Initial setting of these registers is performed before setting the TPnCE bit to 1.

The TPnCKS0 to TPnCKS2 bits can be set at the same time when counting has been started (TPnCE bit = 1).

<2> Count operation stop flow

The counter is initialized and counting is stopped by clearing the TPnCE bit to 0.

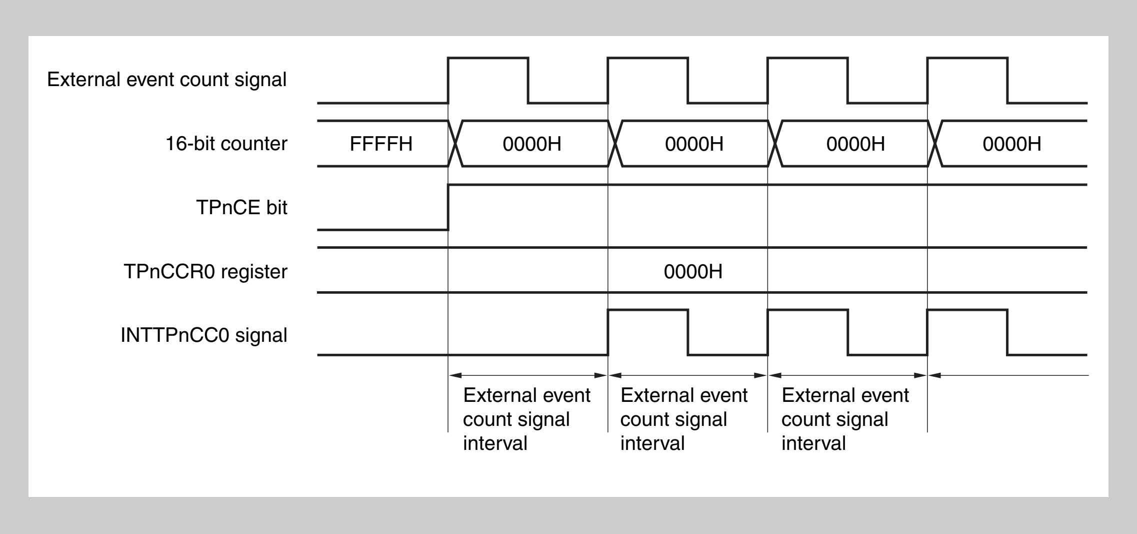
Figure 11-10 Flow of software processing in external event count mode

(3) Operation timing in external event count mode

(a) Operation if TPnCCR0 register is set to 0000H

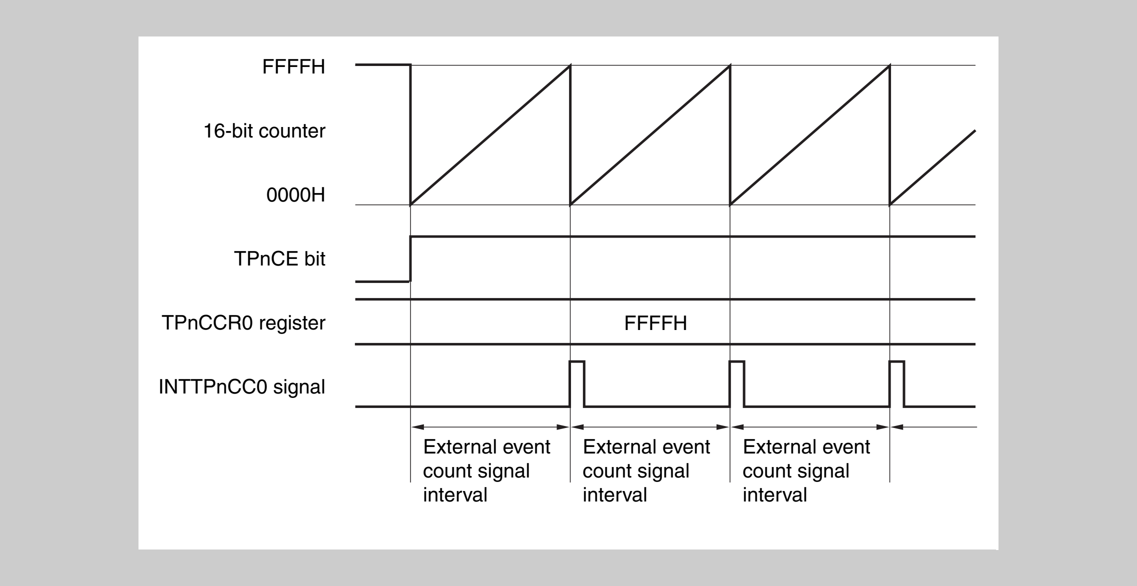
If the TPnCCR0 register is set to 0000H, the INTTPnCC0 signal is generated each time the valid signal of the external event count signal has been detected.

The 16-bit counter is always 0000H.



(b) Operation if TPnCCR0 register is set to FFFFH

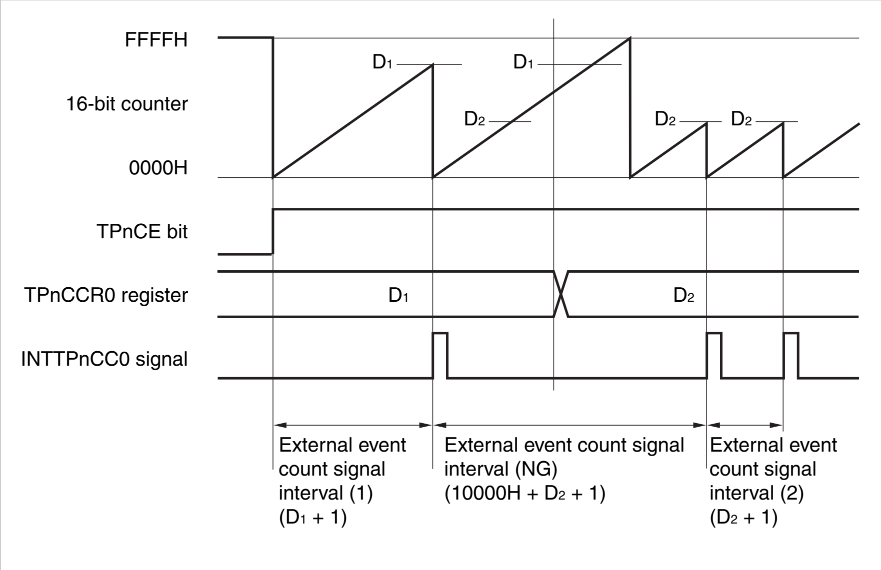
If the TPnCCR0 register is set to FFFFH, the 16-bit counter counts to FFFFH each time the valid edge of the external event count signal has been detected. The 16-bit counter is cleared to 0000H in synchronization with the next count-up timing, and the INTTPnCC0 signal is generated. At this time, the TPnOPT0.TPnOVF bit is not set.



(c) Notes on rewriting the TPnCCR0 register

To change the value of the TPnCCR0 register to a smaller value, stop counting once and then change the set value.

If the value of the TPnCCR0 register is rewritten to a smaller value during counting, the 16-bit counter may overflow.



If the value of the TPnCCR0 register is changed from D_1 to D_2 while the count value is greater than D_2 but less than D_1, the count value is transferred to the CCR0 buffer register as soon as the TPnCCR0 register has been rewritten. Consequently, the value that is compared with the 16-bit counter is D_2.

Because the count value has already exceeded D_2, however, the 16-bit counter counts up to FFFFH, overflows, and then counts up again from 0000H. When the count value matches D_2, the INTTPnCC0 signal is generated.

Therefore, the INTTPnCC0 signal may not be generated at the valid edge count of “(D_1 + 1) times” or “(D_2 + 1) times” originally expected, but may be generated at the valid edge count of “(10000H + D_2 + 1) times”.

(d) Operation of TPnCCR1 register

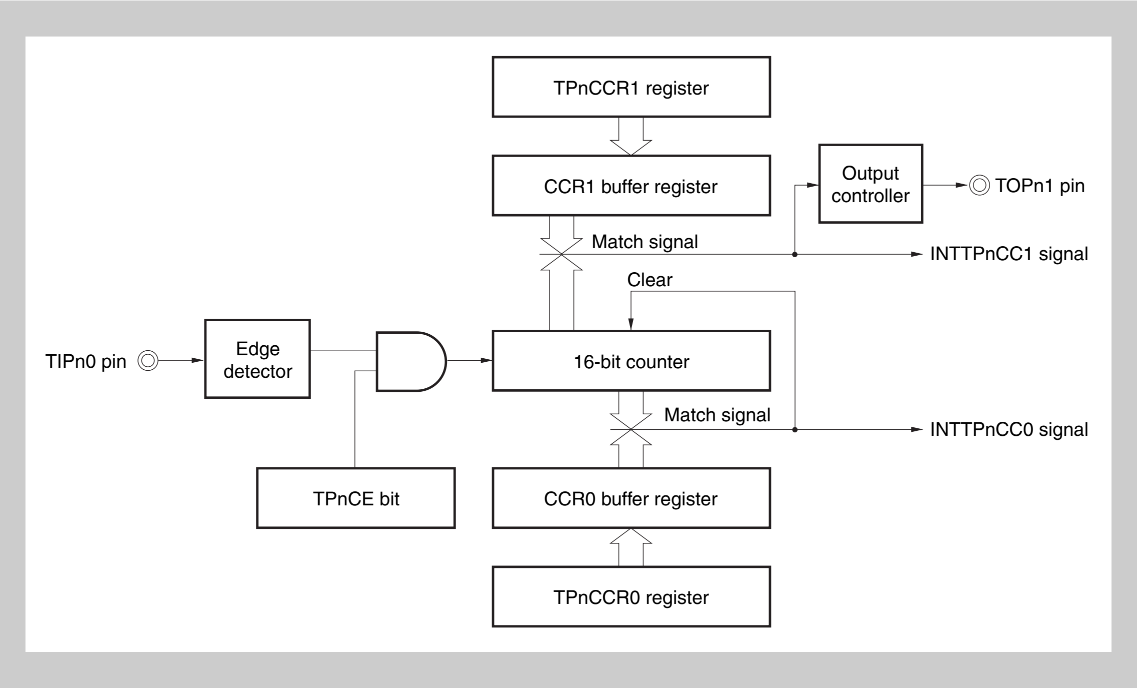


Figure 11-11 Configuration of TPnCCR1 register

If the set value of the TPnCCR1 register is smaller than the set value of the TPnCCR0 register, the INTTPnCC1 signal is generated once per cycle. At the same time, the output signal of the TOPn1 pin is inverted.

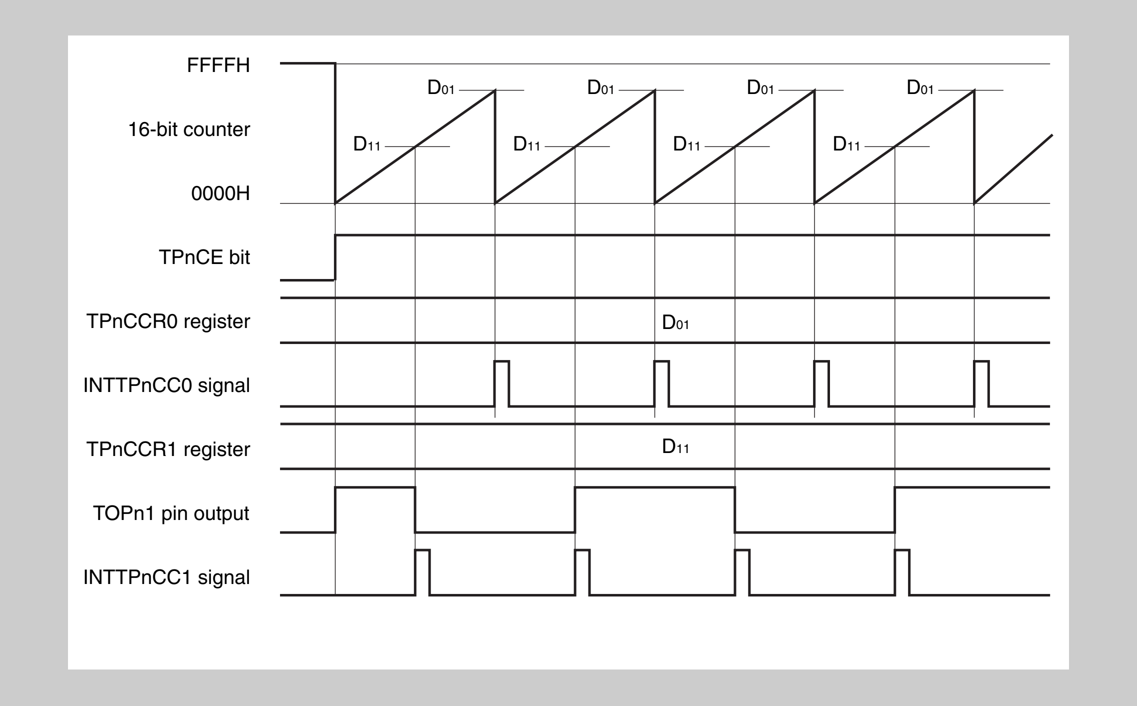


Figure 11-12 Timing chart when $D_{01} \geq D_{11}$

If the set value of the TPnCCR1 register is greater than the set value of the TPnCCR0 register, the INTTPnCC1 signal is not generated because the count value of the 16-bit counter and the value of the TPnCCR1 register do not match. Nor is the output signal of the TOPn1 pin changed.

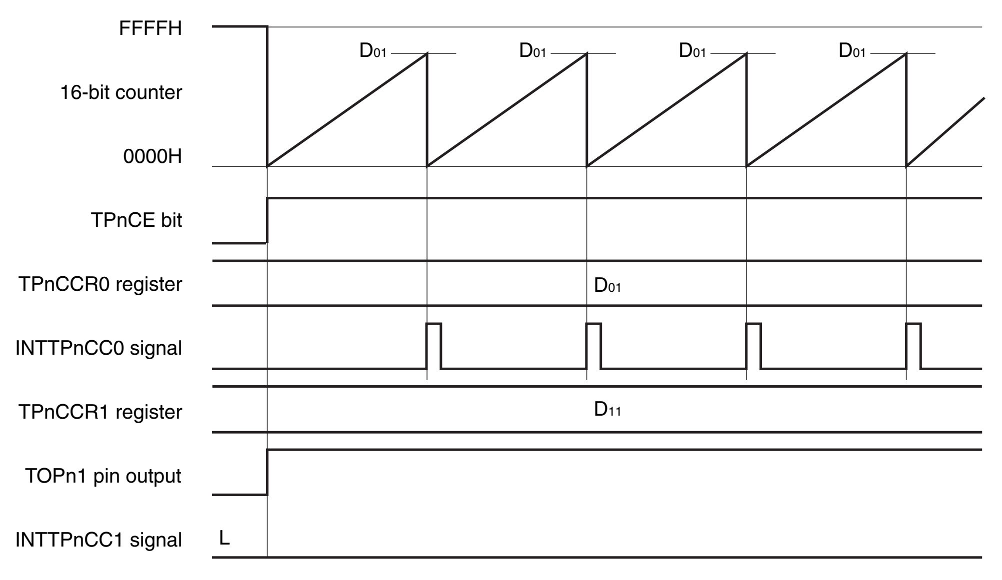


Figure 11-13 Timing chart when $D_{01} < D_{11}$

11.5.3 External trigger pulse output mode (TPnMD2 to TPnMD0 = 010)

In the external trigger pulse output mode, 16-bit timer/event counter P waits for a trigger when the TPnCTL0.TPnCE bit is set to 1. When the valid edge of an external trigger input signal is detected, 16-bit timer/event counter P starts counting, and outputs a PWM waveform from the TOPn1 pin.

Pulses can also be output by generating a software trigger instead of using the external trigger. When using a software trigger, a square wave that has one cycle of the PWM waveform as half its cycle can also be output from the TOPn0 pin.

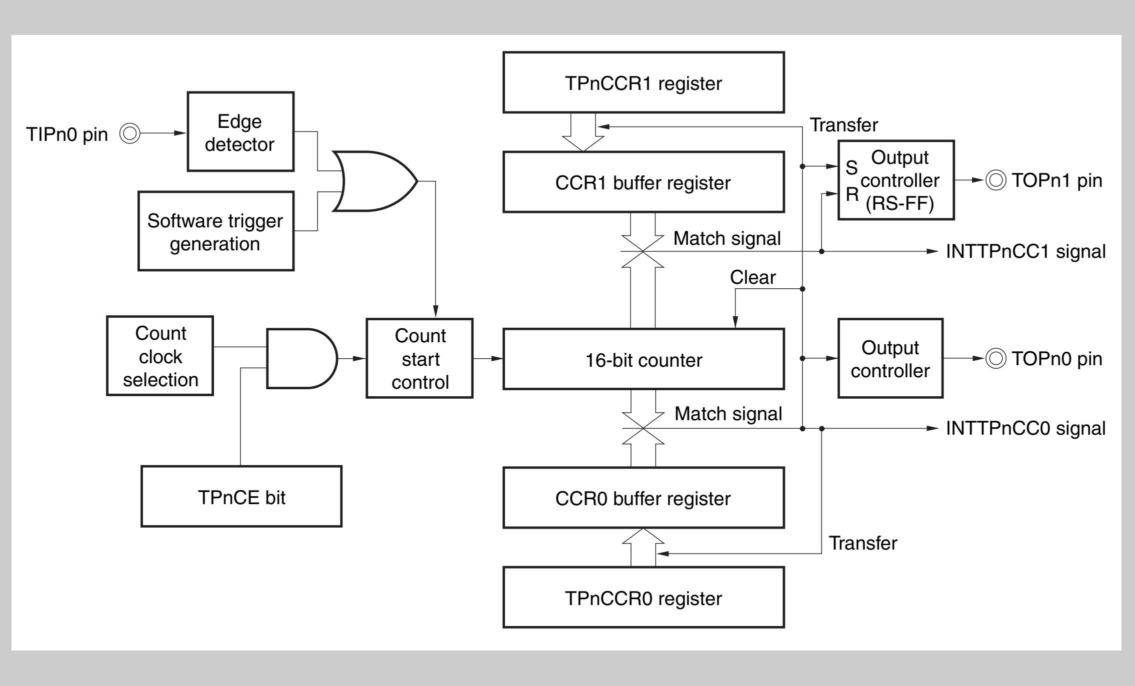


Figure 11-14 Configuration in external trigger pulse output mode

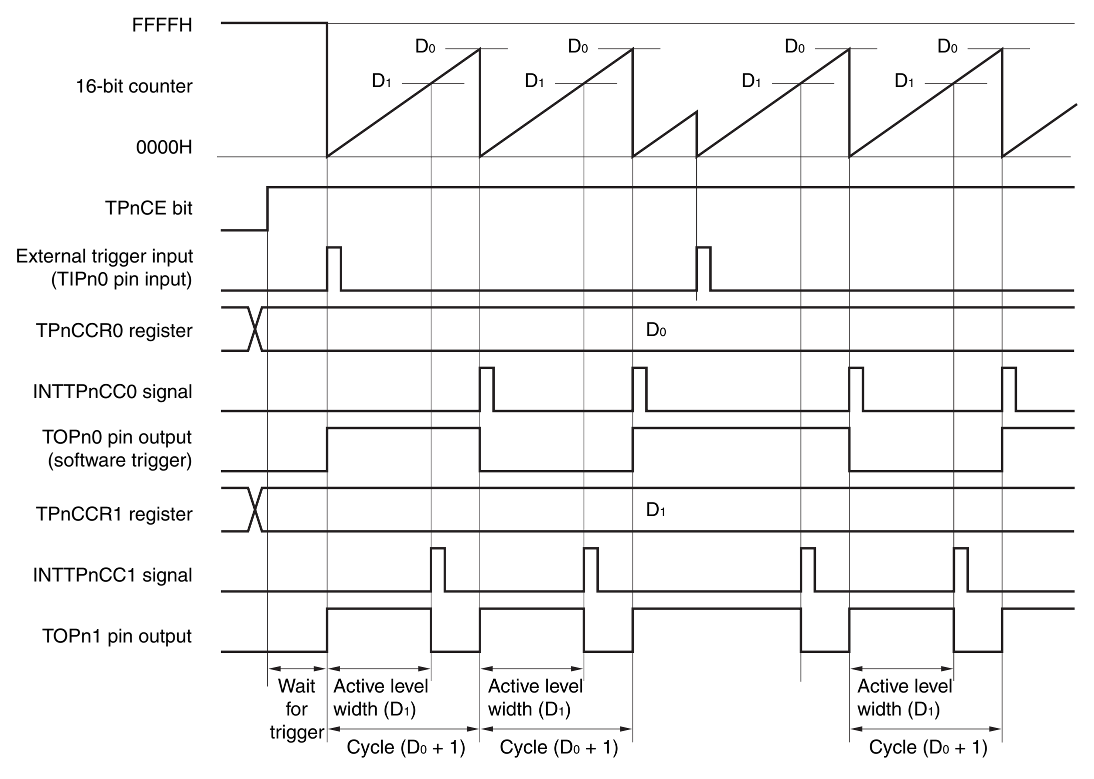


Figure 11-15 Basic timing in external trigger pulse output mode

16-bit timer/event counter P waits for a trigger when the TPnCE bit is set to 1. When the trigger is generated, the 16-bit counter is cleared from FFFFH to 0000H, starts counting at the same time, and outputs a PWM waveform from the TOPn1 pin.

If the trigger is generated again while the counter is operating, the counter is cleared to 0000H and restarted.

The active level width, cycle, and duty factor of the PWM waveform can be calculated as follows.

Active level width = (Set value of TPnCCR1 register) × Count clock cycle

Cycle = (Set value of TPnCCR0 register + 1) × Count clock cycle

Duty factor = (Set value of TPnCCR1 register)/(Set value of TPnCCR0 register + 1)

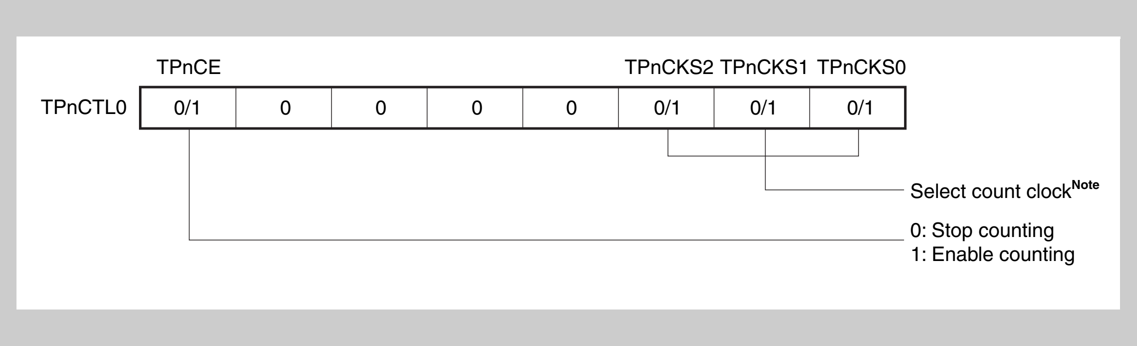
The compare match request signal INTTPnCC0 is generated when the 16-bit counter counts next time after its count value matches the value of the CCR0 buffer register, and the 16-bit counter is cleared to 0000H. The compare match interrupt request signal INTTPnCC1 is generated when the count value of the 16-bit counter matches the value of the CCR1 buffer register.

The value set to the TPnCCRm register is transferred to the CCRm buffer register when the count value of the 16-bit counter matches the value of the CCRm buffer register and the 16-bit counter is cleared to 0000H.

The valid edge of an external trigger input signal, or setting the software trigger (TPnCTL1.TPnEST bit) to 1 is used as the trigger.

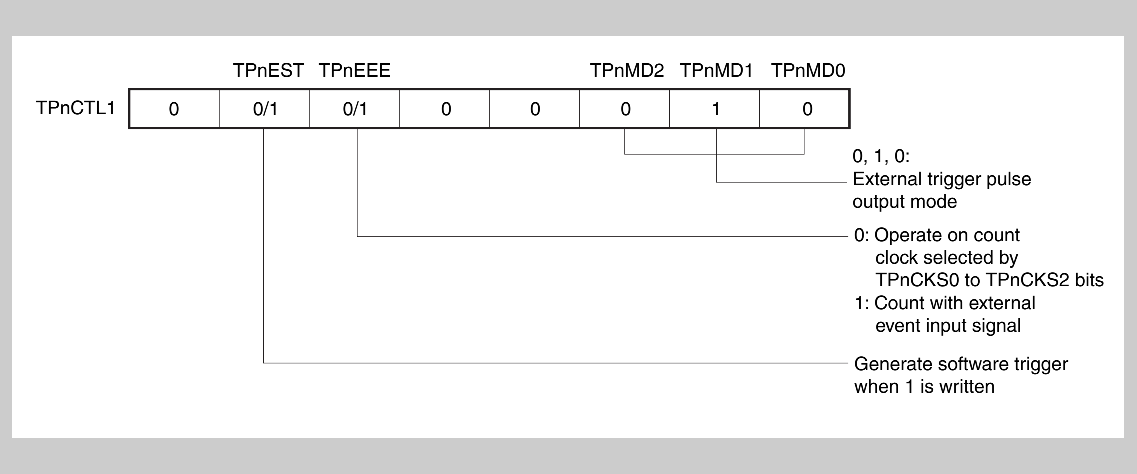
(1) Setting of registers in external trigger pulse output mode

(a) TMPn control register 0 (TPnCTL0)

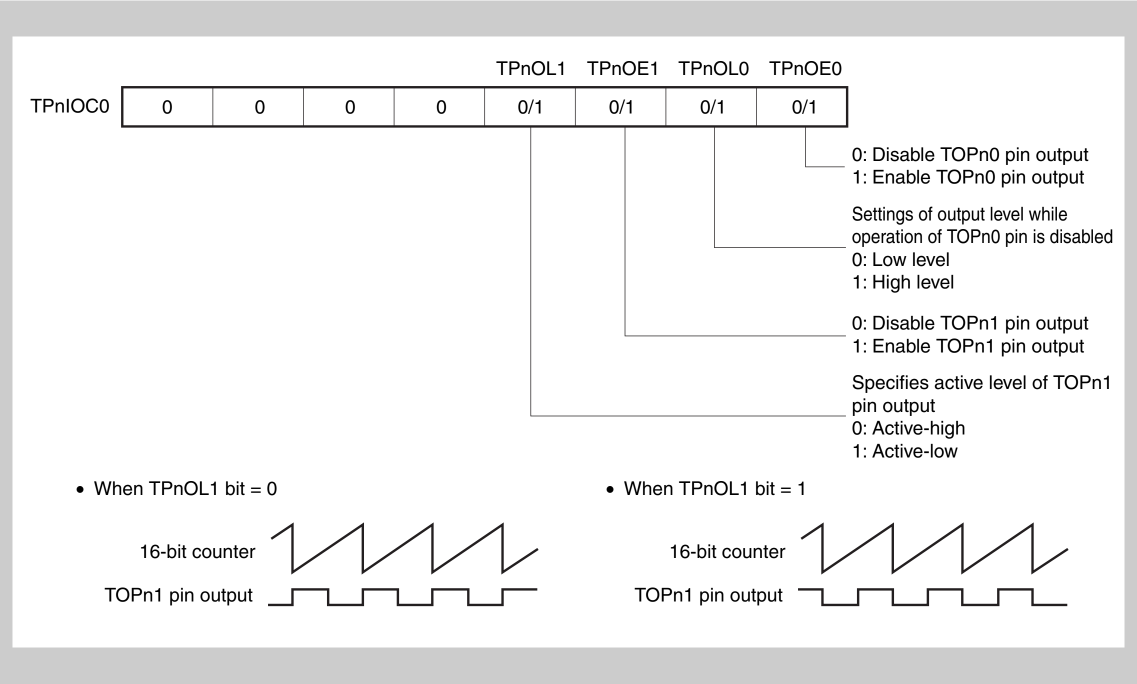


Note The setting is invalid when the TPnCTL1.TPnEEE bit = 1.

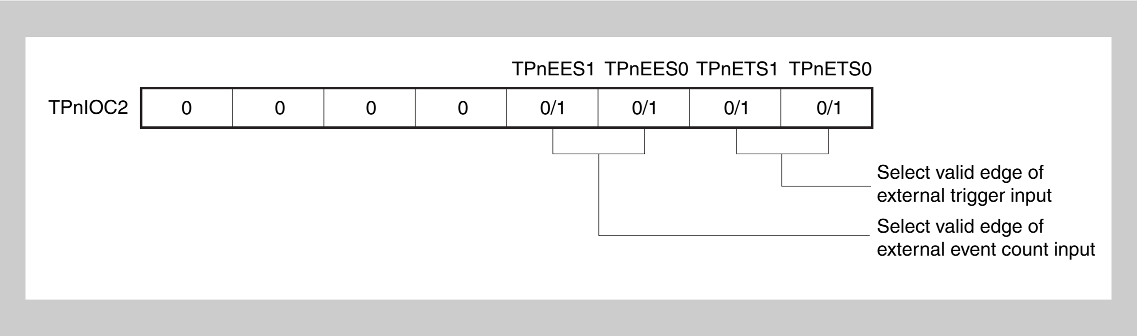
(b) TMPn control register 1 (TPnCTL1)



(c) TMPn I/O control register 0 (TPnIOC0)



(d) TMPn I/O control register 2 (TPnIOC2)



(e) TMPn counter read buffer register (TPnCNT)

The value of the 16-bit counter can be read by reading the TPnCNT register.

(f) TMPn capture/compare registers 0 and 1 (TPnCCR0 and TPnCCR1)

If D_0 is set to the TPnCCR0 register and D_1 to the TPnCCR1 register, the cycle and active level of the PWM waveform are as follows.

Cycle = $(D_0 + 1)$ × Count clock cycle

Active level width = D_1 × Count clock cycle

Note TMPn I/O control register 1 (TPnIOC1) and TMPn option register 0 (TPnOPT0) are not used in the external trigger pulse output mode.

(2) Operation flow in external trigger pulse output mode

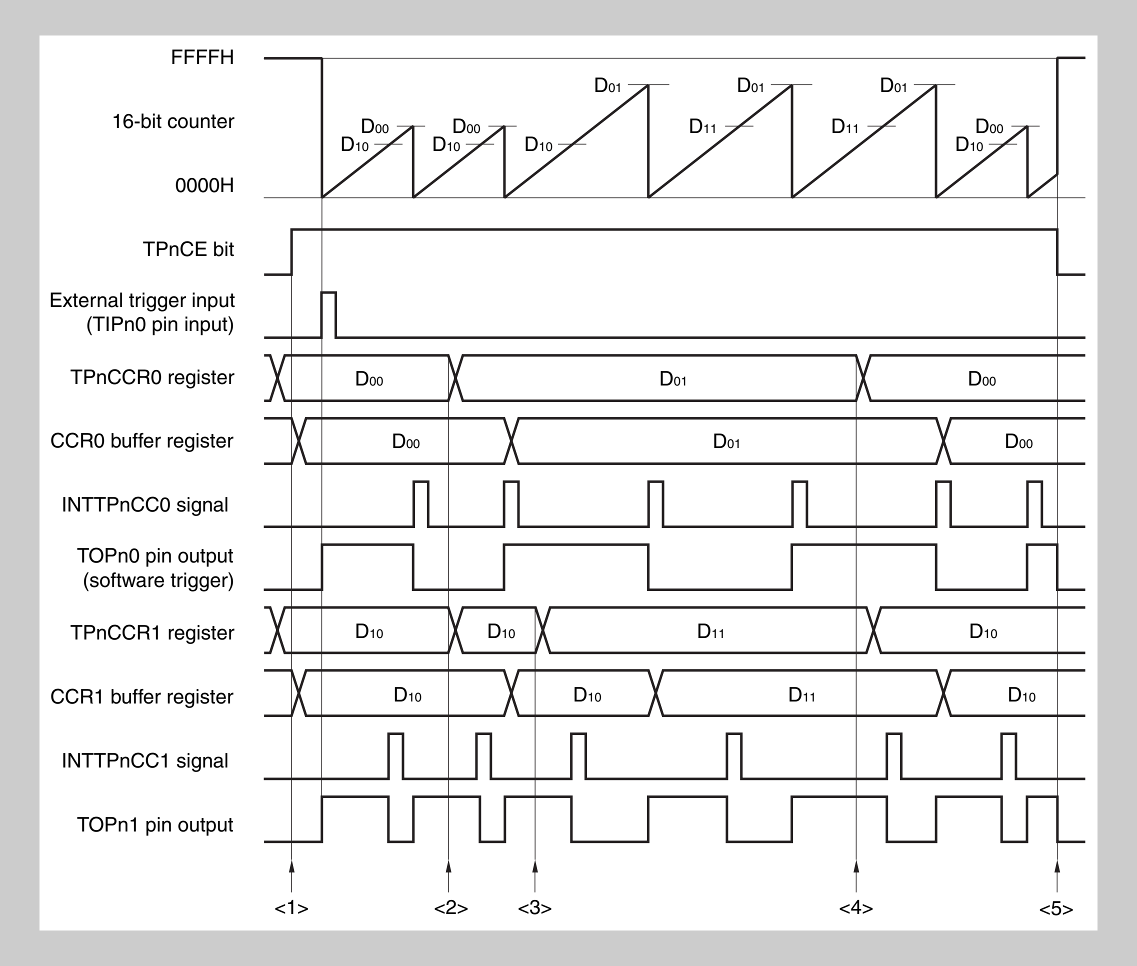
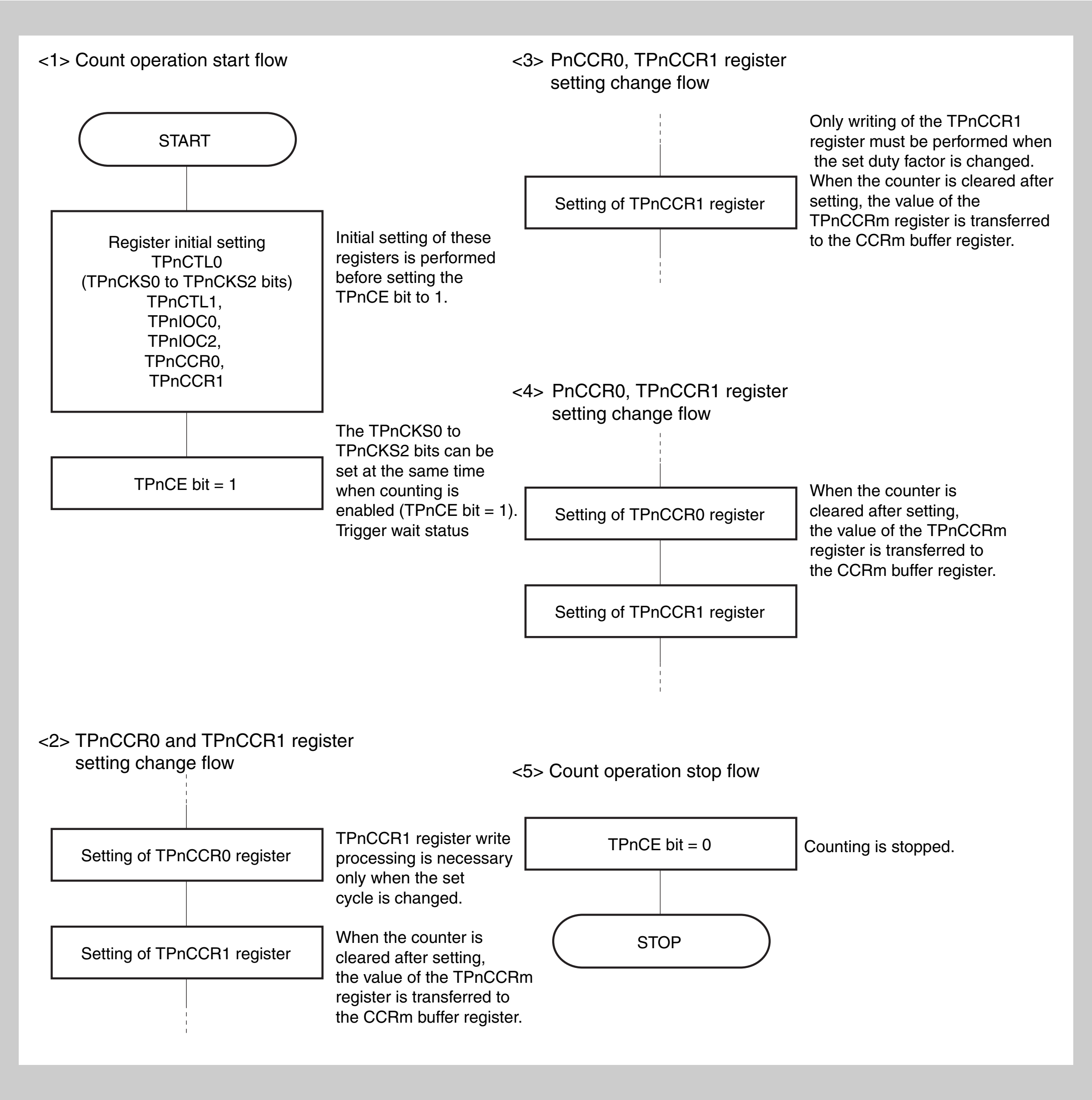


Figure 11-16 Software processing flow in external trigger pulse output mode (1/2)



Initial setting of these registers is performed before setting the TPnCE bit to 1.

The TPnCKS0 to TPnCKS2 bits can be set at the same time when counting is enabled (TPnCE bit = 1). Trigger wait status

TPnCCR1 register write processing is necessary only when the set cycle is changed.

When the counter is cleared after setting, the value of the TPnCCRm register is transferred to the CCRm buffer register.

Only writing of the TPnCCR1 register must be performed when the set duty factor is changed. When the counter is cleared after setting, the value of the TPnCCRm register is transferred to the CCRm buffer register.

When the counter is cleared after setting, the value of the TPnCCRm register is transferred to the CCRm buffer register.

Counting is stopped.

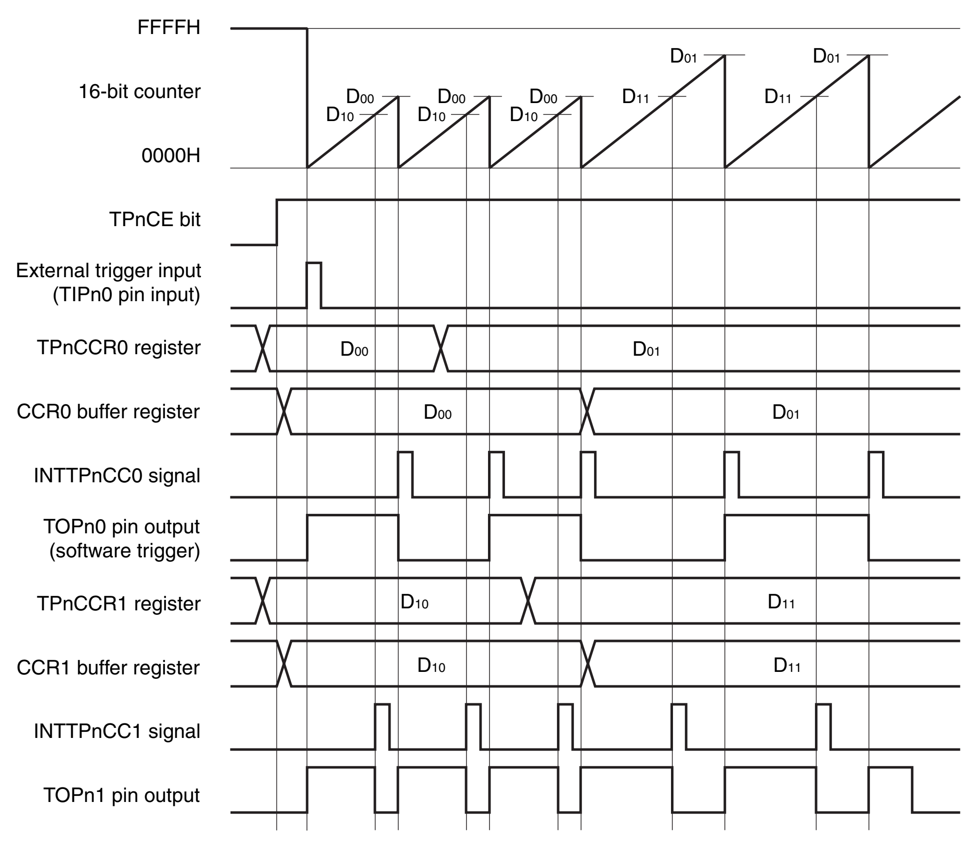
Figure 11-17 Software processing flow in external trigger pulse output mode (2/2)

(3) External trigger pulse output mode operation timing

(a) Note on changing pulse width during operation

To change the PWM waveform while the counter is operating, write the TPnCCR1 register last.

Rewrite the TPnCCRm register after writing the TPnCCR1 register after the INTTPnCC0 signal is detected.



In order to transfer data from the TPnCCRm register to the CCRm buffer register, the TPnCCR1 register must be written.

To change both the cycle and active level width of the PWM waveform at this time, first set the cycle to the TPnCCR0 register and then set the active level width to the TPnCCR1 register.

To change only the cycle of the PWM waveform, first set the cycle to the TPnCCR0 register, and then write the same value to the TPnCCR1 register.

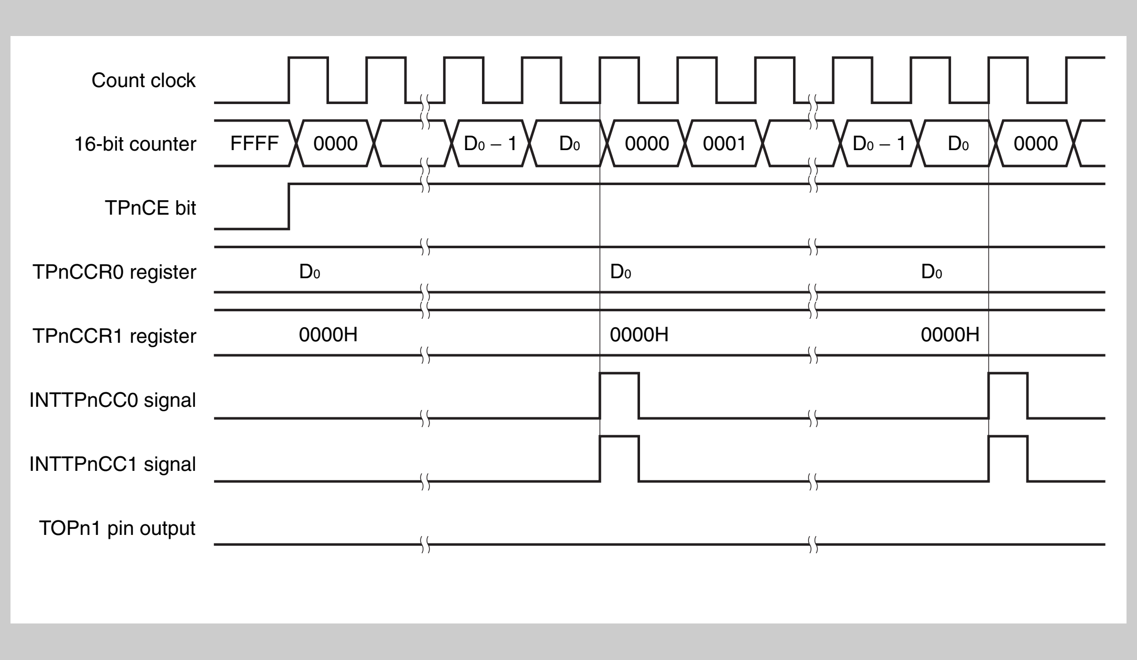
To change only the active level width (duty factor) of the PWM waveform, only the TPnCCR1 register has to be set.

After data is written to the TPnCCR1 register, the value written to the TPnCCRm register is transferred to the CCRm buffer register in synchronization with clearing of the 16-bit counter, and is used as the value compared with the 16-bit counter.

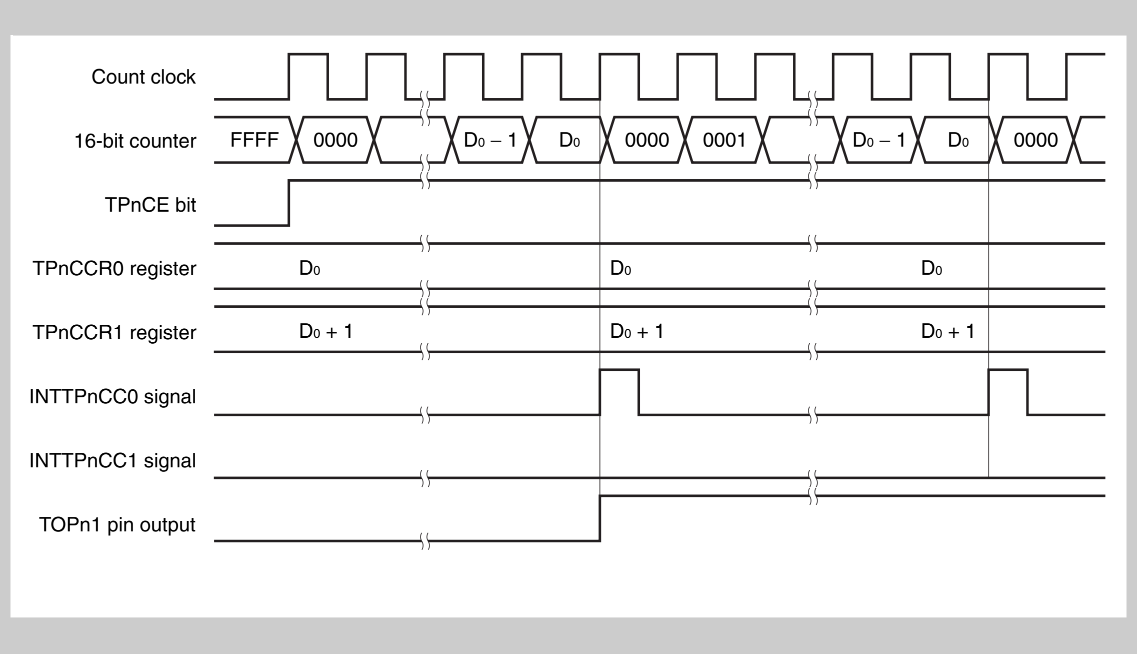
To write the TPnCCR0 or TPnCCR1 register again after writing the TPnCCR1 register once, do so after the INTTPnCC0 signal is generated. Otherwise, the value of the CCRm buffer register may become undefined because the timing of transferring data from the TPnCCRm register to the CCRm buffer register conflicts with writing the TPnCCRm register.

(b) 0%/100% output of PWM waveform

To output a 0% waveform, set the TPnCCR1 register to 0000H. If the set value of the TPnCCR0 register is FFFFH, the INTTPnCC1 signal is generated periodically.

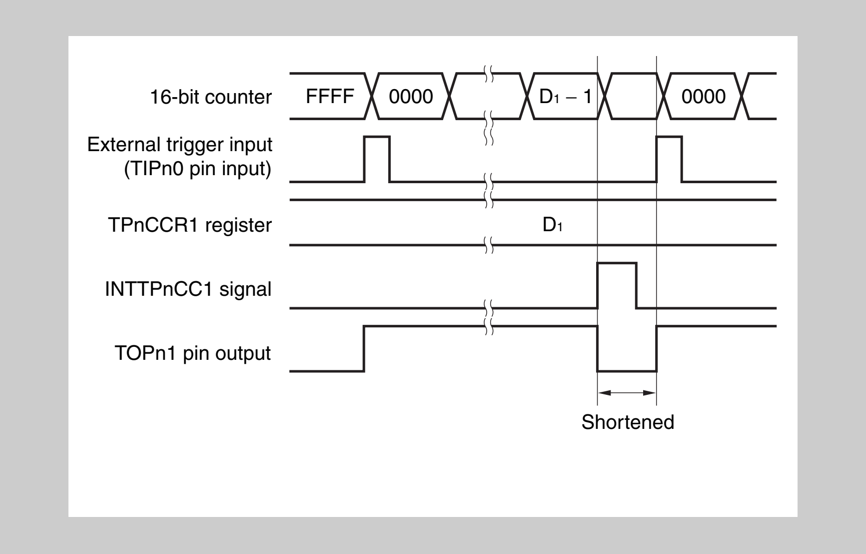


To output a 100% waveform, set a value of (set value of TPnCCR0 register + 1) to the TPnCCR1 register. If the set value of the TPnCCR0 register is FFFFH, 100% output cannot be produced.

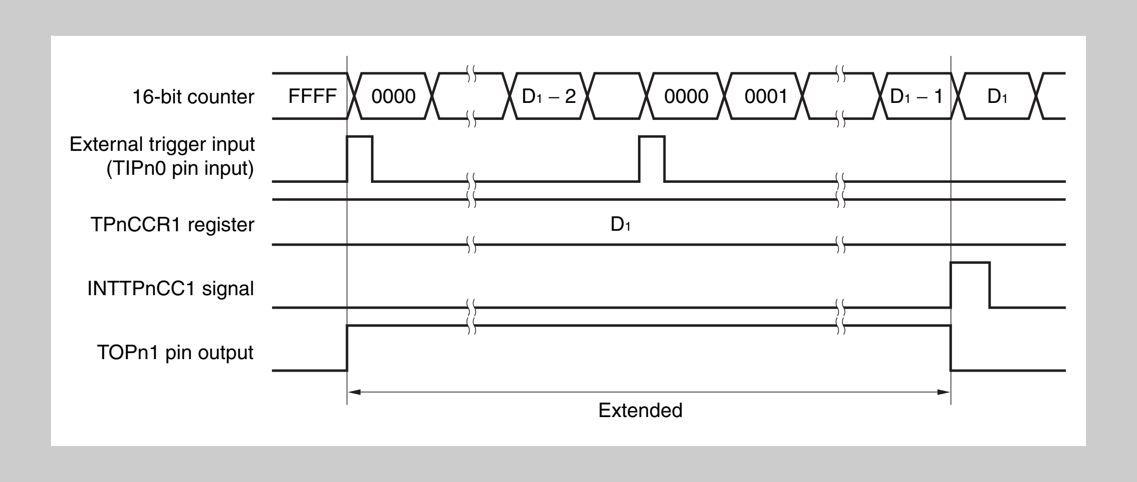


(c) Conflict between trigger detection and match with TPnCCR1 register

If the trigger is detected immediately after the INTTPnCC1 signal is generated, the 16-bit counter is immediately cleared to 0000H, the output signal of the TOPn1 pin is asserted, and the counter continues counting. Consequently, the inactive period of the PWM waveform is shortened.

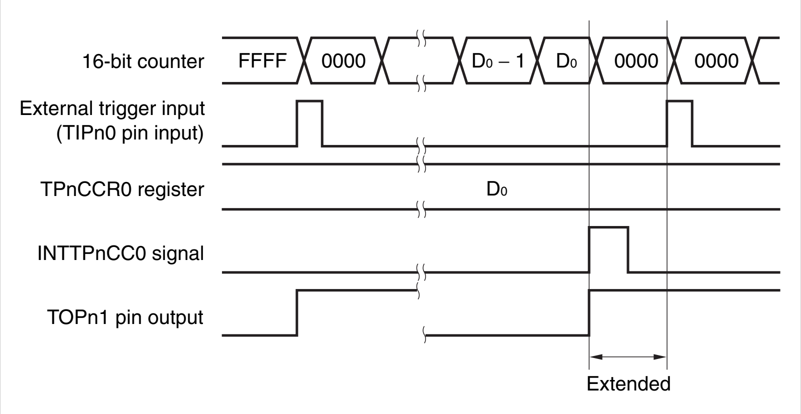


If the trigger is detected immediately before the INTTPnCC1 signal is generated, the INTTPnCC1 signal is not generated, and the 16-bit counter is cleared to 0000H and continues counting. The output signal of the TOPn1 pin remains active. Consequently, the active period of the PWM waveform is extended.

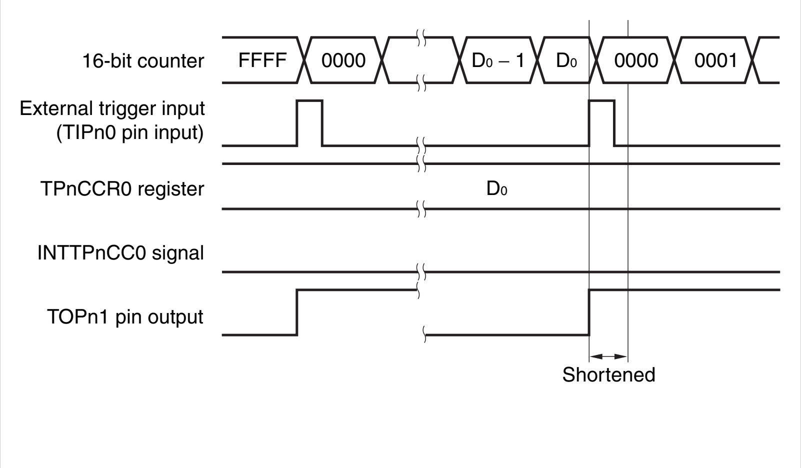


(d) Conflict between trigger detection and match with TPnCCR0 register

If the trigger is detected immediately after the INTTPnCC0 signal is generated, the 16-bit counter is cleared to 0000H and continues counting up. Therefore, the active period of the TOPn1 pin is extended by time from generation of the INTTPnCC0 signal to trigger detection.

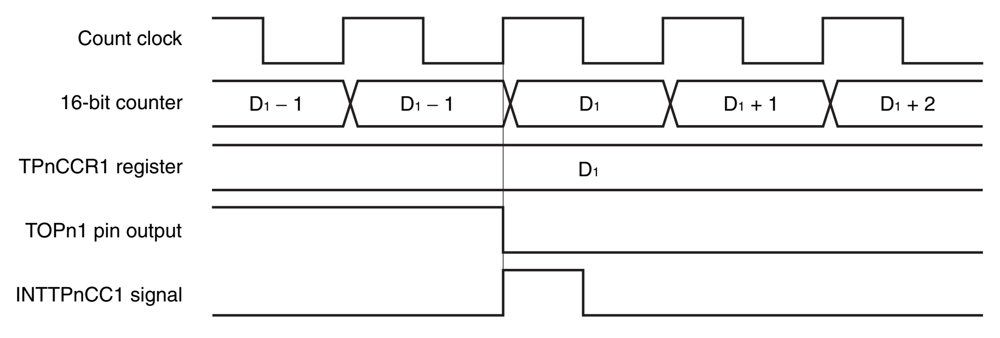


If the trigger is detected immediately before the INTTPnCC0 signal is generated, the INTTPnCC0 signal is not generated. The 16-bit counter is cleared to 0000H, the TOPn1 pin is asserted, and the counter continues counting. Consequently, the inactive period of the PWM waveform is shortened.



(e) Generation timing of compare match interrupt request signal (INTTPnCC1)

The timing of generation of the INTTPnCC1 signal in the external trigger pulse output mode differs from the timing of other INTTPnCC1 signals; the INTTPnCC1 signal is generated when the count value of the 16-bit counter matches the value of the TPnCCR1 register.



Usually, the INTTPnCC1 signal is generated in synchronization with the next count up, after the count value of the 16-bit counter matches the value of the TPnCCR1 register.

In the external trigger pulse output mode, however, it is generated one clock earlier. This is because the timing is changed to match the timing of changing the output signal of the TOPn1 pin.

11.5.4 One-shot pulse output mode (TPnMD2 to TPnMD0 = 011)

In the one-shot pulse output mode, 16-bit timer/event counter P waits for a trigger when the TPnCTL0.TPnCE bit is set to 1. When the valid edge of an external trigger input is detected, 16-bit timer/event counter P starts counting, and outputs a one-shot pulse from the TOPn1 pin.

Instead of the external trigger, a software trigger can also be generated to output the pulse. When the software trigger is used, the TOPn0 pin outputs the active level while the 16-bit counter is counting, and the inactive level when the counter is stopped (waiting for a trigger).

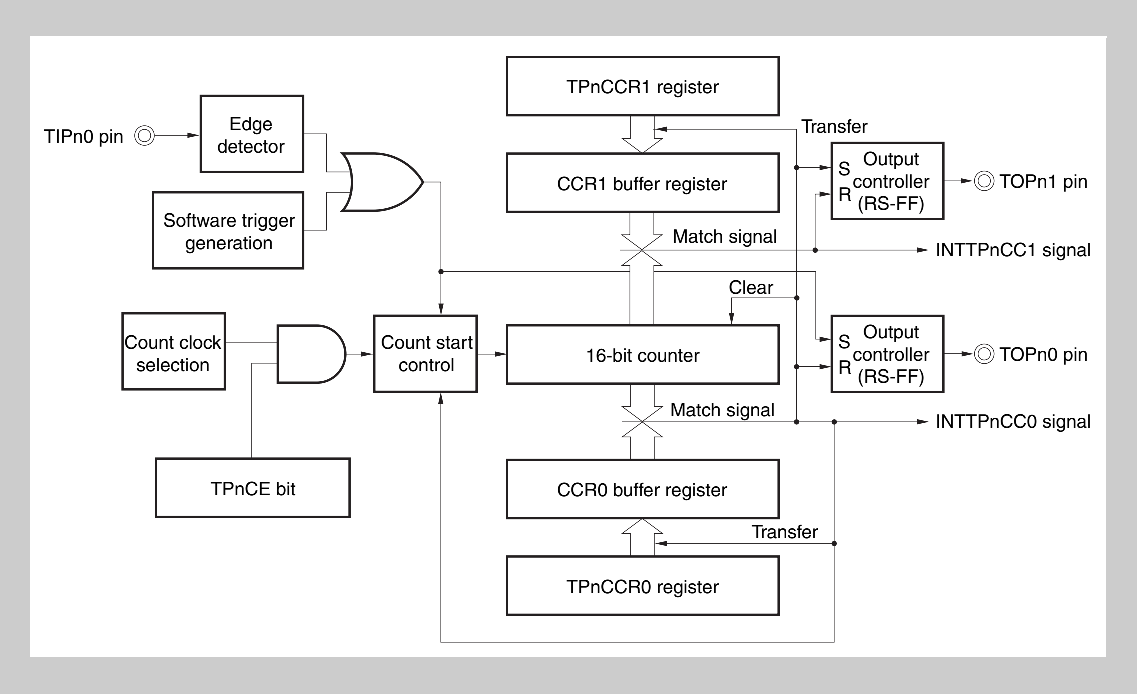


Figure 11-18 Configuration in one-shot pulse output mode

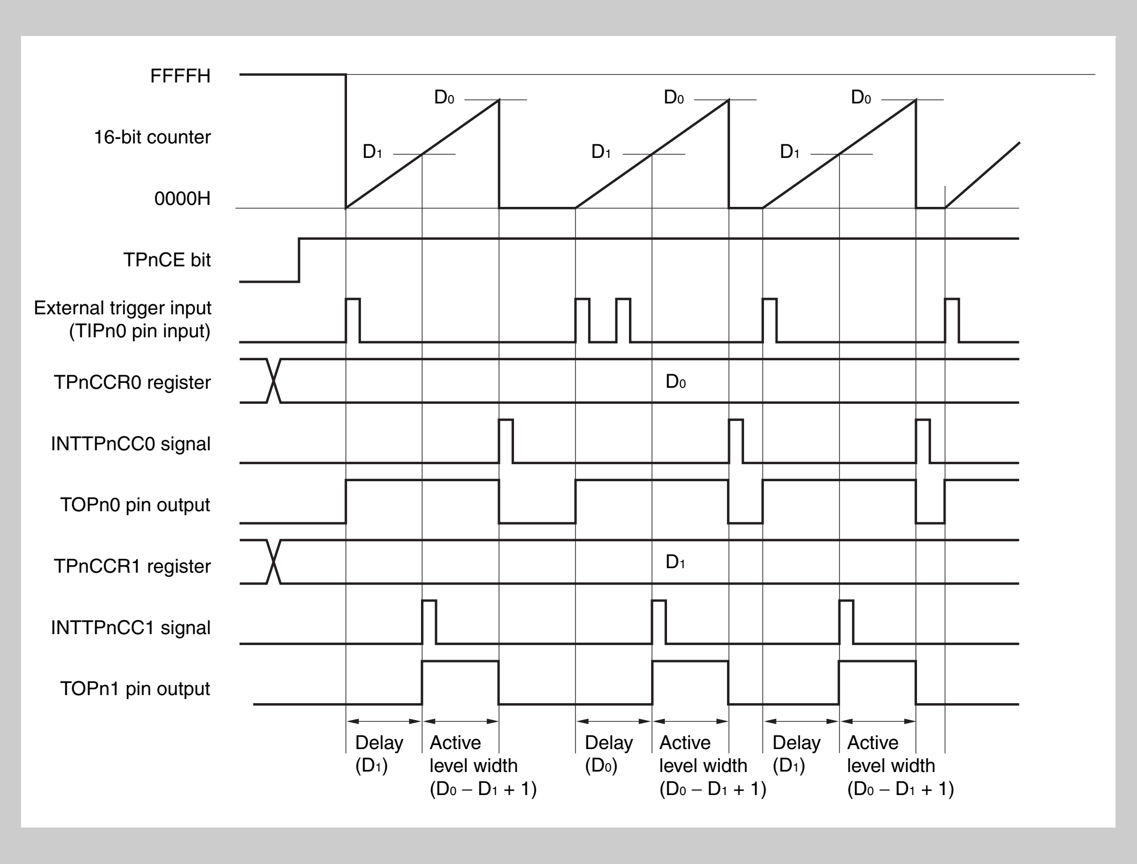


Figure 11-19 Basic timing in one-shot pulse output mode

When the TPnCE bit is set to 1, 16-bit timer/event counter P waits for a trigger. When the trigger is generated, the 16-bit counter is cleared from FFFFH to 0000H, starts counting, and outputs a one-shot pulse from the TOPn1 pin. After the one-shot pulse is output, the 16-bit counter is set to FFFFH, stops counting, and waits for a trigger. If a trigger is generated again while the one-shot pulse is being output, it is ignored.

The output delay period and active level width of the one-shot pulse can be calculated as follows.

Output delay period = (Set value of TPnCCR1 register) × Count clock cycle

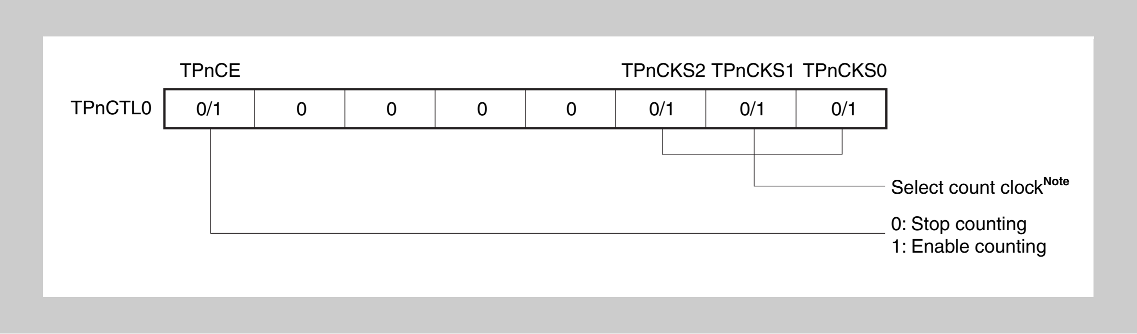
Active level width =
(Set value of TPnCCR0 register – Set value of TPnCCR1 register + 1) × Count clock cycle

The compare match interrupt request signal INTTPnCC0 is generated when the 16-bit counter counts after its count value matches the value of the CCR0 buffer register. The compare match interrupt request signal INTTPnCC1 is generated when the count value of the 16-bit counter matches the value of the CCR1 buffer register.

The valid edge of an external trigger input or setting the software trigger (TPnCTL1.TPnEST bit) to 1 is used as the trigger.

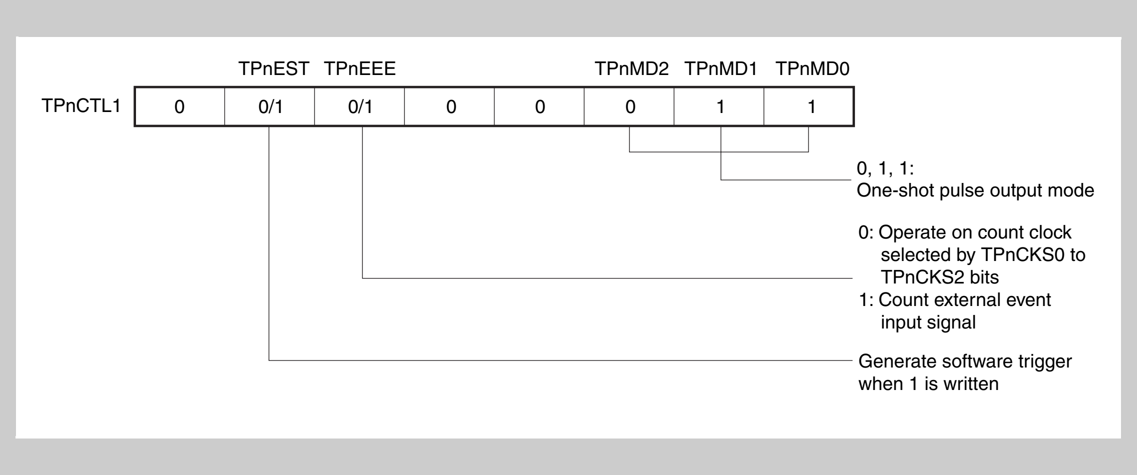
(1) Setting of registers in one-shot pulse output mode

(a) TMPn control register 0 (TPnCTL0)

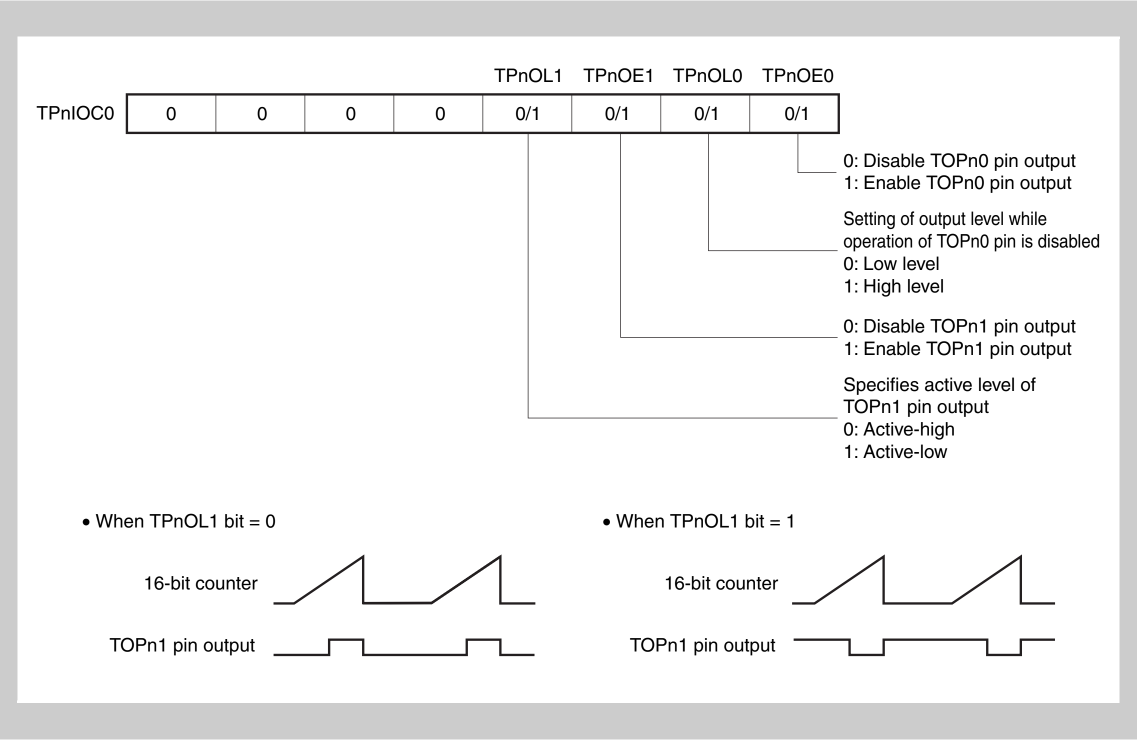


Note The setting is invalid when the TPnCTL1.TPnEEE bit = 1.

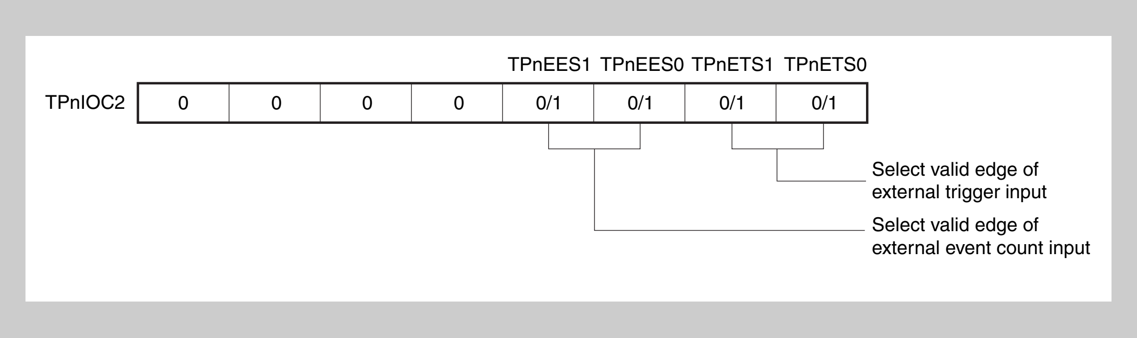
(b) TMPn control register 1 (TPnCTL1)



(c) TMPn I/O control register 0 (TPnIOC0)



(d) TMPn I/O control register 2 (TPnIOC2)



(e) TMPn counter read buffer register (TPnCNT)

The value of the 16-bit counter can be read by reading the TPnCNT register.

(f) TMPn capture/compare registers 0 and 1 (TPnCCR0 and TPnCCR1)

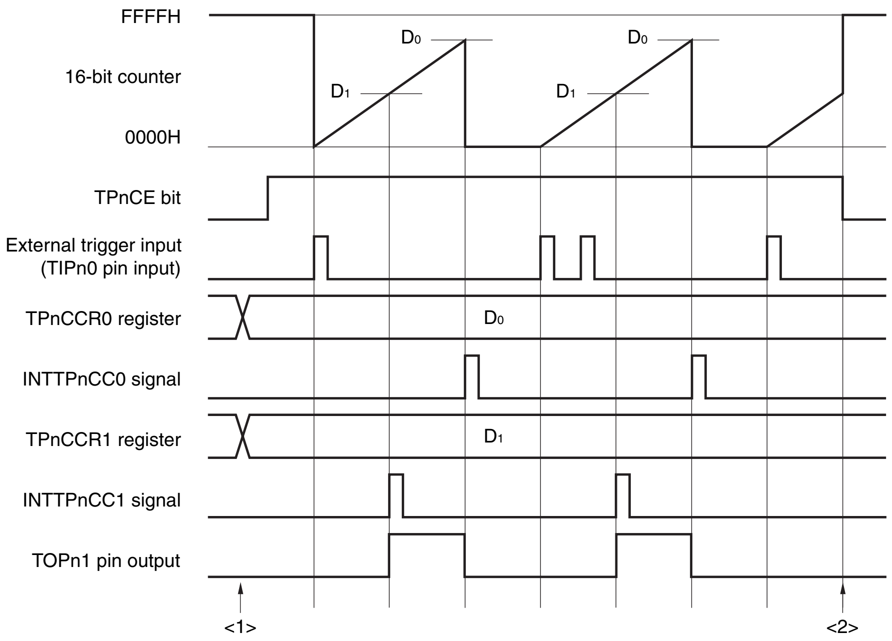
If D_0 is set to the TPnCCR0 register and D_1 to the TPnCCR1 register, the active level width and output delay period of the one-shot pulse are as follows.

Active level width = $(D_1 - D_0 + 1) \times$ Count clock cycle

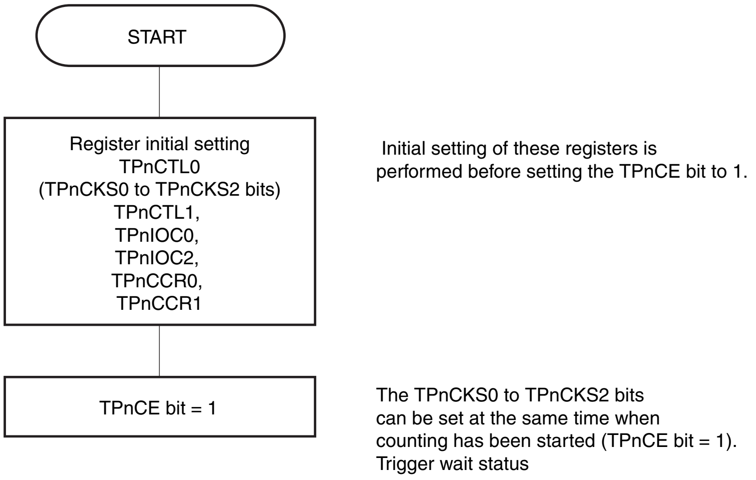
Output delay period = $D_1 \times$ Count clock cycle

Note TMPn I/O control register 1 (TPnIOC1) and TMPn option register 0 (TPnOPT0) are not used in the one-shot pulse output mode.

(2) Operation flow in one-shot pulse output mode



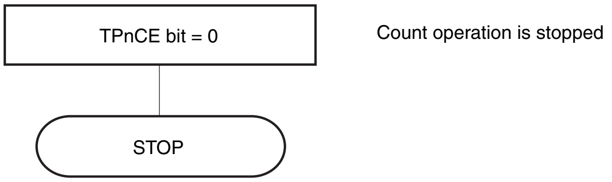
<1> Count operation start flow



Initial setting of these registers is performed before setting the TPnCE bit to 1.

The TPnCKS0 to TPnCKS2 bits can be set at the same time when counting has been started (TPnCE bit = 1). Trigger wait status

<2> Count operation stop flow



Count operation is stopped

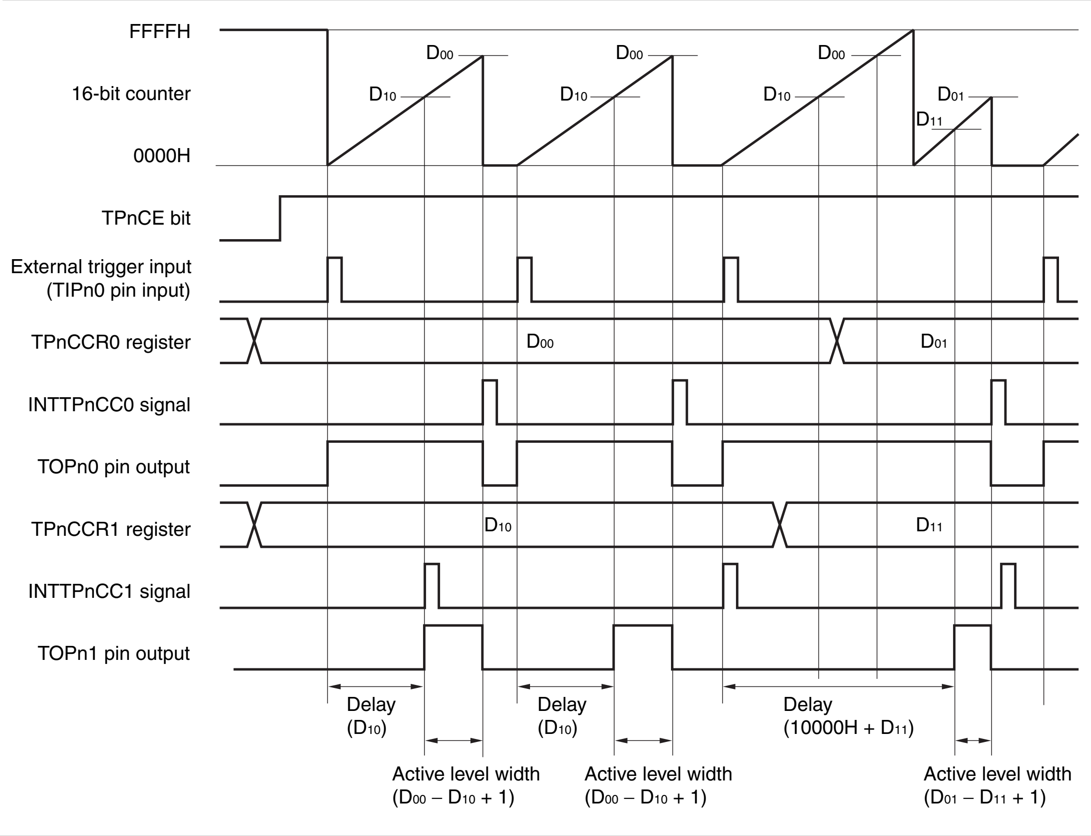
Figure 11-20 Software processing flow in one-shot pulse output mode

(3) Operation timing in one-shot pulse output mode

(a) Note on rewriting TPnCCRm register

To change the set value of the TPnCCRm register to a smaller value, stop counting once, and then change the set value.

If the value of the TPnCCRm register is rewritten to a smaller value during counting, the 16-bit counter may overflow.

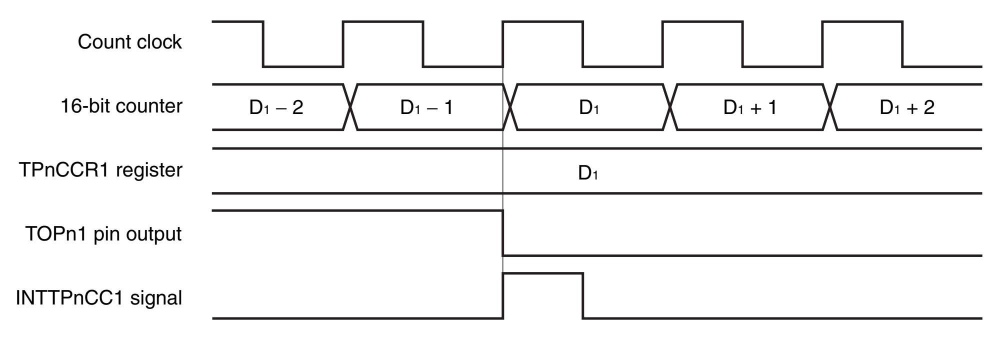


When the TPnCCR0 register is rewritten from D_{00} to D_{01} and the TPnCCR1 register from D_{10} to D_{11} where $D_{00} > D_{01}$ and $D_{10} > D_{11}$, if the TPnCCR1 register is rewritten when the count value of the 16-bit counter is greater than D_{11} and less than D_{10} and if the TPnCCR0 register is rewritten when the count value is greater than D_{01} and less than D_{00}, each set value is reflected as soon as the register has been rewritten and compared with the count value. The counter counts up to FFFFH and then counts up again from 0000H. When the count value matches D_{11}, the counter generates the INTTPnCC1 signal and asserts the TOPn1 pin. When the count value matches D_{01}, the counter generates the INTTPnCC0 signal, deasserts the TOPn1 pin, and stops counting.

Therefore, the counter may output a pulse with a delay period or active period different from that of the one-shot pulse that is originally expected.

(b) Generation timing of compare match interrupt request signal (INTTPnCC1)

The generation timing of the INTTPnCC1 signal in the one-shot pulse output mode is different from other INTTPnCC1 signals; the INTTPnCC1 signal is generated when the count value of the 16-bit counter matches the value of the TPnCCR1 register.



Usually, the INTTPnCC1 signal is generated when the 16-bit counter counts up next time after its count value matches the value of the TPnCCR1 register.

In the one-shot pulse output mode, however, it is generated one clock earlier. This is because the timing is changed to match the change timing of the TOPn1 pin.

11.5.5 PWM output mode (TPnMD2 to TPnMD0 = 100)

In the PWM output mode, a PWM waveform is output from the TOPn1 pin when the TPnCTL0.TPnCE bit is set to 1.

In addition, a pulse with one cycle of the PWM waveform as half its cycle is output from the TOPn0 pin.

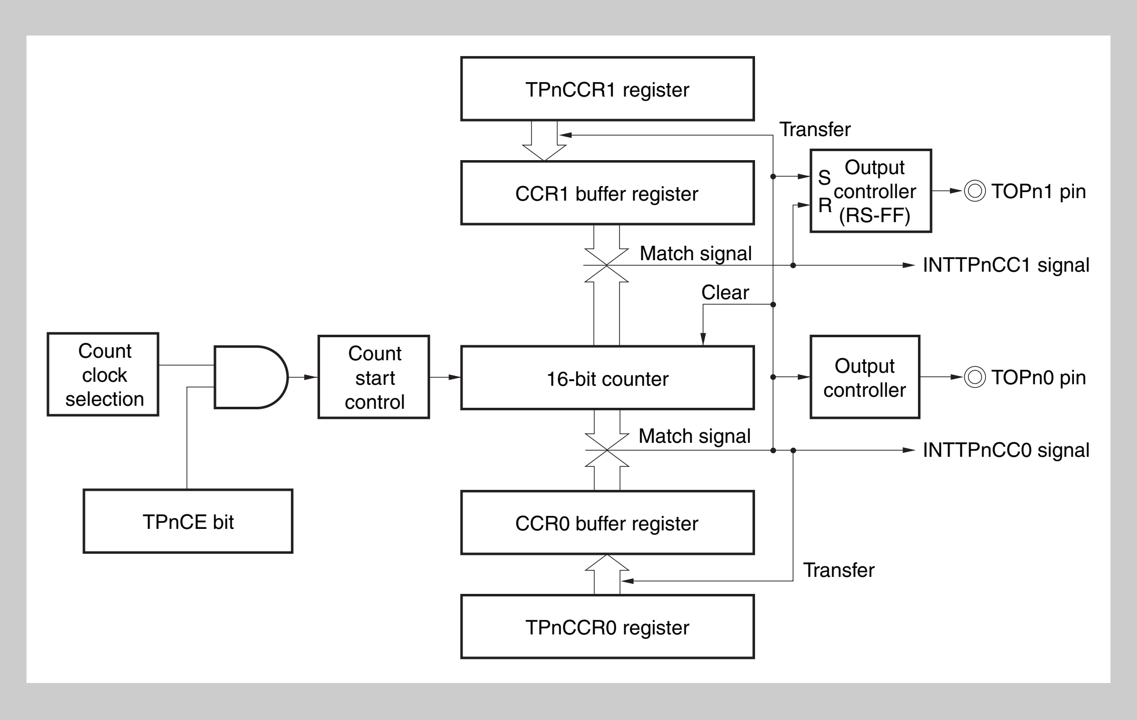


Figure 11-21 Configuration in PWM output mode

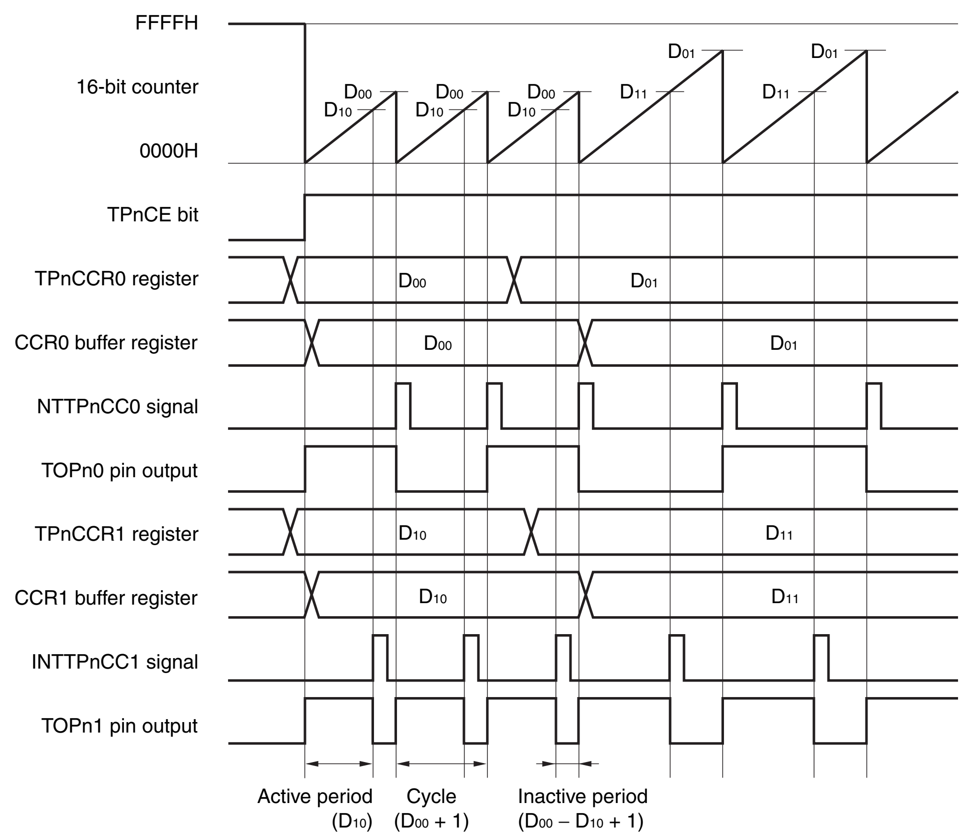


Figure 11-22 Basic timing in PWM output mode

When the TPnCE bit is set to 1, the 16-bit counter is cleared from FFFFH to 0000H, starts counting, and outputs a PWM waveform from the TOPn1 pin.

The active level width, cycle, and duty factor of the PWM waveform can be calculated as follows.

Active level width = (Set value of TPnCCR1 register) × Count clock cycle

Cycle = (Set value of TPnCCR0 register + 1) × Count clock cycle

Duty factor = (Set value of TPnCCR1 register)/(Set value of TPnCCR0 register + 1)

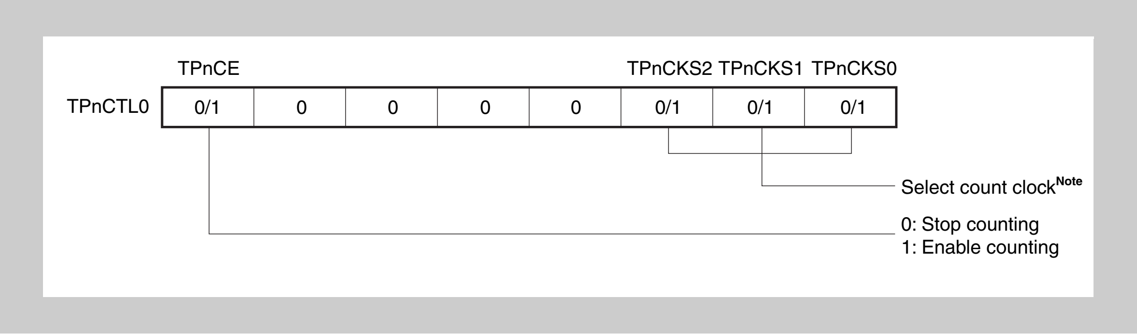
The PWM waveform can be changed by rewriting the TPnCCRm register while the counter is operating. The newly written value is reflected when the count value of the 16-bit counter matches the value of the CCR0 buffer register and the 16-bit counter is cleared to 0000H.

The compare match interrupt request signal INTTPnCC0 is generated when the 16-bit counter counts next time after its count value matches the value of the CCR0 buffer register, and the 16-bit counter is cleared to 0000H. The compare match interrupt request signal INTTPnCC1 is generated when the count value of the 16-bit counter matches the value of the CCR1 buffer register.

The value set to the TPnCCRm register is transferred to the CCRm buffer register when the count value of the 16-bit counter matches the value of the CCRm buffer register and the 16-bit counter is cleared to 0000H.

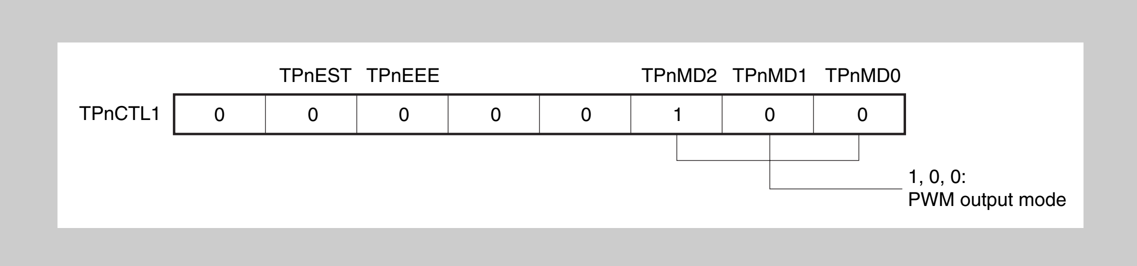
(1) Setting of registers in PWM output mode

(a) TMPn control register 0 (TPnCTL0)

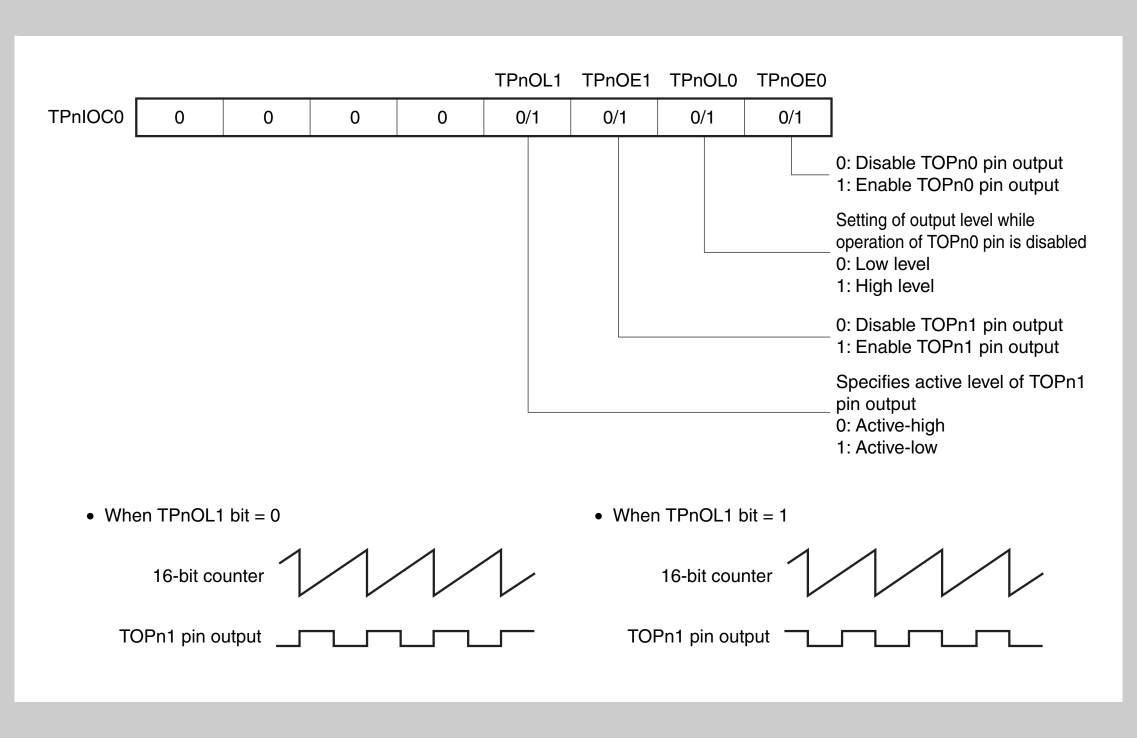


Note The setting is invalid when the TPnCTL1.TPnEEE bit = 1.

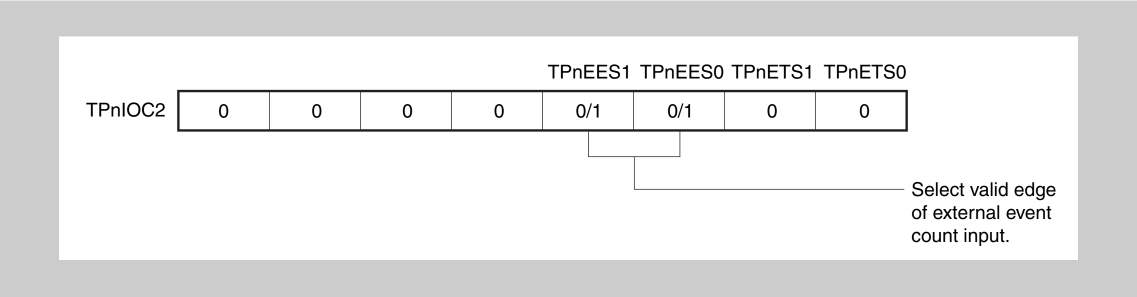
(b) TMPn control register 1 (TPnCTL1)



(c) TMPn I/O control register 0 (TPnIOC0)



(d) TMPn I/O control register 2 (TPnIOC2)



(e) TMPn counter read buffer register (TPnCNT)

The value of the 16-bit counter can be read by reading the TPnCNT register.

(f) TMPn capture/compare registers 0 and 1 (TPnCCR0 and TPnCCR1)

If D_0 is set to the TPnCCR0 register and D_1 to the TPnCCR1 register, the cycle and active level of the PWM waveform are as follows.

Cycle = $(D_0 + 1) \times$ Count clock cycle

Active level width = $D_1 \times$ Count clock cycle

Note TMPn I/O control register 1 (TPnIOC1) and TMPn option register 0 (TPnOPT0) are not used in the PWM output mode.

(2) Operation flow in PWM output mode

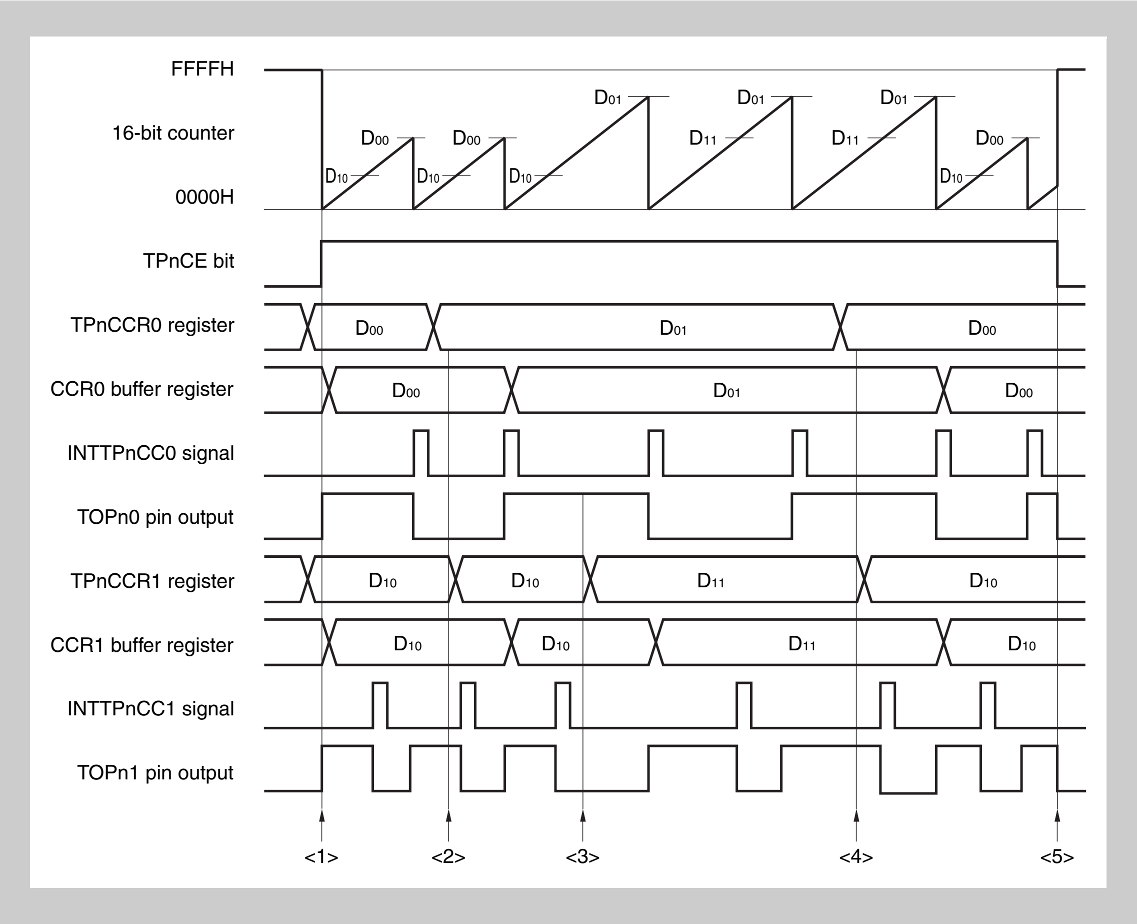
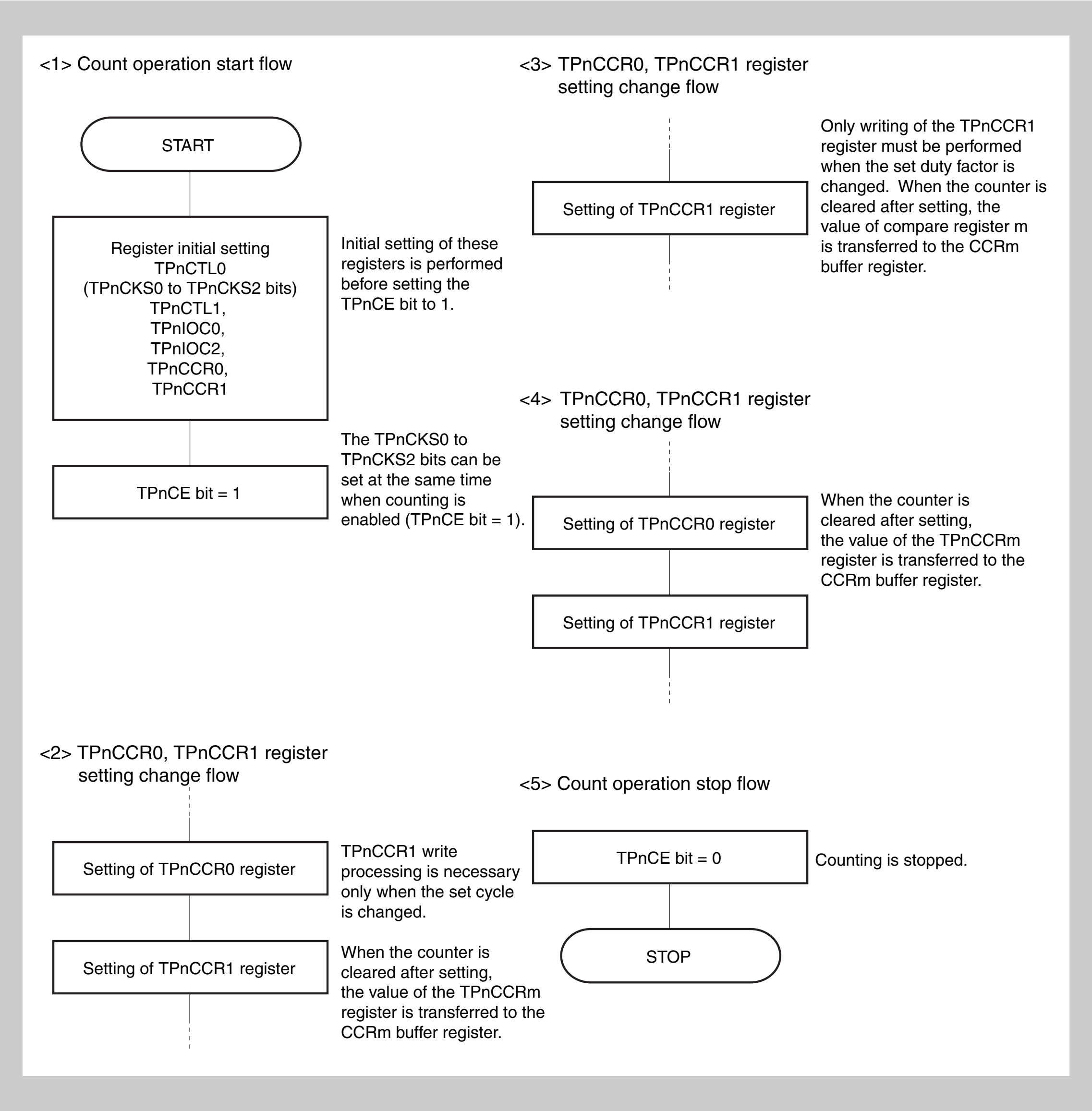


Figure 11-23 Software processing flow in PWM output mode (1/2)



Initial setting of these registers is performed before setting the TPnCE bit to 1.

The TPnCKS0 to TPnCKS2 bits can be set at the same time when counting is enabled (TPnCE bit = 1).

TPnCCR1 write processing is necessary only when the set cycle is changed.

When the counter is cleared after setting, the value of the TPnCCRm register is transferred to the CCRm buffer register.

Only writing of the TPnCCR1 register must be performed when the set duty factor is changed. When the counter is cleared after setting, the value of compare register m is transferred to the CCRm buffer register.

When the counter is cleared after setting, the value of the TPnCCRm register is transferred to the CCRm buffer register.

Counting is stopped.

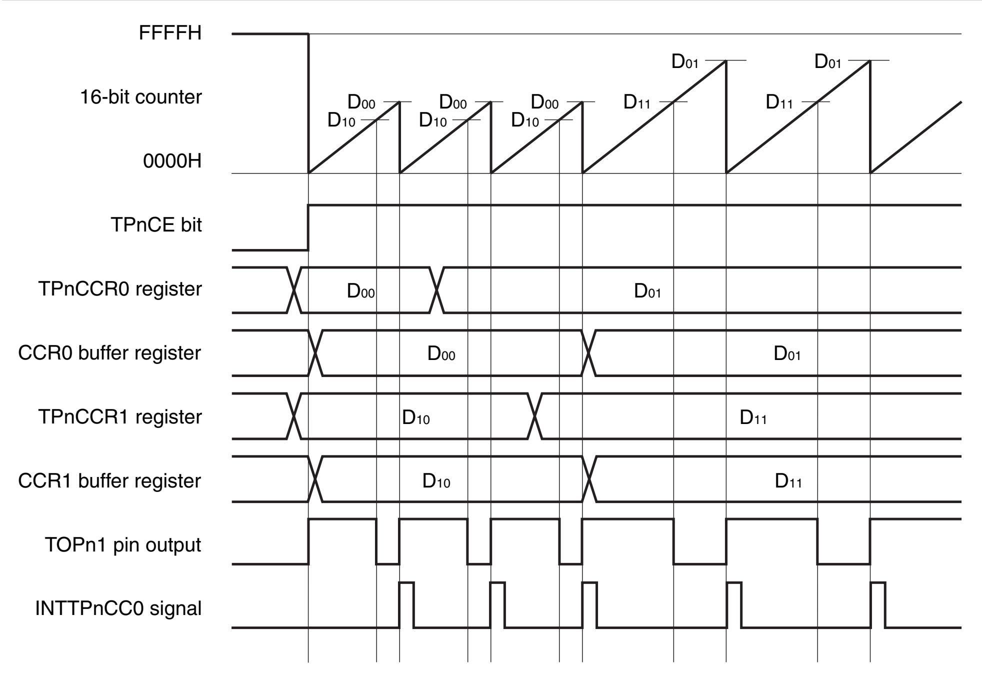
Figure 11-24 Software processing flow in PWM output mode (1/2)

(3) PWM output mode operation timing

(a) Changing pulse width during operation

To change the PWM waveform while the counter is operating, write the TPnCCR1 register last.

Rewrite the TPnCCRm register after writing the TPnCCR1 register after the INTTPnCC1 signal is detected.



To transfer data from the TPnCCRm register to the CCRm buffer register, the TPnCCR1 register must be written.

To change both the cycle and active level of the PWM waveform at this time, first set the cycle to the TPnCCR0 register and then set the active level to the TPnCCR1 register.

To change only the cycle of the PWM waveform, first set the cycle to the TPnCCR0 register, and then write the same value to the TPnCCR1 register.

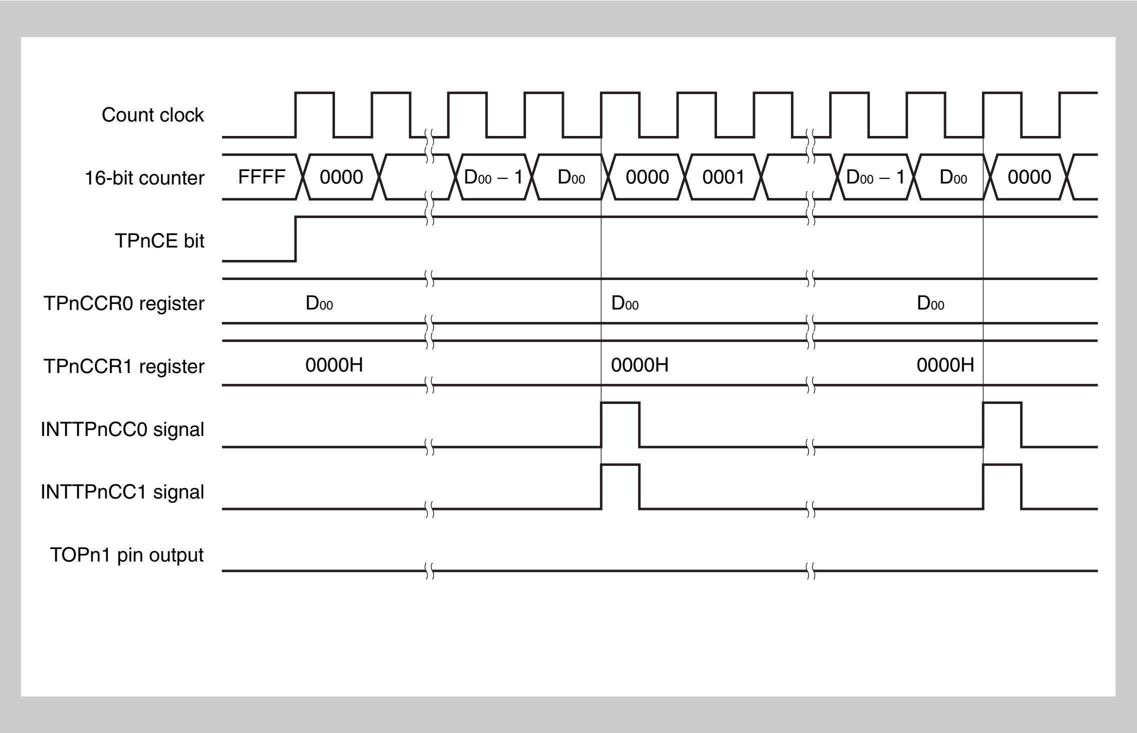
To change only the active level width (duty factor) of the PWM waveform, only the TPnCCR1 register has to be set.

After data is written to the TPnCCR1 register, the value written to the TPnCCRm register is transferred to the CCRm buffer register in synchronization with clearing of the 16-bit counter, and is used as the value compared with the 16-bit counter.

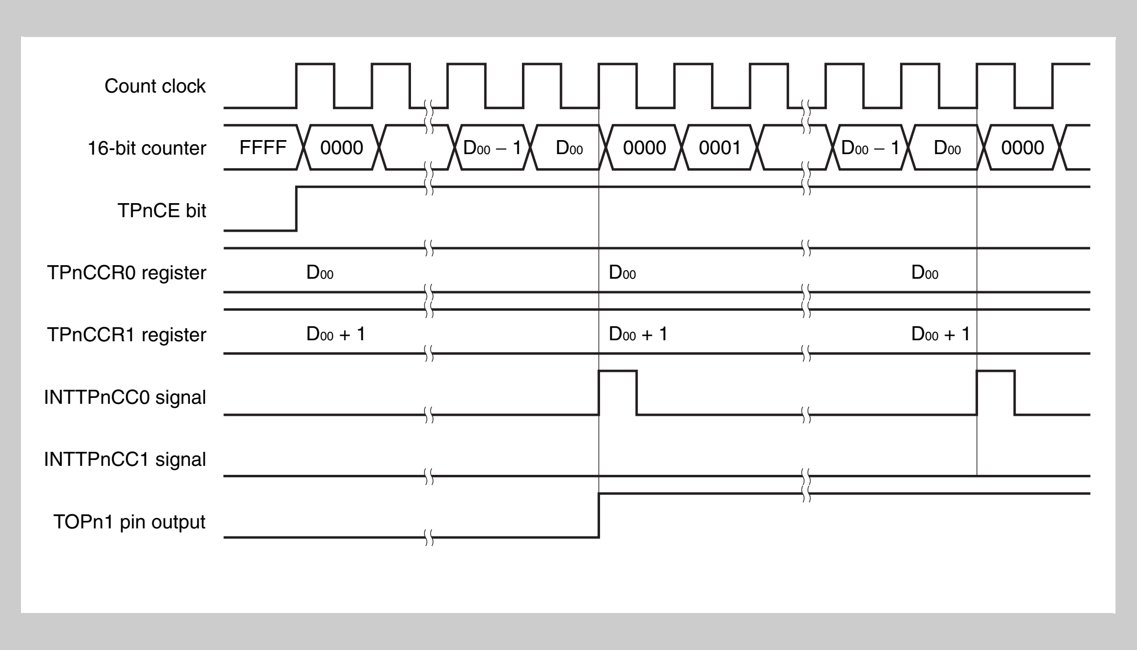
To write the TPnCCR0 or TPnCCR1 register again after writing the TPnCCR1 register once, do so after the INTTPnCC0 signal is generated. Otherwise, the value of the CCRm buffer register may become undefined because the timing of transferring data from the TPnCCRm register to the CCRm buffer register conflicts with writing the TPnCCRm register.

(b) 0%/100% output of PWM waveform

To output a 0% waveform, set the TPnCCR1 register to 0000H. If the set value of the TPnCCR0 register is FFFFH, the INTTPnCC1 signal is generated periodically.

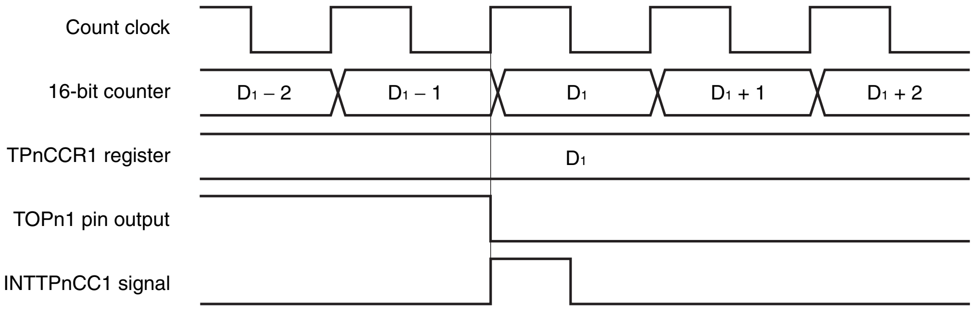


To output a 100% waveform, set a value of (set value of TPnCCR0 register + 1) to the TPnCCR1 register. If the set value of the TPnCCR0 register is FFFFH, 100% output cannot be produced.



(c) Generation timing of compare match interrupt request signal (INTTPnCC1)

The timing of generation of the INTTPnCC1 signal in the PWM output mode differs from the timing of other INTTPnCC1 signals; the INTTPnCC1 signal is generated when the count value of the 16-bit counter matches the value of the TPnCCR1 register.



Usually, the INTTPnCC1 signal is generated in synchronization with the next counting up after the count value of the 16-bit counter matches the value of the TPnCCR1 register.

In the PWM output mode, however, it is generated one clock earlier. This is because the timing is changed to match the change timing of the output signal of the TOPn1 pin.

11.5.6 Free-running timer mode (TPnMD2 to TPnMD0 = 101)

In the free-running timer mode, 16-bit timer/event counter P starts counting when the TPnCTL0.TPnCE bit is set to 1. At this time, the TPnCCRm register can be used as a compare register or a capture register, depending on the setting of the TPnOPT0.TPnCCS0 and TPnOPT0.TPnCCS1 bits.

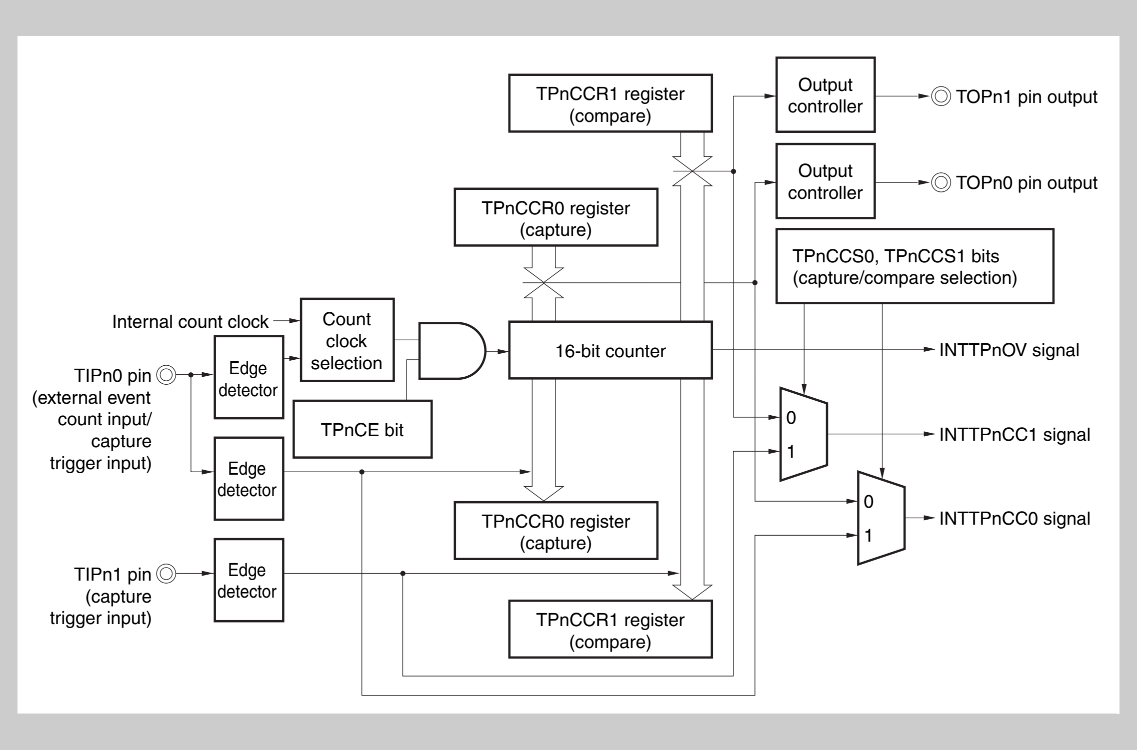


Figure 11-25 Configuration in free-running timer mode

When the TPnCE bit is set to 1, 16-bit timer/event counter P starts counting, and the output signals of the TOPn0 and TOPn1 pins are inverted. When the count value of the 16-bit counter later matches the set value of the TPnCCRm register, a compare match interrupt request signal (INTTPnCCm) is generated, and the output signal of the TOPnm pin is inverted.

The 16-bit counter continues counting in synchronization with the count clock. When it counts up to FFFFH, it generates an overflow interrupt request signal (INTTPnOV) at the next clock, is cleared to 0000H, and continues counting. At this time, the overflow flag (TPnOPT0.TPnOVF bit) is also set to 1. Clear the overflow flag to 0 by executing the CLR instruction by software.

The TPnCCRm register can be rewritten while the counter is operating. If it is rewritten, the new value is reflected at that time, and compared with the count value.

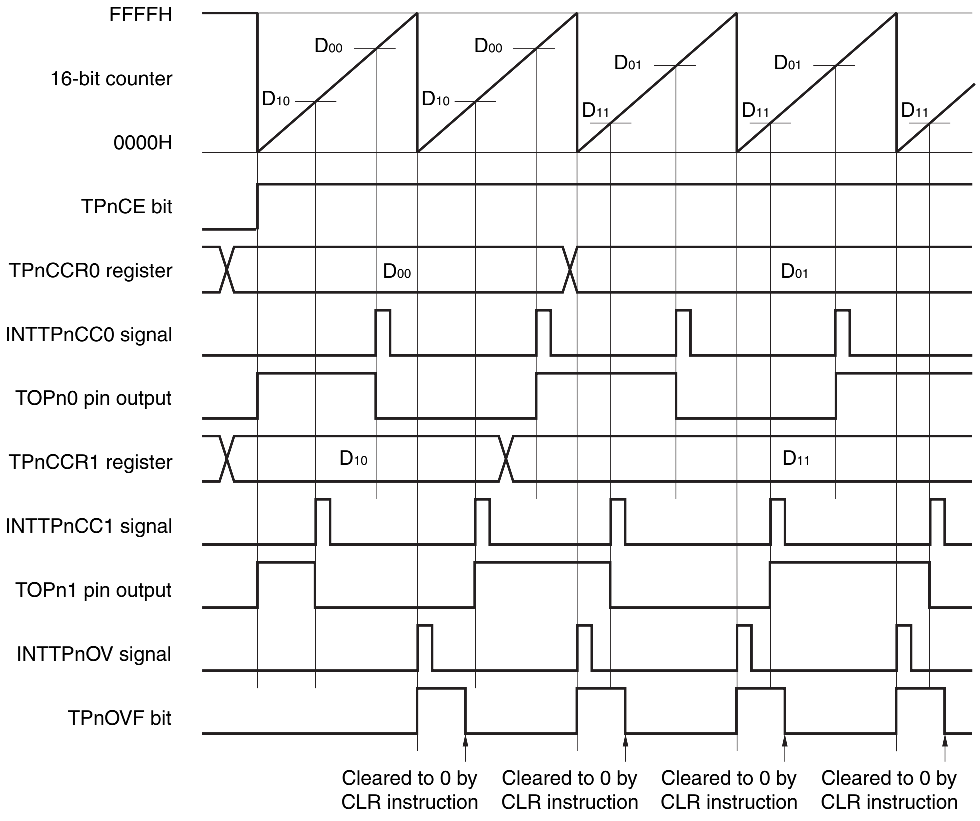


Figure 11-26 Basic timing in free-running timer mode (compare function)

When the TPnCE bit is set to 1, the 16-bit counter starts counting. When the valid edge input to the TIPnm pin is detected, the count value of the 16-bit counter is stored in the TPnCCRm register, and a capture interrupt request signal (INTTPnCCm) is generated.

The 16-bit counter continues counting in synchronization with the count clock. When it counts up to FFFFH, it generates an overflow interrupt request signal (INTTPnOV) at the next clock, is cleared to 0000H, and continues counting. At this time, the overflow flag (TPnOPT0.TPnOVF bit) is also set to 1. Clear the overflow flag to 0 by executing the CLR instruction by software.

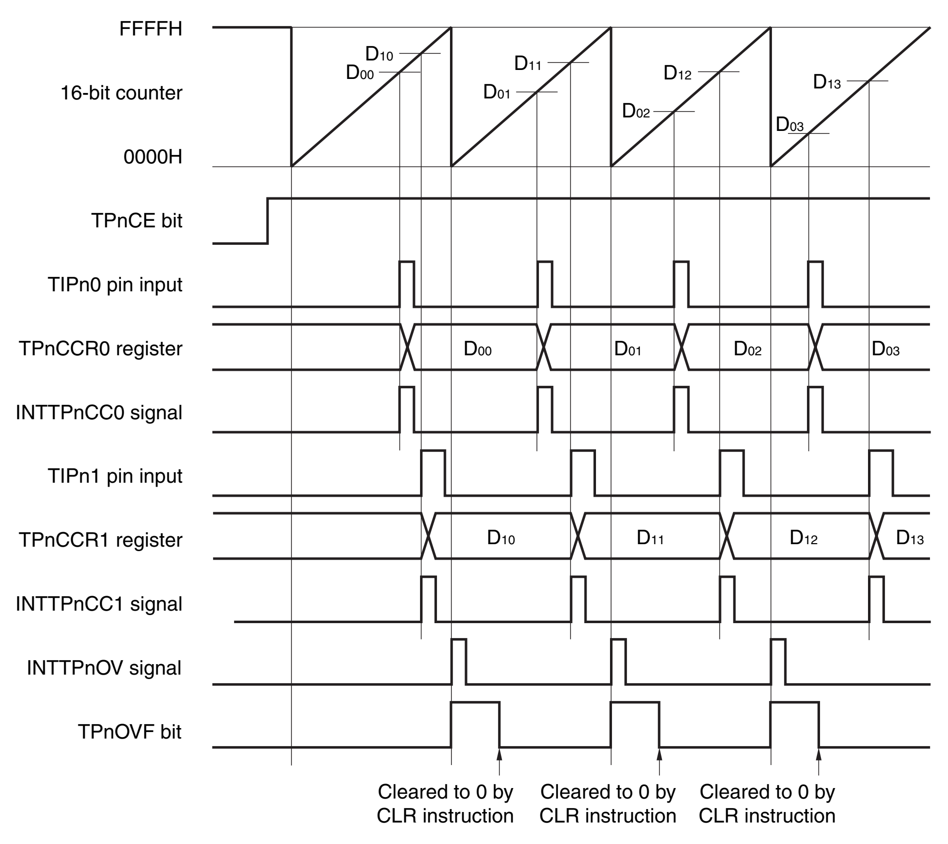
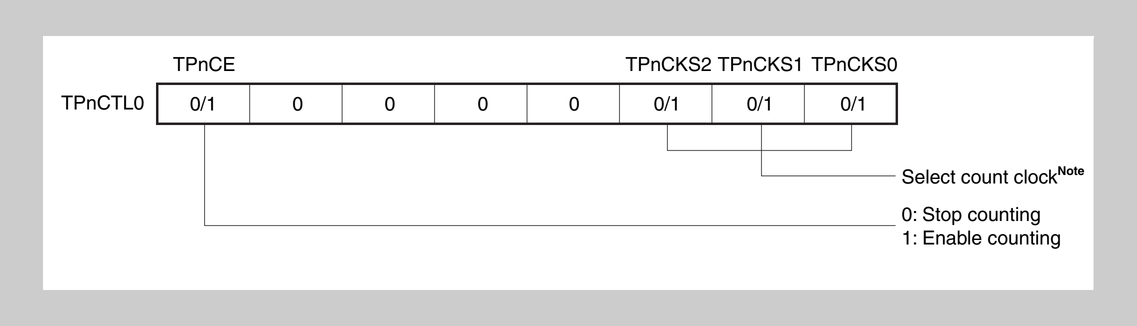


Figure 11-27 Basic timing in free-running timer mode (capture function)

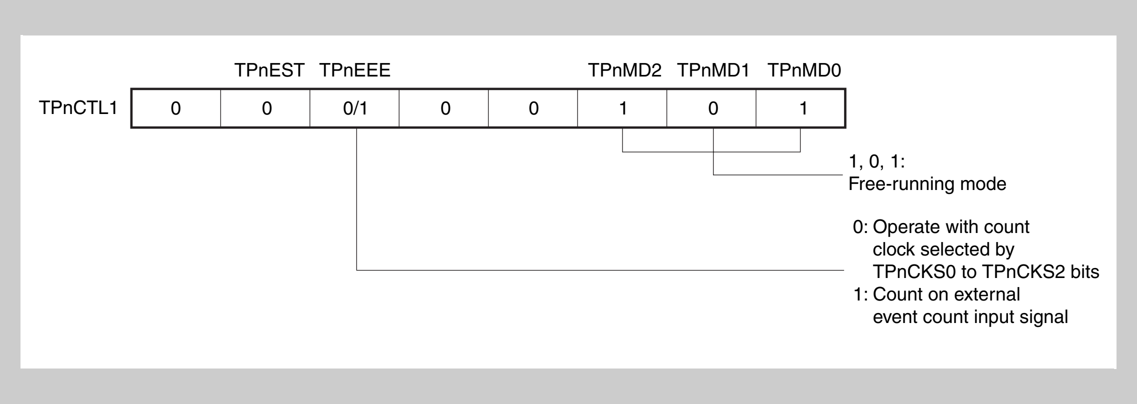
(1) Register setting in free-running timer mode

(a) TMPn control register 0 (TPnCTL0)

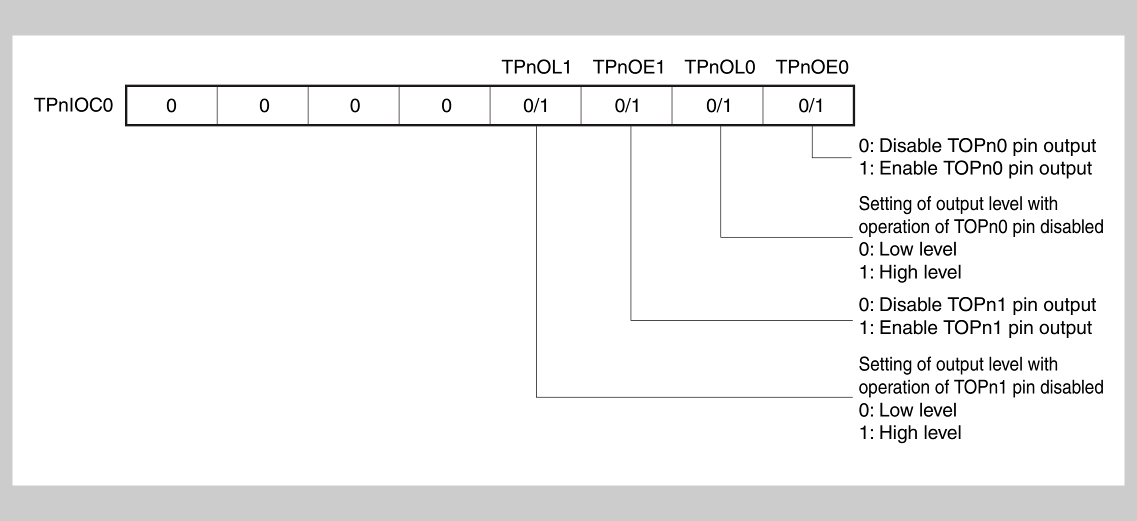


Note The setting is invalid when the TPnCTL1.TPnEEE bit = 1

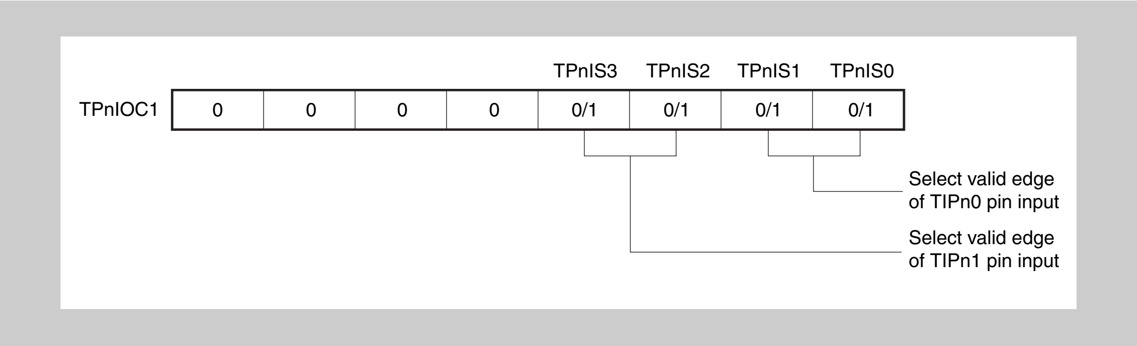
(b) TMPn control register 1 (TPnCTL1)



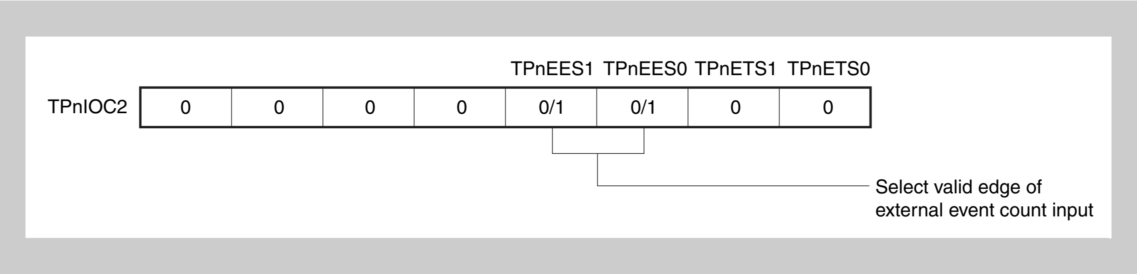
(c) TMPn I/O control register 0 (TPnIOC0)



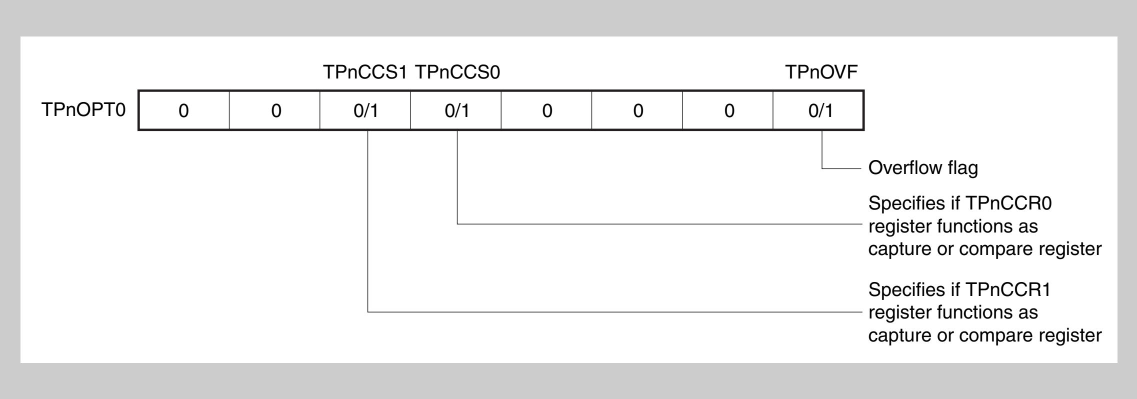
(d) TMPn I/O control register 1 (TPnIOC1)



(e) TMPn I/O control register 2 (TPnIOC2)



(f) TMPn option register 0 (TPnOPT0)



(g) TMPn counter read buffer register (TPnCNT)

The value of the 16-bit counter can be read by reading the TPnCNT register.

(h) TMPn capture/compare registers 0 and 1 (TPnCCR0 and TPnCCR1)

These registers function as capture registers or compare registers depending on the setting of the TPnOPT0.TPnCCSm bit.

When the registers function as capture registers, they store the count value of the 16-bit counter when the valid edge input to the TIPnm pin is detected.

When the registers function as compare registers and when D_m is set to the TPnCCRm register, the INTTPnCCm signal is generated when the counter reaches (D_m + 1), and the output signal of the TOPnm pin is inverted.

(2) Operation flow in free-running timer mode

(a) When using capture/compare register as compare register

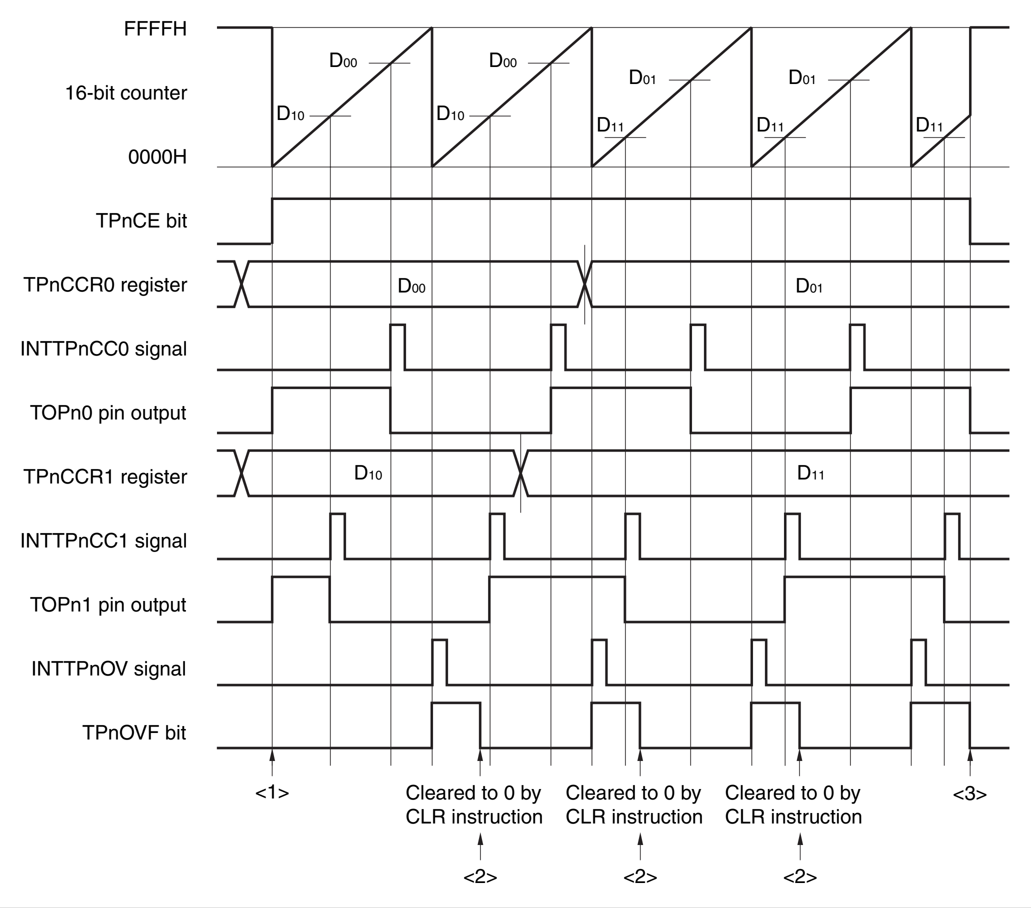


Figure 11-28 Software processing flow in free-running timer mode (compare function) (1/2)

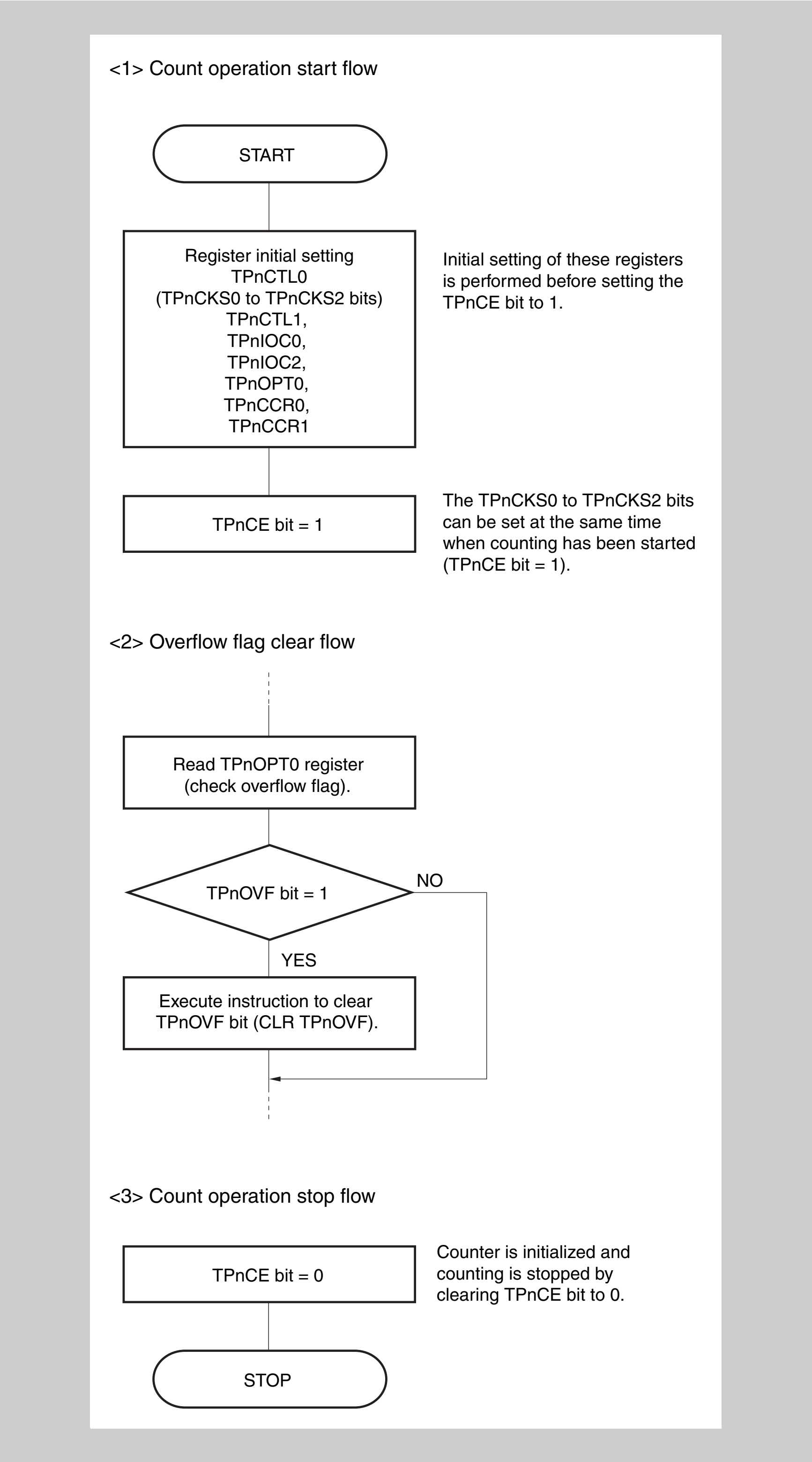


Figure 11-29 **Software processing flow in free-running timer mode (compare function) (2/2)**

(b) When using capture/compare register as capture register

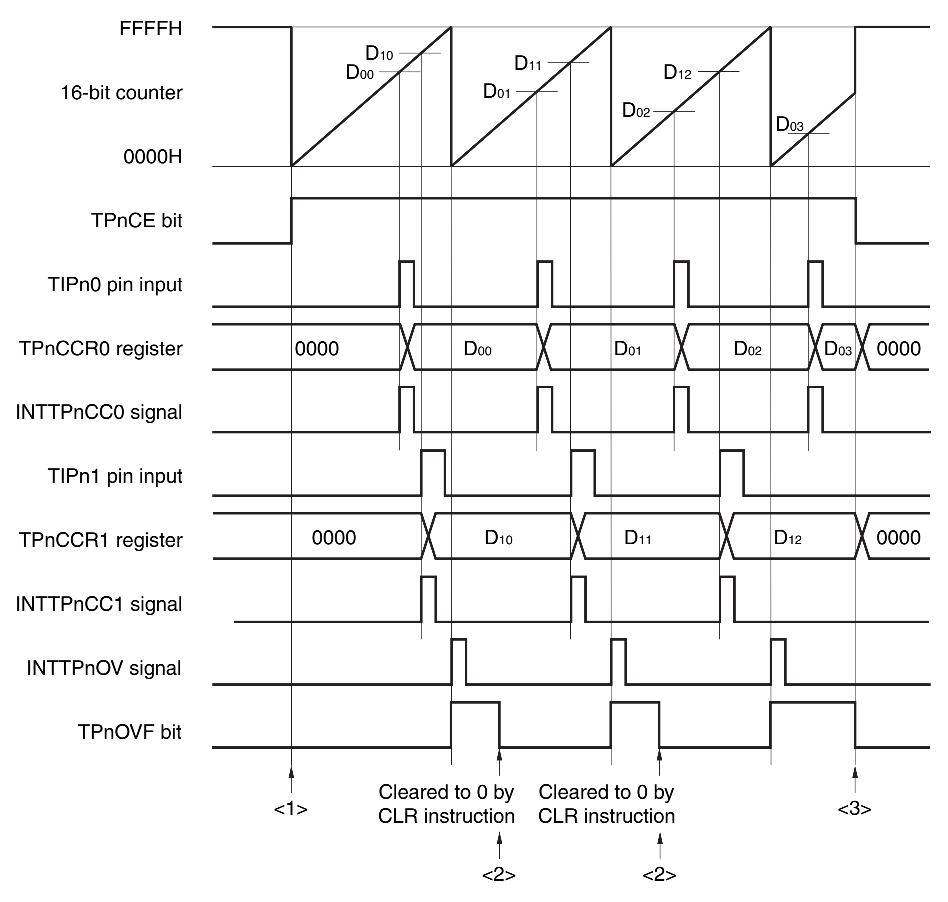


Figure 11-30 Software processing flow in free-running timer mode (capture function) (1/2)

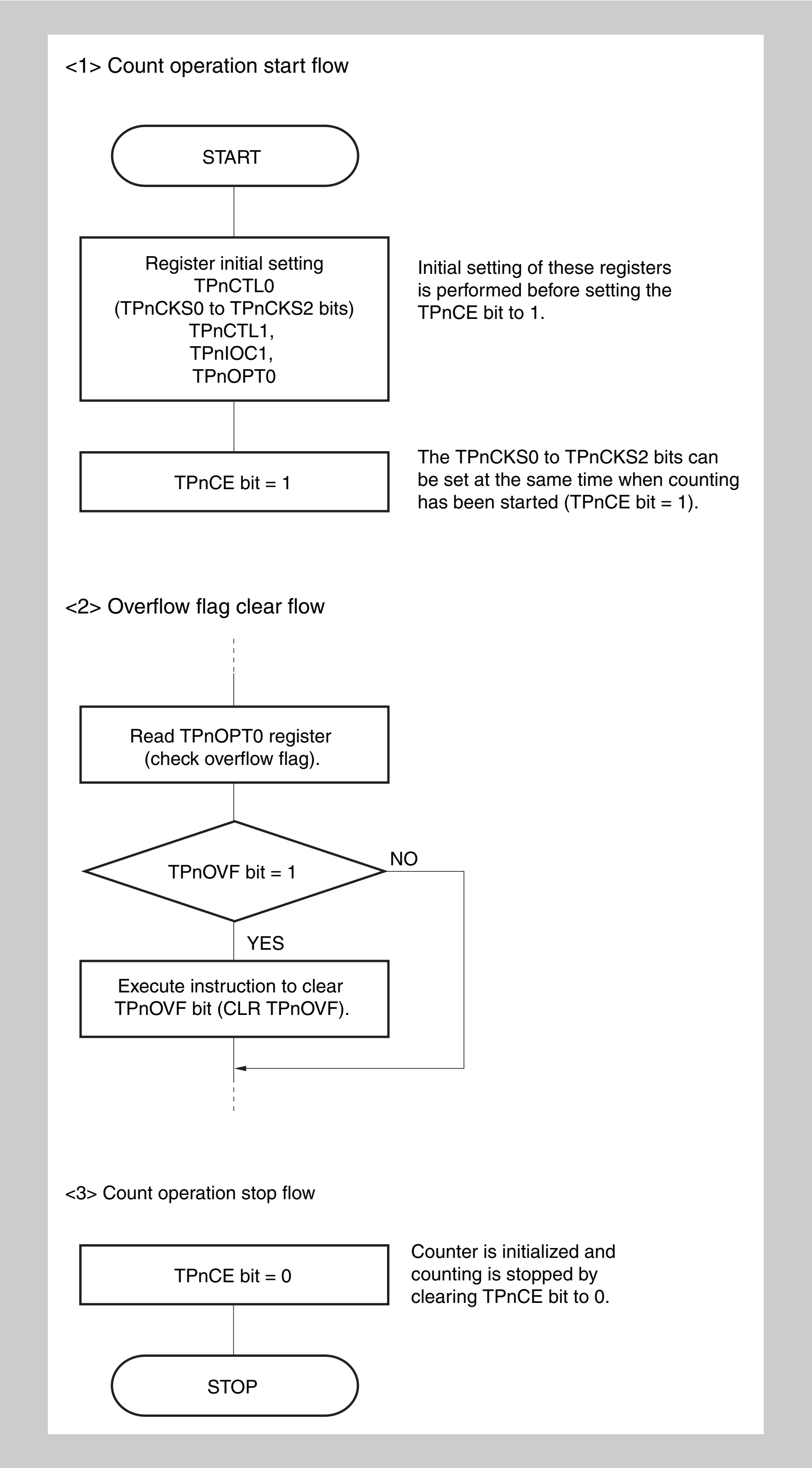
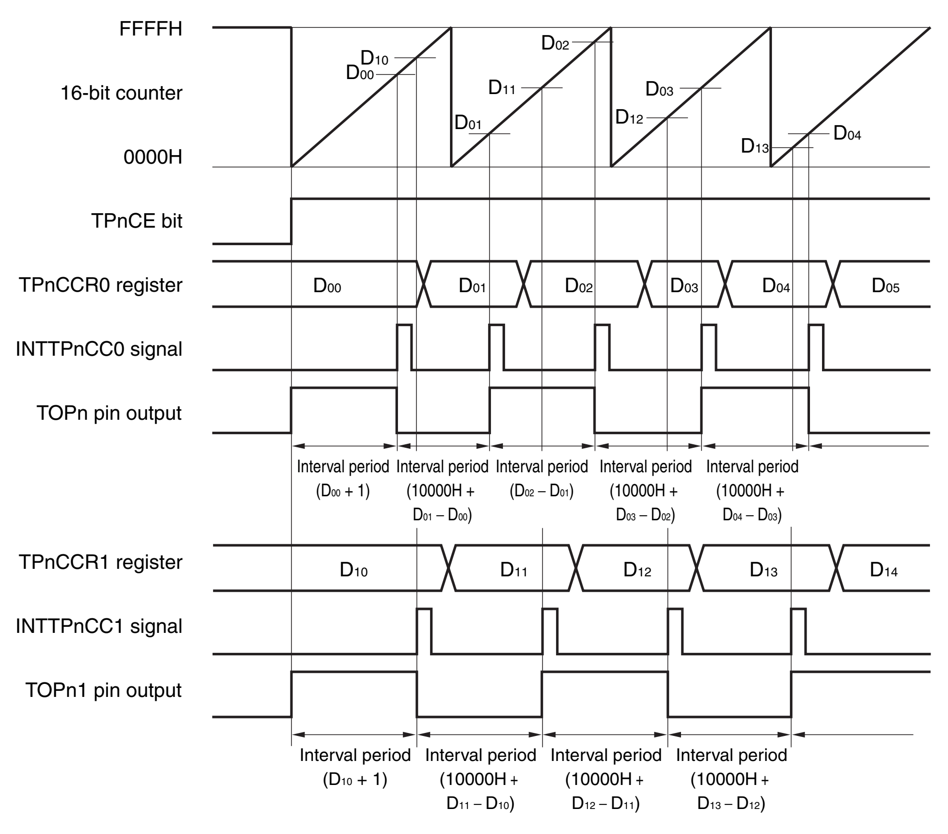


Figure 11-31 Software processing flow in free-running timer mode (capture function) (2/2)

(3) Operation timing in free-running timer mode

(a) Interval operation with compare register

When 16-bit timer/event counter P is used as an interval timer with the TPnCCRm register used as a compare register, software processing is necessary for setting a comparison value to generate the next interrupt request signal each time the INTTPnCCm signal has been detected.



When performing an interval operation in the free-running timer mode, two intervals can be set with one channel.

To perform the interval operation, the value of the corresponding TPnCCRm register must be re-set in the interrupt servicing that is executed when the INTTPnCCm signal is detected.

The set value for re-setting the TPnCCRm register can be calculated by the following expression, where “D_m” is the interval period.

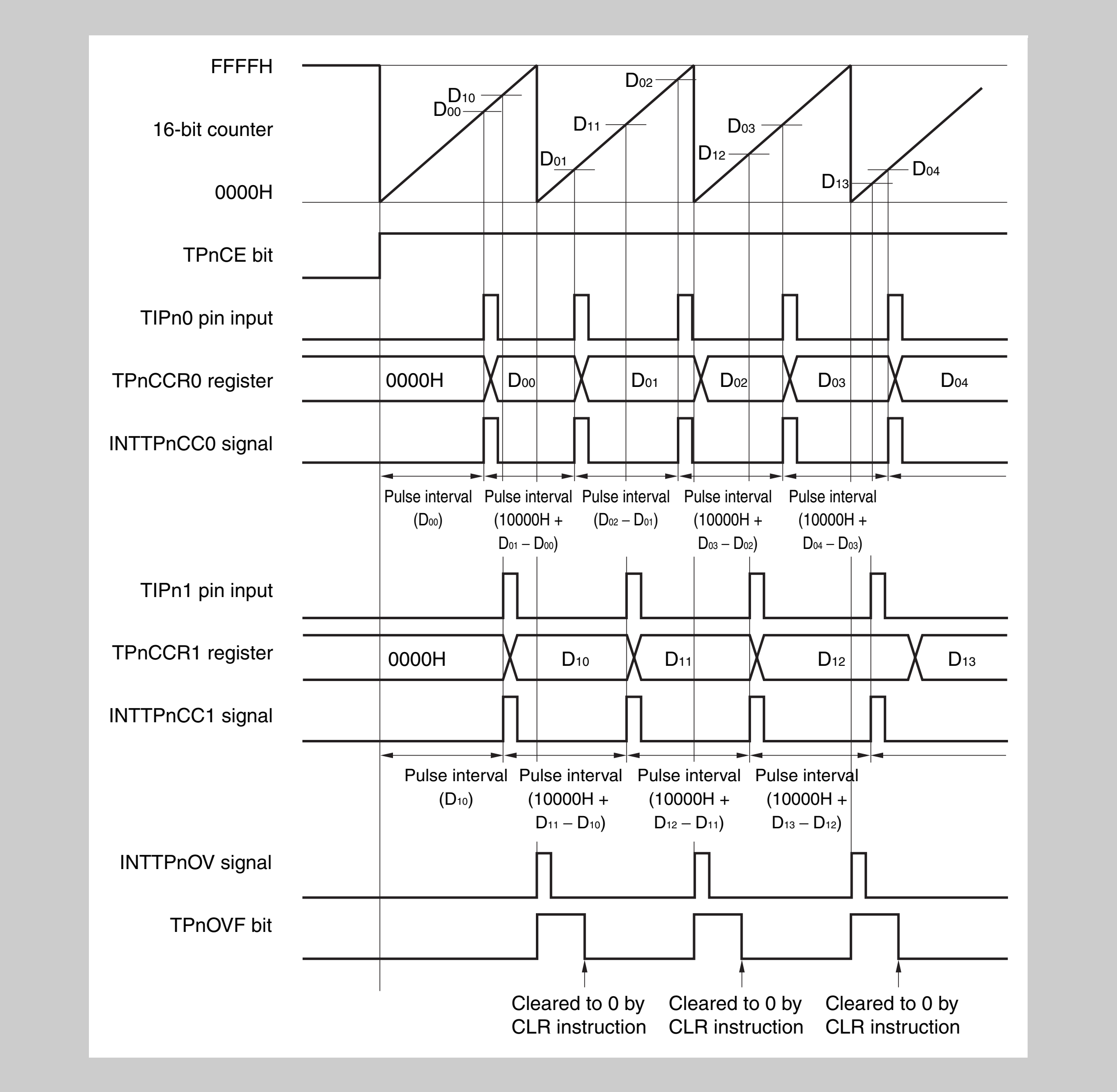
Compare register default value: $D_m - 1$

Value set to compare register second and subsequent time:
Previous set value + D_m

(If the calculation result is greater than FFFFH, subtract 10000H from the result and set this value to the register.)

(b) Pulse width measurement with capture register

When pulse width measurement is performed with the TPnCCRm register used as a capture register, software processing is necessary for reading the capture register each time the INTTPnCCm signal has been detected and for calculating an interval.



When executing pulse width measurement in the free-running timer mode, two pulse widths can be measured with one channel.

To measure a pulse width, the pulse width can be calculated by reading the value of the TPnCCRm register in synchronization with the INTTPnCCm signal, and calculating the difference between the read value and the previously read value.

(c) Processing of overflow when two capture registers are used

Care must be exercised in processing the overflow flag when two capture registers are used. First, an example of incorrect processing is shown below.

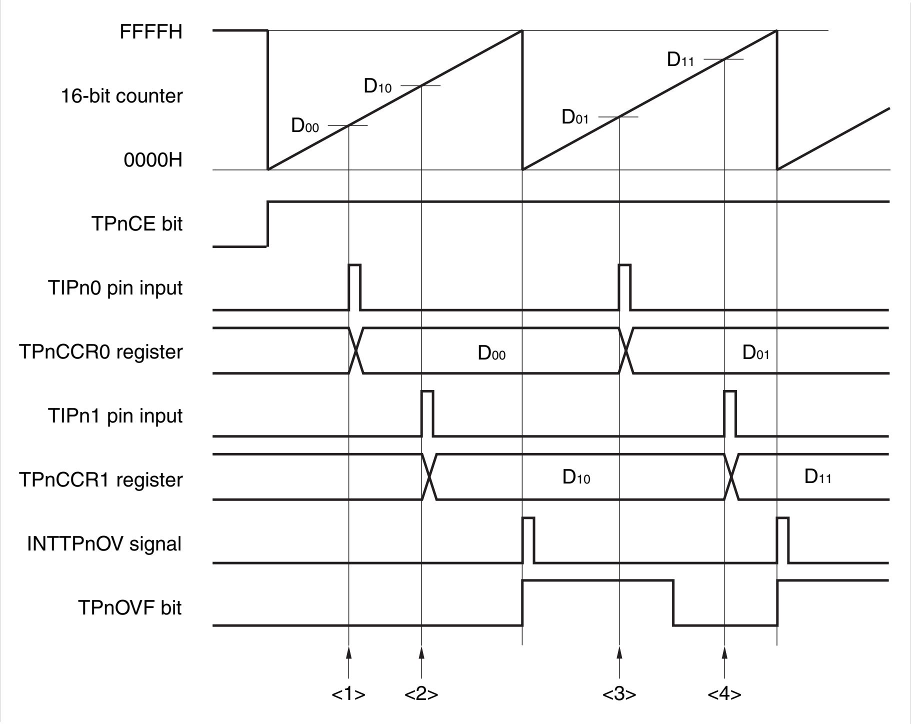


Figure 11-32 Example of incorrect processing when two capture registers are used

The following problem may occur when two pulse widths are measured in the free-running timer mode.

<1> Read the TPnCCR0 register (setting of the default value of the TIPn0 pin input).

<2> Read the TPnCCR1 register (setting of the default value of the TIPn1 pin input).

<3> Read the TPnCCR0 register.

Read the overflow flag. If the overflow flag is 1, clear it to 0.

Because the overflow flag is 1, the pulse width can be calculated by (10000H + D_{01} - D_{00}).

<4> Read the TPnCCR1 register.

Read the overflow flag. Because the flag is cleared in <3>, 0 is read.

Because the overflow flag is 0, the pulse width can be calculated by (D_{11} - D_{10}) (incorrect).

When two capture registers are used, and if the overflow flag is cleared to 0 by one capture register, the other capture register may not obtain the correct pulse width.

Use software when using two capture registers. An example of how to use software is shown below.

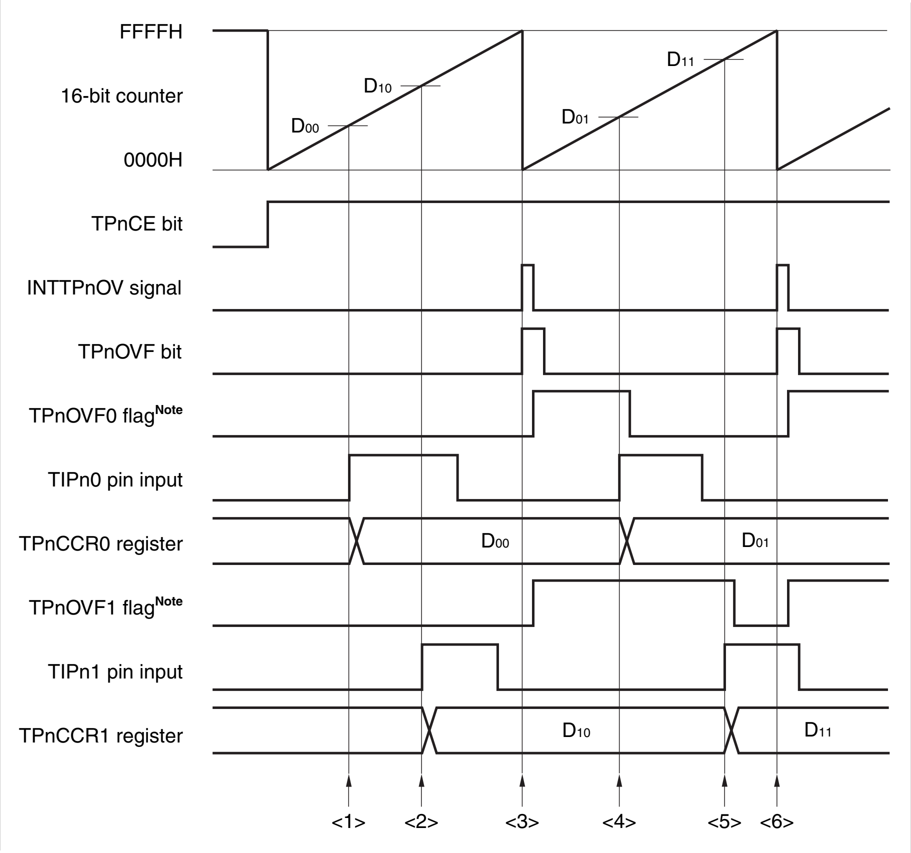


Figure 11-33 Example when two capture registers are used (using overflow interrupt)

Note The TPnOVF0 and TPnOVF1 flags are set on the internal RAM by software.

<1> Read the TPnCCR0 register (setting of the default value of the TIPn0 pin input).

<2> Read the TPnCCR1 register (setting of the default value of the TIPn1 pin input).

<3> An overflow occurs. Set the TPnOVF0 and TPnOVF1 flags to 1 in the overflow interrupt servicing, and clear the overflow flag to 0.

<4> Read the TPnCCR0 register.

Read the TPnOVF0 flag. If the TPnOVF0 flag is 1, clear it to 0.

Because the TPnOVF0 flag is 1, the pulse width can be calculated by (10000H + D_{01} - D_{00}).

<5> Read the TPnCCR1 register.

Read the TPnOVF1 flag. If the TPnOVF1 flag is 1, clear it to 0 (the TPnOVF0 flag is cleared in <4>, and the TPnOVF1 flag remains 1).

Because the TPnOVF1 flag is 1, the pulse width can be calculated by (10000H + D_{11} - D_{10}) (correct).

<6> Same as <3>

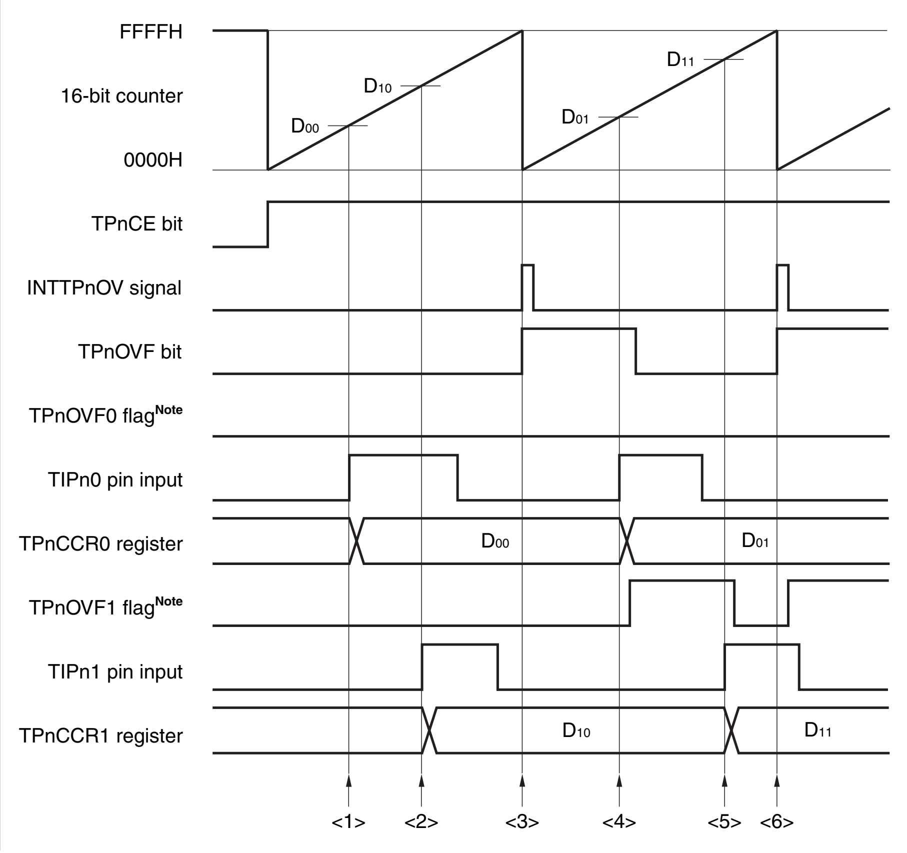


Figure 11-34 **Example when two capture registers are used (without using overflow interrupt)**

Note The TPnOVF0 and TPnOVF1 flags are set on the internal RAM by software.

<1> Read the TPnCCR0 register (setting of the default value of the TIPn0 pin input).

<2> Read the TPnCCR1 register (setting of the default value of the TIPn1 pin input).

<3> An overflow occurs. Nothing is done by software.

<4> Read the TPnCCR0 register.

Read the overflow flag. If the overflow flag is 1, set only the TPnOVF1 flag to 1, and clear the overflow flag to 0.

Because the overflow flag is 1, the pulse width can be calculated by (10000H + D_{01} - D_{00}).

<5> Read the TPnCCR1 register.

Read the overflow flag. Because the overflow flag is cleared in <4>, 0 is read.

Read the TPnOVF1 flag. If the TPnOVF1 flag is 1, clear it to 0.

Because the TPnOVF1 flag is 1, the pulse width can be calculated by (10000H + D_{11} - D_{10}) (correct).

<6> Same as <3>

(d) Processing of overflow if capture trigger interval is long

If the pulse width is greater than one cycle of the 16-bit counter, care must be exercised because an overflow may occur more than once from the first capture trigger to the next. First, an example of incorrect processing is shown below.

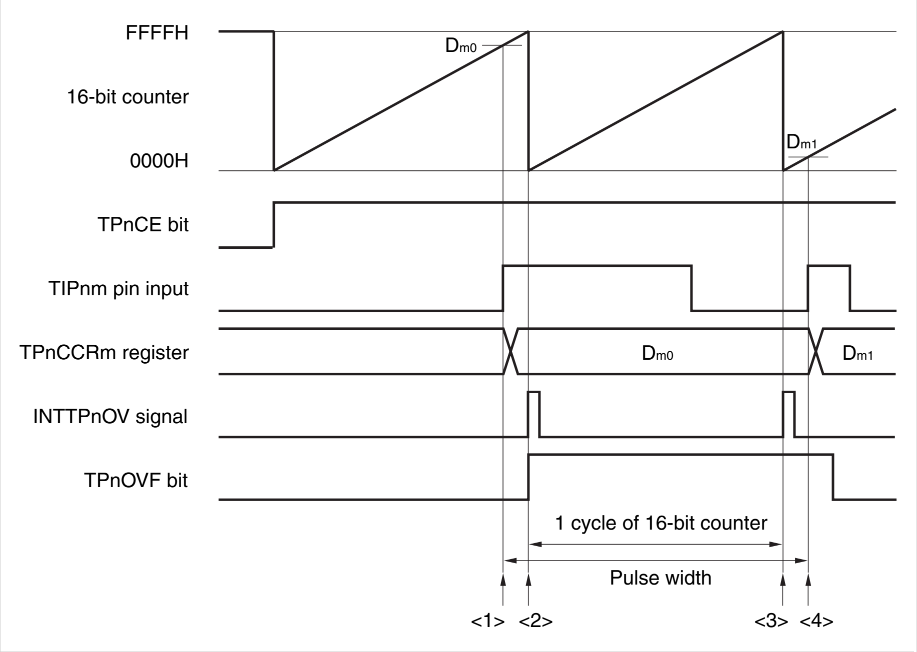


Figure 11-35 Example of incorrect processing when capture trigger interval is long

The following problem may occur when long pulse width is measured in the free-running timer mode.

<1> Read the TPnCCRm register (setting of the default value of the TIPnm pin input).

<2> An overflow occurs. Nothing is done by software.

<3> An overflow occurs a second time. Nothing is done by software.

<4> Read the TPnCCRm register.

Read the overflow flag. If the overflow flag is 1, clear it to 0.

Because the overflow flag is 1, the pulse width can be calculated by (10000H + D_{m1} - D_{m0}) (incorrect).

Actually, the pulse width must be (20000H + D_{m1} - D_{m0}) because an overflow occurs twice.

If an overflow occurs twice or more when the capture trigger interval is long, the correct pulse width may not be obtained.

If the capture trigger interval is long, slow the count clock to lengthen one cycle of the 16-bit counter, or use software. An example of how to use software is shown next.

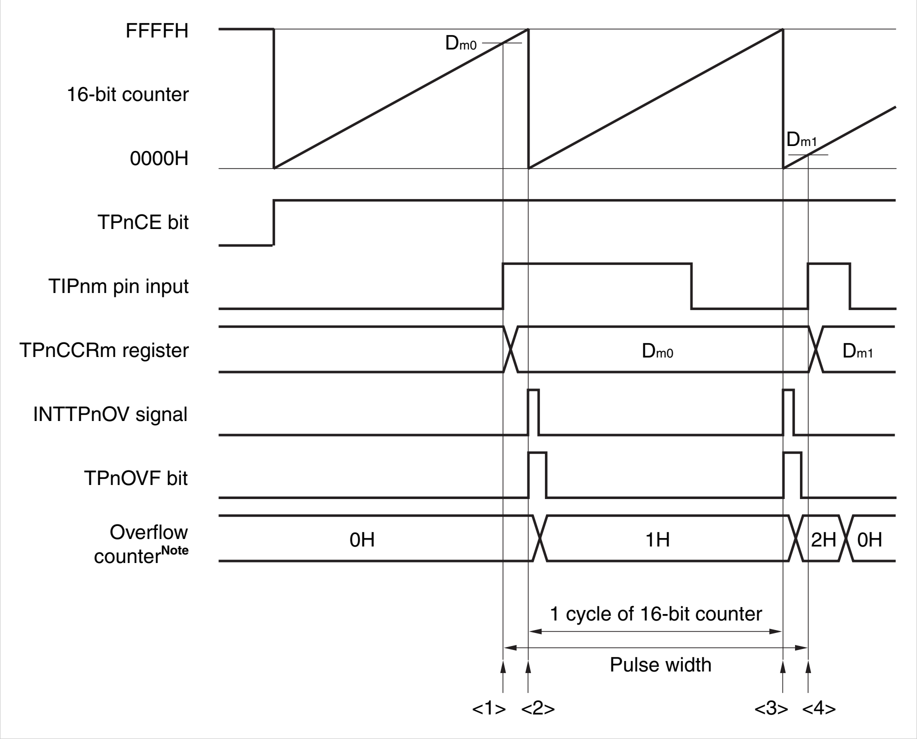


Figure 11-36 Example when capture trigger interval is long

Note The overflow counter is set arbitrarily by software on the internal RAM.

<1> Read the TPnCCRm register (setting of the default value of the TIPnm pin input).

<2> An overflow occurs. Increment the overflow counter and clear the overflow flag to 0 in the overflow interrupt servicing.

<3> An overflow occurs a second time. Increment (+1) the overflow counter and clear the overflow flag to 0 in the overflow interrupt servicing.

<4> Read the TPnCCRm register.

Read the overflow counter.

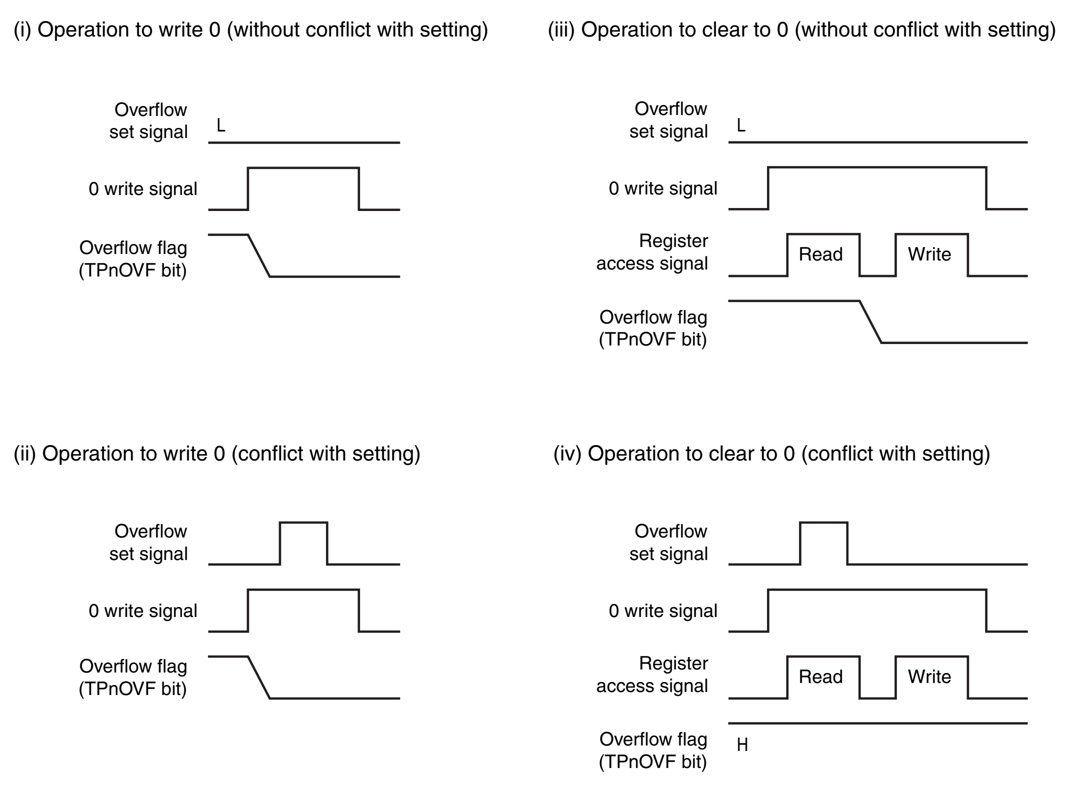
When the overflow counter is “N”, the pulse width can be calculated by (N × 10000H + D_{m1} – D_{m0}).

In this example, the pulse width is (20000H + D_{m1} – D_{m0}) because an overflow occurs twice.

Clear the overflow counter (0H).

(e) Clearing overflow flag

The overflow flag can be cleared to 0 by clearing the TPnOVF bit to 0 with the CLR instruction and by writing 8-bit data (bit 0 is 0) to the TPnOPT0 register. To accurately detect an overflow, read the TPnOVF bit when it is 1, and then clear the overflow flag by using a bit manipulation instruction.



To clear the overflow flag to 0, read the overflow flag to check if it is set to 1, and clear it with the CLR instruction. If 0 is written to the overflow flag without checking if the flag is 1, the set information of overflow may be erased by writing 0 ((ii) in the above chart). Therefore, software may judge that no overflow has occurred even when an overflow actually has occurred.

If execution of the CLR instruction conflicts with occurrence of an overflow when the overflow flag is cleared to 0 with the CLR instruction, the overflow flag remains set even after execution of the clear instruction.

11.5.7 Pulse width measurement mode (TPnMD2 to TPnMD0 = 110)

In the pulse width measurement mode, 16-bit timer/event counter P starts counting when the TPnCTL0.TPnCE bit is set to 1. Each time the valid edge input to the TIPnm pin has been detected, the count value of the 16-bit counter is stored in the TPnCCRm register, and the 16-bit counter is cleared to 0000H.

The interval of the valid edge can be measured by reading the TPnCCRm register after a capture interrupt request signal (INTTPnCCm) occurs.

Select either the TIPn0 or TIPn1 pin as the capture trigger input pin. Specify "No edge detected" by using the TPnIOC1 register for the unused pins.

When an external clock is used as the count clock, measure the pulse width of the TIPn1 pin because the external clock is fixed to the TIPn0 pin. At this time, clear the TPnIOC1.TPnIS1 and TPnIOC1.TPnIS0 bits to 00 (capture trigger input (TIPn0 pin): No edge detected).

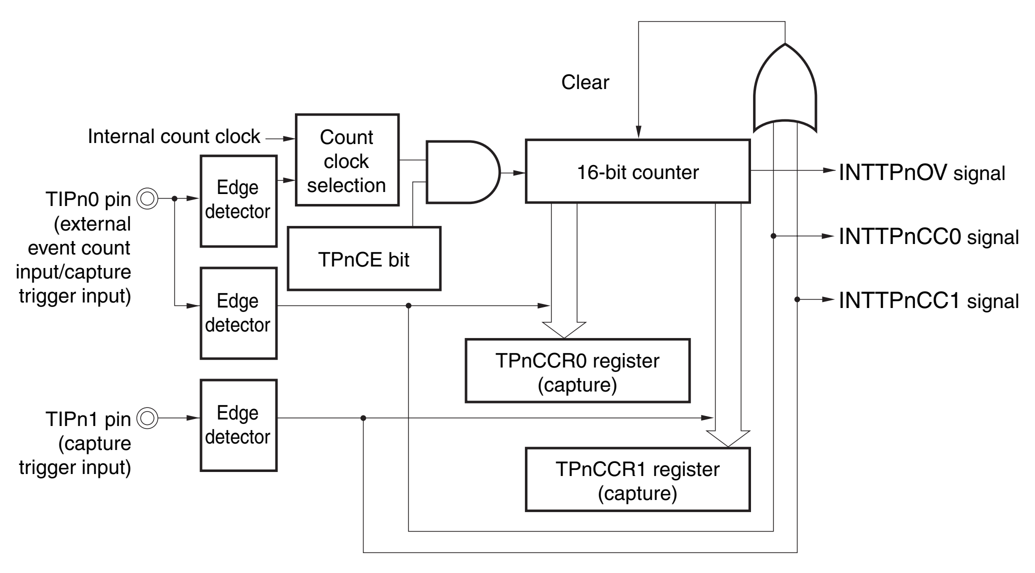


Figure 11-37 Configuration in pulse width measurement mode

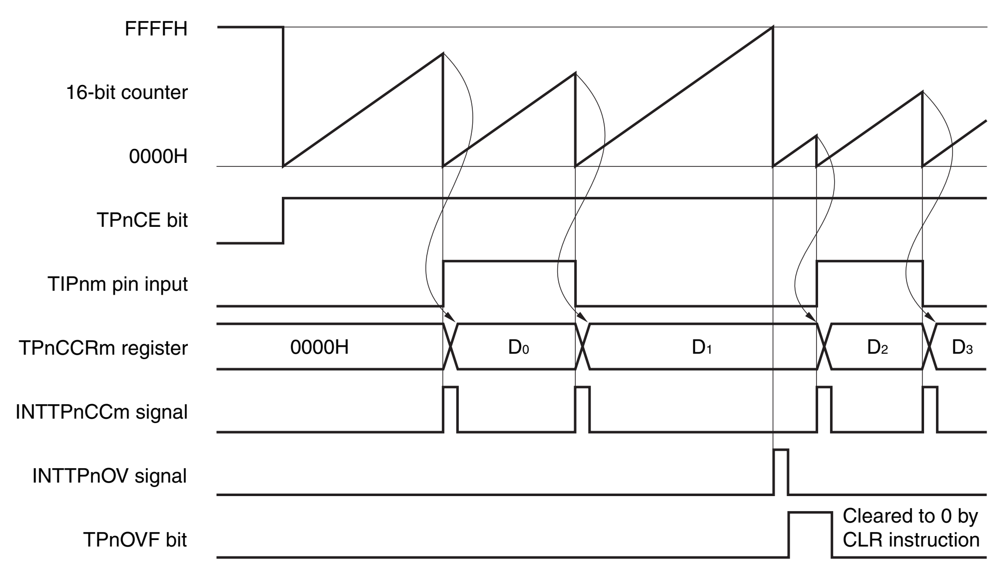


Figure 11-38 Basic timing in pulse width measurement mode

When the TPnCE bit is set to 1, the 16-bit counter starts counting. When the valid edge input to the TIPnm pin is later detected, the count value of the 16-bit counter is stored in the TPnCCRm register, the 16-bit counter is cleared to 0000H, and a capture interrupt request signal (INTTPnCCm) is generated.

The pulse width is calculated as follows.

First pulse width = $(D_0 + 1)$ × Count clock cycle

Second and subsequent pulse width = $(D_N - D_{N-1})$ × Count clock cycle

If the valid edge is not input to the TIPnm pin even when the 16-bit counter counted up to FFFFH, an overflow interrupt request signal (INTTPnOV) is generated at the next count clock, and the counter is cleared to 0000H and continues counting. At this time, the overflow flag (TPnOPT0.TPnOVF bit) is also set to 1. Clear the overflow flag to 0 by executing the CLR instruction via software.

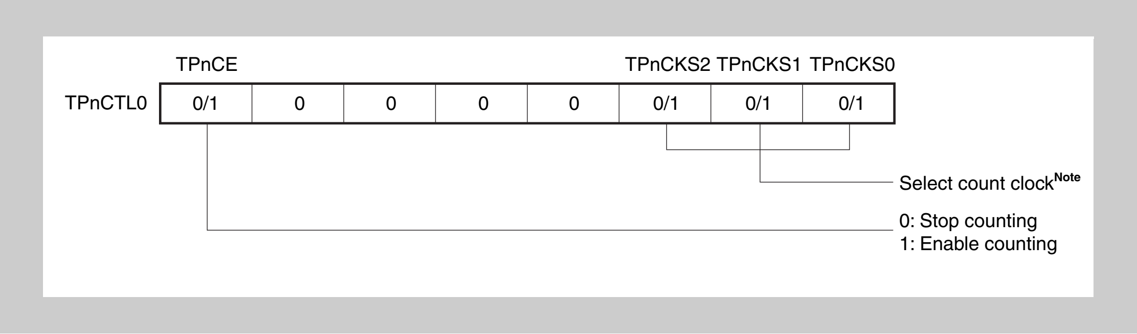
If the overflow flag is set to 1, the pulse width can be calculated as follows.

First pulse width = $(D_0 + 10001H)$ × Count clock cycle

Second pulse width and on = $(10000H + D_N - D_{N-1})$ × Count clock cycle

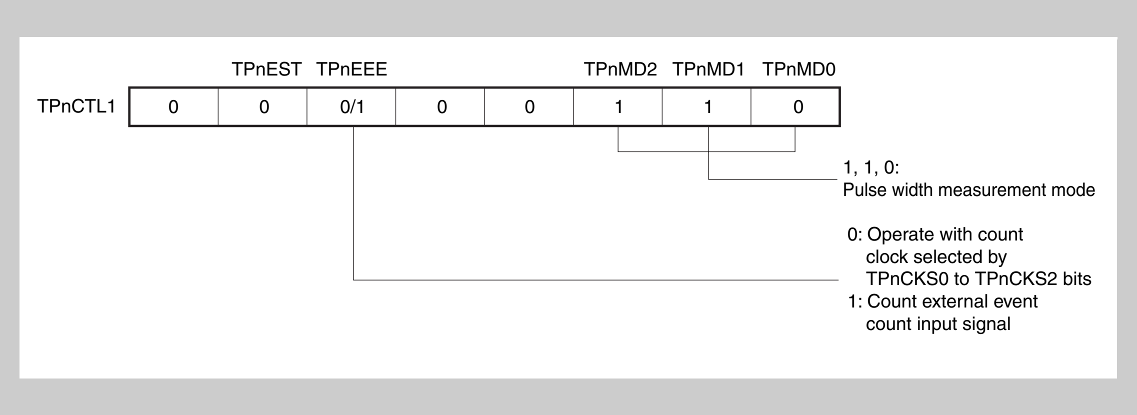
(1) Register setting in pulse width measurement mode

(a) TMPn control register 0 (TPnCTL0)

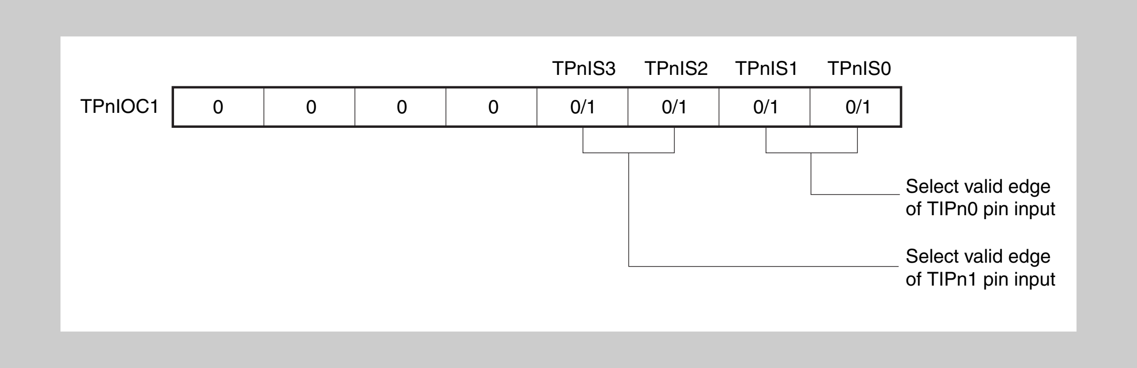


Note Setting is invalid when the TPnEEE bit = 1.

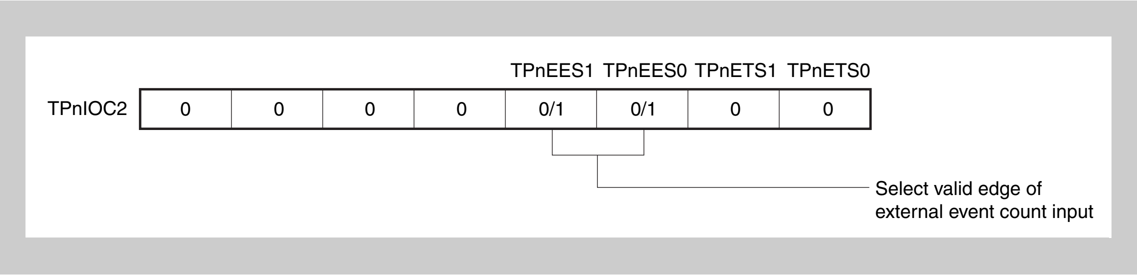
(b) TMPn control register 1 (TPnCTL1)



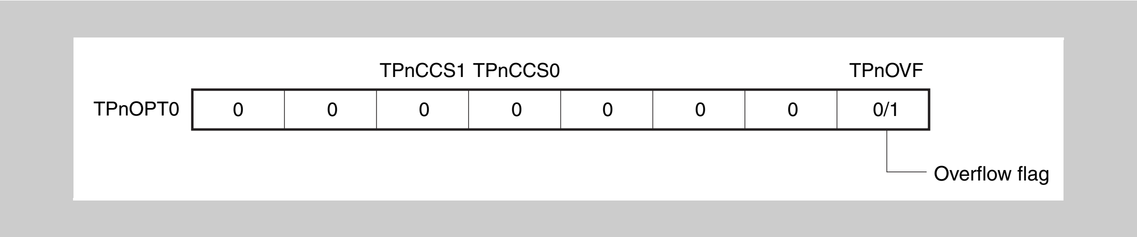
(c) TMPn I/O control register 1 (TPnIOC1)



(d) TMPn I/O control register 2 (TPnIOC2)



(e) TMPn option register 0 (TPnOPT0)



(f) TMPn counter read buffer register (TPnCNT)

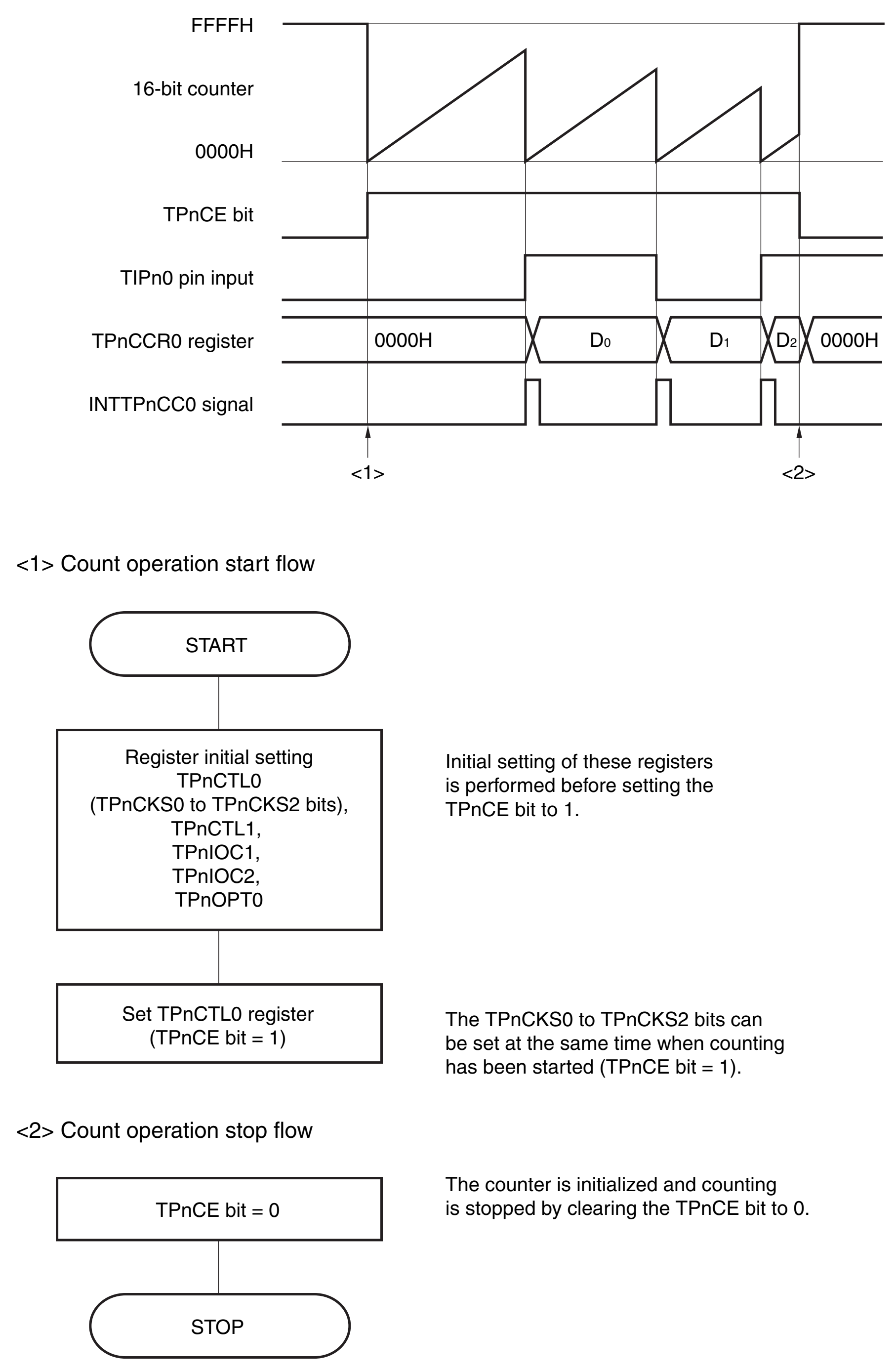
The value of the 16-bit counter can be read by reading the TPnCNT register.

(g) TMPn capture/compare registers 0 and 1 (TPnCCR0 and TPnCCR1)

These registers store the count value of the 16-bit counter when the valid edge input to the TIPnm pin is detected.

Note TMPn I/O control register 0 (TPnIOC0) is not used in the pulse width measurement mode.

(2) Operation flow in pulse width measurement mode



Initial setting of these registers is performed before setting the TPnCE bit to 1.

The TPnCKS0 to TPnCKS2 bits can be set at the same time when counting has been started (TPnCE bit = 1).

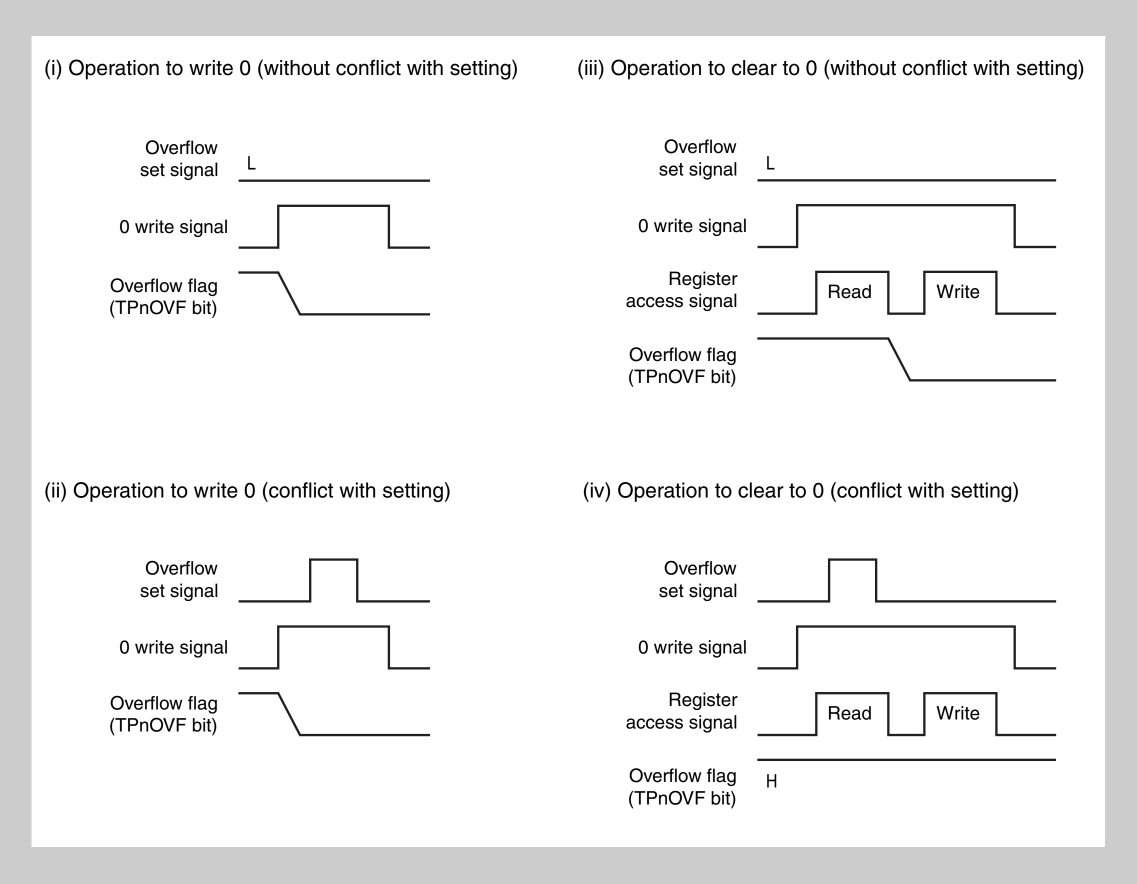
The counter is initialized and counting is stopped by clearing the TPnCE bit to 0.

Figure 11-39 Software processing flow in pulse width measurement mode

(3) Operation timing in pulse width measurement mode

(a) Clearing overflow flag

The overflow flag can be cleared to 0 by clearing the TPnOVF bit to 0 with the CLR instruction and by writing 8-bit data (bit 0 is 0) to the TPnOPT0 register. To accurately detect an overflow, read the TPnOVF bit when it is 1, and then clear the overflow flag by using a bit manipulation instruction.



To clear the overflow flag to 0, read the overflow flag to check if it is set to 1, and clear it with the CLR instruction. If 0 is written to the overflow flag without checking if the flag is 1, the set information of overflow may be erased by writing 0 ((ii) in the above chart). Therefore, software may judge that no overflow has occurred even when an overflow actually has occurred.

If execution of the CLR instruction conflicts with occurrence of an overflow when the overflow flag is cleared to 0 with the CLR instruction, the overflow flag remains set even after execution of the clear instruction.

11.5.8 Timer output operations

The following table shows the operations and output levels of the TOPn0 and TOPn1 pins.

Table 11-11 Timer output control in each mode

Operation Mode	TOPn1 Pin	TOPn0 Pin
Interval timer mode	Square wave output	
External event count mode	Square wave output	–
External trigger pulse output mode	External trigger pulse output	Square wave output
One-shot pulse output mode	One-shot pulse output	
PWM output mode	PWM output	
Free-running timer mode	Square wave output (only when compare function is used)	
Pulse width measurement mode	–	

Table 11-12 Truth table of TOPn0 and TOPn1 pins under control of timer output control bits

TPnIOC0.TPnOLm Bit	TPnIOC0.TPnOEm Bit	TPnCTL0.TPnCE Bit	Level of TOPnm Pin
0	0	×	Low-level output
	1	0	Low-level output
		1	Low level immediately before counting, high level after counting is started
1	0	×	High-level output
	1	0	High-level output
		1	High level immediately before counting, low level after counting is started

11.6 Operating Precautions

11.6.1 Capture operation in pulse width measurement and free-running mode

When the capture operation is used in pulse width measurement or free-running mode the first captured counter value of the capture registers TPnCCR0/ TPnCCR, i.e. after the timer is enabled (TPnCTL0.TPnCE = 1), may be $FFFF_H$ instead of 0000_H if the chosen count clock of the TMP is not the maximum, i.e. if TPnCTL0.TPnCKS[2:0] $\neq$ 0.

11.6.2 Count jitter for PCLK4 to PCLK7 count clocks

When specifying PCLK4 to PCLK7 as the count clock, a jitter of maximum ± 1 period of PCLK0 may be applied to the counter's count clock input.

Chapter 12 16-bit Interval Timer Z (TMZ)

Timer Z (TMZ) is a general purpose 16-bit timer/counter.

The V850E/Dx3 microcontrollers have following instances of the general purpose Timer Z:

TMZ	**µPD70F3427, µPD70F3426 µPD70F3425, µPD70F3424**	**µPD70F3423, µPD70(F)3422 µPD70(F)3421, µPD70(F)3420**
Instances	10	6
Names	TMZ0 to TMZ9	TMZ0 to TMZ5

Throughout this chapter, the individual instances of Timer Z are identified by “n”, for example TMZn, or TZnCTL for the TMZn control register.

12.1 Overview

Each Timer Z has one down-counter. When the counter reaches zero, the timer generates the maskable interrupt INTTZnUV.

The TMZ can be used as:

- Interval timer
- Free running timer

Features summary

Special features of the TMZ are:

- One of six peripheral clocks can be selected
- One reload register
- Two readable counter registers
- When the device is in debug mode, the timer can be stopped at breakpoint
- TMZ0, TMZ1, and TMZ2 can be used for triggering the DMA Controller.
- TMZ5 can be used for triggering the A/D Converter.
- All TMZn can be optionally stopped when a breakpoint is hit during debugging (refer to *“On-Chip Debug Unit“ on page 877*).

12.1.1 Description

The TMZ has no external connections. It is built up as illustrated in the following figure.

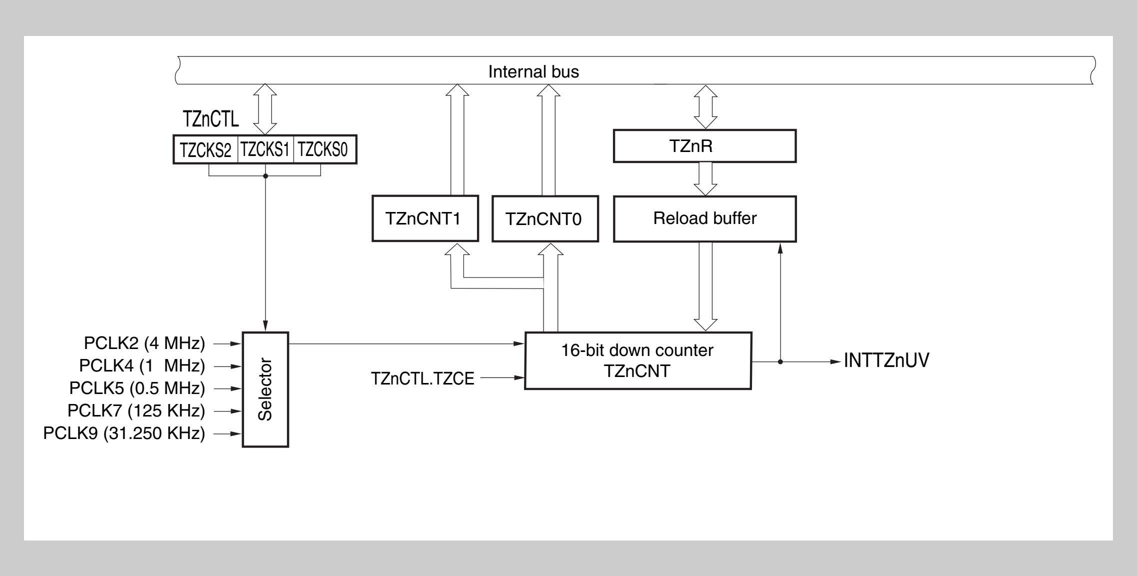


Figure 12-1 Block diagram of Timer Z (TMZn)

The control register TZnCTL allows you to choose the clock and to enable the timer. The latter is done by setting TZnCTL.TZCE to 1.

As soon as the timer is enabled, it is possible to write a start value to the reload register TZnR.

12.1.2 Principle of operation

When it is enabled, the counter starts as soon as a non-zero value is written to the reload register TZnR and copied to the reload buffer.

When the counter reaches zero, it generates an INTTZnUV interrupt, reloads its start value from the reload buffer, and continues counting.

Two read-only registers (TZnCNT0 and TZnCNT1) provide the updated counter value. For details about these registers please refer to *“TZnCNT0 - TMZn synchronized counter register“ on page 433* and *“TZnCNT1 - TMZn non-synchronized counter register“ on page 433*.

12.2 TMZ Registers

Each Timer Z is controlled and operated by means of the following four registers:

Table 12-1 Timer Z registers overview

Register name	Shortcut	Address
Timer Z synchronized read register	TZnCNT0	<base>
Timer Z non-synchronized read register	TZnCNT1	<base> + 2_H
Timer Z reload register	TZnR	<base> + 4_H
Timer Z control register	TZnCTL	<base> + 6_H

Table 12-2 Base addresses of Timer Z

Timer	Base address
TMZ0	FFFF $F600_H$
TMZ1	FFFF $F608_H$
TMZ2	FFFF $F610_H$
TMZ3	FFFF $F618_H$
TMZ4	FFFF $F620_H$
TMZ5	FFFF $F628_H$
TMZ6	FFFF $F630_H$
TMZ7	FFFF $F638_H$
TMZ8	FFFF $F640_H$
TMZ9	FFFF $F648_H$

(1) TZnCTL - TMZn timer control register

The 8-bit TZnCTL register controls the operation of the Timer Z.

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

Address <base> + 6_H

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

7	6	5	4	3	2	1	0
TZCE	0	0	0	0	TZCKS2	TZCKS1	TZCKS0
R/W	R	R	R	R	R/W	R/W	R/W

Table 12-3 TZnCTL register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>TZCE</td><td>Timer Z counter enable:
0: Disable count operation (the timer stops immediately with the count value 0000_H and does not operate).
1: Enable count operation (the timer starts when a non-zero start value is written to the register TZnR after TZnCTL.TZCE=1).</td></tr>
<tr><td>2 to 0</td><td>TZCKS[2:0]</td><td>Selects the counter clock:
<table>
<tr><th>TZCKS2</th><th>TZCKS1</th><th>TZCKS0</th><th>Counter clock selection</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>PCLK2 (4 MHz)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>PCLK4 (1 MHz)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>PCLK5 (0.5 MHz)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>PCLK7 (0.125 MHz)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>PCLK9 (31.250 KHz)</td></tr>
<tr><td colspan="3">Others than above</td><td>Setting prohibited</td></tr>
</table>
</td></tr>
</table>

Note Change bits TZnCTL.TZCKS[2:0] only when TZnCTL.TZCE = 0.

When TZnCTL.TZCE = 0, it is possible to select the clock and enable the counter with one write operation.

(2) TZnCNT0 - TMZn synchronized counter register

The TZnCNT0 register is the synchronized register that can be used to read the present value of the 16-bit counter.

“Synchronized” means that the read access via the internal bus is synchronized with the maximum counter clock (PCLK2). The synchronization process may cause a delay, but the resulting value is reliable.

Access This register is read-only, in 16-bit units.

Address <base> of TMZn

Initial Value 0000_H. This register is cleared by any reset and when TZnCTL.TZCE = 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Updated counter value (synchronized)															
R															

(3) TZnCNT1 - TMZn non-synchronized counter register

The TZnCNT1 register is the non-synchronized register that can be used to read the present value of the corresponding 16-bit counter.

“Non-synchronized” means that the read access via the internal bus is not synchronized with the counter clock. It returns the instantaneous value immediately, with the risk that this value is just being updated by the counter and therefore in doubt.

Access This register is read-only, in 16-bit units.

Address <base> + 2_H

Initial Value 0000_H. This register is cleared by any reset and when TZnCTL.TZCE = 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Instantaneous counter value (non-synchronized)															
R															

Note The value read from this register can be incorrect, because the read access is not synchronized with the counter clock.

Therefore, this register shall be read multiple times within one period of the counter clock cycle.

If the difference between the first and the second value is not greater than one, you can consider the second value to be correct. If the difference between the two values is greater than one, you have to read the register a third time and compare the third value with the second. Again, the difference must not be greater than one.

If the read accesses do not happen within one period of the counter clock cycle, the difference between the last two values will usually be greater than one. In this case, you can only repeat the procedure or estimate the updated counter value.

(4) TZnR - Reload register

The TZnR register is a dedicated register for setting the reload value of the corresponding counter.

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

Address <base> + 4_H

Initial Value 0000_H. This register is cleared by any reset and when TZnCTL.TZCE = 0.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">Down-counter load value</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

Note

1. TZnR can only be written when TZnCTL.TZCE = 1.
2. The load value must be non-zero (0001_H … $FFFF_H$).
3. To operate the timer in free running mode, set TZnR to $FFFF_H$.
4. The first interval after starting the counter can require one additional clock cycle. For details refer to *“Timer start and stop“ on page 436*.

12.3 Timing

The contents of the reload register TZnR can be changed at any time, provided the timer is enabled. The contents is then copied to the reload buffer. However, the counter reloads its start value from the buffer only upon underflow.

Caution When specifying PCLK4, PLCK5, PCLK7 or PCLK9 as the count clock, a jitter of maximum ± 1 period of PCLK2 may be applied to the TZnCNT counter's count clock input.

12.3.1 Steady operation

Steady operation is illustrated in the following figure.

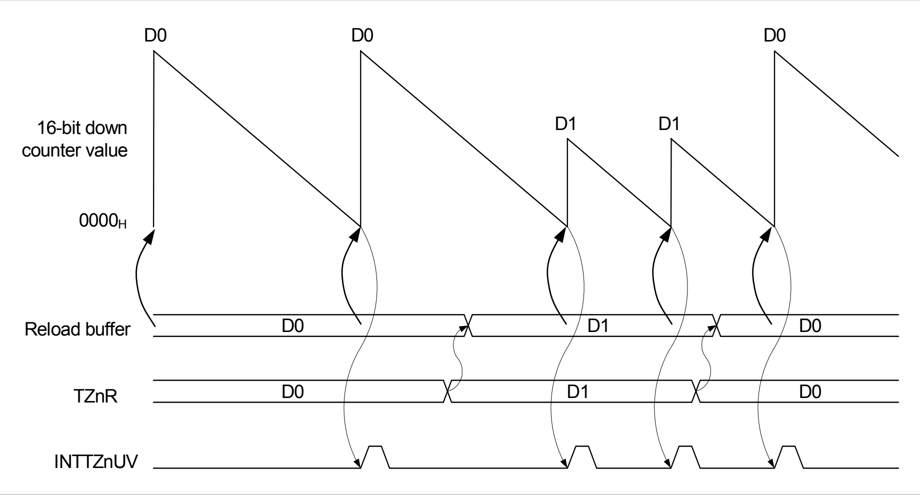


Figure 12-2 Reload timing and interrupt generation

D0 and D1 are two different reload values.

Note that there is a delay between writing to TZnR and making the data available in the reload buffer, depending on the previous reload value and the chosen count clock.

12.3.2 Timer start and stop

The timer TZn is enabled by setting TZnCTL.TZCE to 1.

Start The subsequent write access to register TZnR with non-zero data starts the timer. After that, it is prepared to load the value written to register TZnR into the reload buffer and the counter.

The following interval times are given in periods of PCLK2.

If PCLK2 is chosen as the counter clock ([TZnCTL.TZCKS] = 0), the first and all following interrupts occur after

$$T_{interval} = ([TZnR] + 1)$$

where

[TZnR] = contents of register TZnR

An uncertainty exists for the first interval length, if a clock with a lower frequency is chosen ([TZnCTL.TZCKS] > 0):

$$([TZnR] + 1) \times 2^{[TZCKS]} \leq T_{interval} \leq ([TZnR] + 2) \times 2^{[TZCKS]}$$

where

[TZnR] = contents of register TZnR

[TZCKS] = contents of TZnCTL.TZCKS[2:0]

All following interrupts occur after:

$$T_{interval} = ([TZnR] + 1) \times 2^{[TZCKS]}$$

Stop The timer stops when TZnCTL.TZCE is cleared. This write access is not synchronized. The timer is immediately stopped, and its registers are reset.

Chapter 13 16-bit Multi-Purpose Timer G (TMG)

The V850E/Dx3 microcontrollers have following instances of the 16-bit multi-purpose Timer G:

Throughout this chapter, the individual instances of Timer G are identified by “n”, for example TMGn, or TMGMn for the TMGn mode register.

Note Throughout this chapter, the following indexes are used:

• n:	for each of the Timer G instances
• m = 1 to 4:	for the free assignable Input/Output-channels
• x = 0, 1:	for bit-index, i.e. one of the 2 counters of each Timer Gn
• y = 0 to 5:	for all of the 6 capture/compare-channels

13.1 Features of Timer G

The timers Gn operate as:

- Pulse interval and frequency measurement counter
- Interval timer
- Programmable pulse output
- PWM output timer
- TMG0 can be used for triggering the DMA Controller.

One capture input of Timer G 0 and one capture input of Timer G 1 are connected to the time stamp outputs of CAN0 and CAN1 modules and can therefore be used for CAN time stamp functions.

13.2 Function Overview of Each Timer Gn

- 16-bit timer/counter (TMGn0, TMGn1): 2 channels
- Bit length
  - Timer Gn registers (TMGn0, TMGn1): 16 bits
- Capture/compare register (GCCny): 6
  - 16-bit
  - 2 registers are assigned fix to the corresponding one of the 2 counters
  - 4 free assignable registers to one of the 2 counters
- Count clock division selectable by prescaler (frequency of peripheral clock: f_{SPCLK0} = 16 MHz)
  - In 8 steps from $f_{SPCLK0}/2$ to $f_{SPCLK0}/256$
- Interrupt request sources
  - Edge detection circuit with noise elimination.
  - Compare-match interrupt requests: 6 types
    Perform comparison of capture/compare register with one of the 2 counters (TMGn0, TMGn1) and generate the INTCCGny (y = 0 to 5) interrupt upon compare match.
  - Timer counter overflow interrupt requests: 2 types
    In free run mode the INTTMGn0 (INTTMGn1) interrupt is generated when the count value of TMGn0 (TMGn1) toggles from FFFFH to 0000H.
  - In match and clear mode the INTTMGn0 (INTTMGn1) interrupt is generated when the count value of TMGn0 (TMGn1) matches the GCC0 (GCC1) value.
- PWM output function
  - Control of the outputs of TOGn1- through TOGn4-pin in the compare mode. PWM output can be performed using the compare match timing of the GCCn1 to GCCn4 register and the corresponding timebase (TMGn0, TMGn1).
- Output delay operation
  - A clock-synchronized output delay can be added to the output signal of pins TOGn1 to TOGn4.
  - This is effective as an EMI counter measure.
- Edge detection and noise elimination filter
  - External signals shorter than 1 count clock (f_{COUNTn}, not f_{SPCLK0}) are eliminated as noise.

Note The TIGn1 to TIGn4 and TOGn1 to TOGn4 are each alternative function pins.

The following figure shows the block diagram of Timer Gn.

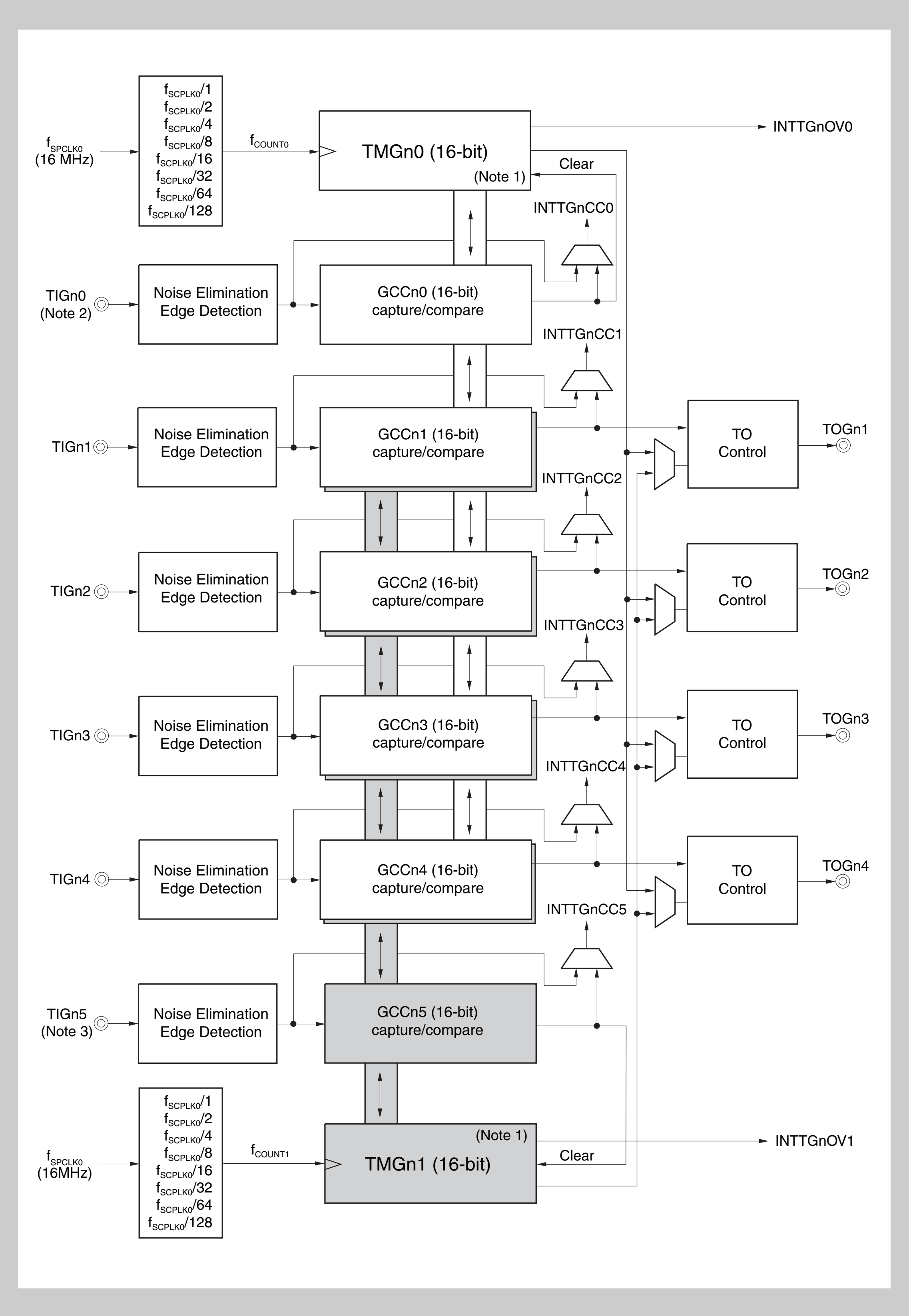


Figure 13-1 Block Diagram of Timer Gn

Note 1. TMGn0/TMGn1 are cleared by GCCn0/GCCn5 register compare match.

2. TIGn0 differs:
   - n = 0: TIG00 not connected
   - n = 1: TIG10 not connected
   - n = 2: TIG20 available as external capture input
3. TIGn5 differs:
   - n = 0: CAN0 time stamp TSOUTCAN0 -> TIG05
   - n = 1: CAN1 time stamp TSOUTCAN1 -> TIG15
   - n = 2: TIG25 available as external capture input

13.3 Basic Configuration

The basic configuration is shown below.

Table 13-1 Timer Gn configuration list

Count clock	Register	R/W	Generated interrupt signal	Capture trigger	Timer output PWM
f_{SPCLK0} f_{SPCLK0} /2, f_{SPCLK0} /4, f_{SPCLK0} /8, f_{SPCLK0} /16, f_{SPCLK0} /32, f_{SPCLK0} /64, f_{SPCLK0} /128	TMGn0	R	INTTMGn0	-	-
	TMGn1	R	INTTMGn1	-	-
	GCCn0	R/W	INTCCGn0	TIGn0	-
	GCCn1	R/W	INTCCGn1	TIGn1	TOGn1
	GCCn2	R/W	INTCCGn2	TIGn2	TOGn2
	GCCn3	R/W	INTCCGn3	TIGn3	TOGn3
	GCCn4	R/W	INTCCGn4	TIGn4	TOGn4
	GCCn5	R/W	INTCCGn5	TIGn5	-

Note f_{SPCLK0}: Internal peripheral clock

13.4 TMG Registers

The Timers Gn are controlled and operated by means of the following registers:

Table 13-2 TMGn registers overview

Register name	Shortcut	Address
Timer Gn mode register	TMGMn	<base>
Timer Gn channel mode register	TMGCMn	<base> + 2_H
Timer Gn output control register	OCTLGn	<base> + 4_H
Timer Gn time base status register	TMGSTn	<base> + 6_H
Timer Gn count register 0	TMG00	<base> + 8_H
Timer Gn count register 1	TMG01	<base> + A_H
Timer Gn capture/compare register 0	GCC00	<base> + C_H
Timer Gn capture/compare register 1	GCC01	<base> + E_H
Timer Gn capture/compare register 2	GCC02	<base> + 10_H
Timer Gn capture/compare register 3	GCC03	<base> + 12_H
Timer Gn capture/compare register 4	GCC04	<base> + 14_H
Timer Gn capture/compare register 5	GCC05	<base> + 16_H

Table 13-3 TMGn register base address

Timer	Base address
TMG0	FFFF $F6A0_H$
TMG1	FFFF $F6C0_H$
TMG2	FFFF $F6E0_H$

(1) TMGMn - Timer Gn mode register

Access This register can be read/written in 16-bit, 8-bit or 1-bit units. The low byte TMGMn.bit[7:0] is accessible separately under the name TMGMnL, the high byte TMGMn.bit[15:8] under the name TMGMnH.

Address
TMGMn, TMGMnL: <base>
TMGMnH: <base> + 1_H

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

15	14	13	12	11	10	9	8
POWERn	OLDEn	CSEn12	CSEn11	CSEn10	CSE002	CSEn01	CSEn00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
CCSGn5	CCSGn0	0	0	CLRGn1	TMGn1E	CLRGn0	TMGn0E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 13-4 TMGMn register contents (1/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>15</td><td>POWERn</td><td>Timer Gn Operation control.
0: operation Stop
the capture registers and TMGSTn register are cleared
the TOGnm pins are inactive all the time
1: operation enable
Note: At least 7 peripheral clocks (f_{SPCLK0}) are needed to start the timer function</td></tr>
<tr><td>14</td><td>OLDEn</td><td>Set Output Delay Operation.
0: Don’t perform output delay operation
1: Set output delay to n count-clocks
Caution: When the POWERn bit is set, the rewriting of this bit is prohibited! Simultaneously writing with the POWERn bit is allowed.
Note: The delay operation is used for EMI counter measures.</td></tr>
<tr><td>13 to 8</td><td>CSEnx[2:0]</td><td>Selects internal count clock of TMG
<table>
<tr><th>CSEnx2</th><th>CSEnx1</th><th>CSEnx0</th><th>Count clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>f_{SPCLK0}</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$f_{SPCLK0}/2$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$f_{SPCLK0}/4$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$f_{SPCLK0}/8$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$f_{SPCLK0}/16$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>$f_{SPCLK0}/32$</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>$f_{SPCLK0}/64$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>$f_{SPCLK0}/128$</td></tr>
</table>
Caution: When the POWERn bit is set, the rewriting of this bits are prohibited! Simultaneously writing with the POWERn bit is allowed.</td></tr>
</table>

Table 13-4 TMGMn register contents (2/2)

Bit position	Bit name	Function
7, 6	CCSGn5 CCSGn0	Specifies the mode of the TMGn0 (TMGn1)(CCSGn5 for TMGn1, CCSGn0 for TMGn0): 0: Free-run mode for TMGn1 (TMGn0), GCCn5 (GCCn0) in capture mode (an detected edge at Pin TIGn5 (TIGn0) stores the value of TMGn1 (TMGn0) in GCCn5 (GCCn0) and an interrupt INTCCGn5 (INTCCGn0) is output) 1: Match and Clear mode of the TMGn1 (TMGn0), GCCn5 (GCCn0) in compare mode (when the data of GCCn5 (GCCn0) match the count value of the TMGn1 (TMGn0), the counter is cleared and the interrupt INTCCGn5 (INTCCGn0) occurs) **Caution:** When the POWERn bit is set, the rewriting of this bits are prohibited! Simultaneously writing with the POWERn bit is allowed.
3, 1	CLRGnx	Specifies software clear for TMGnx 0: Continue TMGnx operation 1: Clears (0) the count value of TMGnx, the corresponding TOGnx is deactivated. **Note:** TMGnx starts 1 peripheral-clock after this bit is set this bit is not readable (always read 0)
2, 0	TMGnxE	Specifies TMGnx count operation enable/disable 0: Stop count operation the counter holds the immediate preceding value the corresponding TOGnx is deactivated 1: Enable count operation **Note:** **1.** the counter needs at least 1 peripheral-clock (f_{SPCLK0}) to stop **2.** the counter needs at least 4 peripheral-clocks (f_{SPCLK0}) to start

(2) TMGCMn - Timer Gn channel mode register

This register specifies the assigned counter (TMGn0 or TMGn1) for the GCCnm register.
Furthermore it specifies the edge detection for the TIGny-input-pins.

Access This register can be read/written in 16-bit, 8-bit or 1-bit units.
The low byte TMGCMn.bit[7:0] is accessible separately under the name TMGCMnL, the high byte TMGCMn.bit[15:8] under the name TMGCMnH.

Address
TMGCMn, TMGCMnL: <base> + 2_H
TMGCMnH: <base> + 3_H

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

15	14	13	12	11	10	9	8
TBGn4	TBGn3	TBGn2	TBGn1	IEGn51	IEGn50	IEGn41	IEGn40
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
IEGn31	IEGn30	IEGn21	IEGn20	IEGn11	IEGn10	IEGn01	IEGn00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 13-5 TMGCMn register contents

Bit position	Bit name	Function
15 to 12	TBGnm	Assigns Capture/Compare registers GCCn1 to GCCn4 to one of the 2 counters TMGn0 or TMGn1: 0: Set TMGn0 as the corresponding counter to GCCnm register and TIGnm/TOGnm-pin 1: Set TMGn1 as the corresponding counter to GCCnm register and TIGnm/TOGnm-pin
11 to 0	IEGny1, IEGny0	Specifies the valid edge of external capture signal input pin (TIGnm) for the capture register performing capture-match with the assigned counter TMGn0 or TMGn1:

IEGny1	IEGny0	Valid edge
0	0	Falling edge
0	1	Rising edge
1	0	No edge detection performed
1	1	Both rising and falling edges

(3) OCTLGn - Timer Gn output control register

This register controls the timer output from the TOGnm pin and the capture or compare modus for the GCCnm register.

Access This register can be read/written in 16-bit, 8-bit or 1-bit units. The low byte OCTLGn.bit[7:0] is accessible separately under the name OCTLGnL, the high byte OCTLGn.bit[15:8] under the name OCTLGnH.

Address
OCTLGn, OCTLGnL: <base> + 4_H
OCTLGnH: <base> + 5_H

Initial Value 4444_H. This register is cleared by any reset.

15	14	13	12	11	10	9	8
SWFGn4	ALVGn4	CCSGn4	0	SWFGn3	ALVGn3	CCSGn3	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
SWFGn2	ALVGn2	CCSGn2	0	SWFGn1	ALVGn1	CCSGn1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution

1. When the POWERn bit is set, the rewriting of CCSGnm is prohibited
2. When the POWERn bit and TMGn0E bit (TMGn1E bit) are set at the same time, the rewriting of the ALVGnm bits is prohibited.

Table 13-6 OCTLGn register contents

Bit position	Bit name	Function
15, 11, 7, 3	SWFGnm	Fixes the TOGnm pin output level according to the setting of ALVGnm bit. 0: disable TOGnm to inactive level 1: enable TOGnm
14, 10, 6, 2	ALVGnm	Specifies the active level of the TGOnm pin output. 0: Active level is 0 1: Active level is 1 **Caution:** Don’t write this bit, before ENFGn0 or ENFGn1 of TMGSTn is 0, so first clear TMGn0E or TMGn1E bit of the TMGMn register and check ENFGn0 or ENFGn1 bit before writing.
13, 9, 5, 1	CCSGnm	Specifies Capture/Compare mode selection: 0: Capture mode: if external edge is detected the INTCCGnm interrupt occurs, the corresponding counter value is written to GCCnm 1: Compare mode: if GCCnm matches with corresponding timebase the INTCCGnm interrupt occurs, if SWFGm is set the PWM output mode is set **Caution:** Don’t write this bit, before POWERn bit of TMGMnH is 0.

(4) TMGSTn - Time base status register

The TMGSTn register indicates the status of TMGn0 and TMGn1. For the CCFGny bit see *“Operation in Free-Run Mode“ on page 451*.

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

Address <base> + 6_H

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

7	6	5	4	3	2	1	0
ENFGn1	ENFGn2	CCFGn5	CCFGn4	CCFGn3	CCFGn2	CCFGn1	CCFGn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 13-7 **TMGSTn register contents**

Bit position	Bit name	Function
5 to 0	CCFGny	Indicates TMGn0 or TMGn1 overflow status. 0: No overflow 1: Overflow **Caution:** The CCFGny bit is set if a TMGnx overflow has occurred between two capture input signals. This flag is only updated if the corresponding GCCny register was read, so first read the GCCny register and then read this flag if necessary
7 to 6	ENFGnx	Indicates TMGnx operation. 0: indicates operation stopped 1: indicates operation

(5) TMGn0, TMGn1 - Timer Gn 16-bit counter registers

The features of the counters TMGn0 and TMGn1 are listed below:

- Free-running counter that enables counter clearing by compare match of registers GCCn0/GCCn5
- Counter clear can be set by software.
- Counter stop can be set by software.

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

Address TMGn0: <base> + 8_H
TMGn1: <base> + A_H

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

<table>
<tr><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 colspan="16">TMGn0/TMGn1 value</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

(6) GCCn0, GCCn5 - Timer Gn capture/compare registers of the 2 counters

The GCCn0, GCCn5 registers are 16-bit capture/compare registers of Timer Gn. These registers are fixed assigned to the counter registers:

- GCCn0 is fixed assigned to timebase TMGn0
- GCCn5 is fixed assigned to timebase TMGn1

Capture mode In the *capture register mode*, GCCn0 (GCCn5) captures the TMGn0 (TMGn1) count value if an edge is detected at Pin TIGn0 (TIGn5).

Compare mode In the *compare register mode*, GCCn0 (GCCn5) detects match with TMGn0 (TMGn1) and clears the assigned Timebase. So this “match and clear mode” is used to reduce the number of valid bits of the counter TMGn0 (TMGn1).

Caution If in Compare Mode write to this registers *before* POWERn and ENFGnx bit are "1" at the same time.

Access In capture mode, these registers can be read in 16-bit units.
In compare mode, these registers can be read/written in 16-bit units.

Address GCCn0: <base> + C_H
GCCn5: <base> + 16_H

Initial Value 0000_H. These registers are cleared by any reset.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">GGCn0/GGCn5 value</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

(7) GCCn1 to GCCn4 - Timer G capture/compare registers with external PWW-output function

The GCCn1 to GCCn4 registers are 16-bit capture/compare registers of Timer Gn. They can be assigned to one of the two counters either TMGn0 or TMGn1.

Capture mode In the capture register mode, these registers capture the value of TMGn0 when the TBGnm bit (m = 1 to 4) of the TMGCMnH register = 0. When the TBGnm bit = 1, these registers hold the value of TMGn1.

Compare mode In compare mode, these registers represent the actual compare value and the TOGnm-Output (m = 1 to 4) can generate a PWW if they are activated.

Access In capture mode, these registers can be read in 16-bit units.
In compare mode, these registers can be read/written in 16-bit units.

Address
GCCn1: <base> + E_H
GCCn2: <base> + 10_H
GCCn3: <base> + 12_H
GCCn4: <base> + 14_H

Initial Value 0000_H. These registers are cleared by any reset.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">GGCn1 to GGCn4 value</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

13.5 Output Delay Operation

When the OLDEn bit is set, different delays of count clock period are added to the TOGnm pins:

Output pin	Delay $1/f_{COUNT}$
TOGn1	0
TOGn2	1
TOGn3	2
TOGn4	3

The figure below shows the timing for the case where the count clock is set to $f_{SPCLK0}/2$. However, 0FFFH is set in GCCn0.

Similar delays are added also when a transition is made from the active to inactive level. So, a relative pulse width is guaranteed.

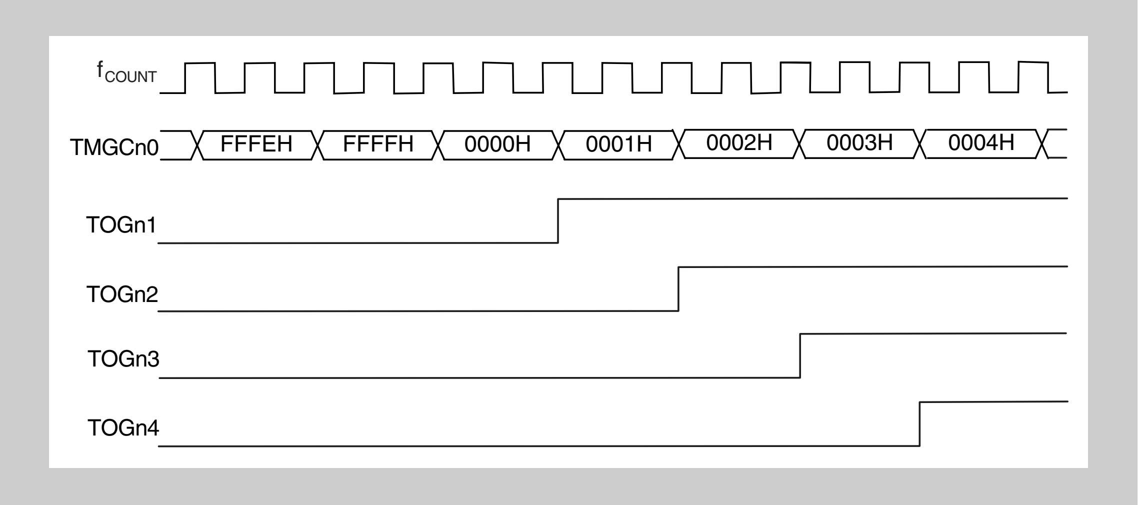


Figure 13-2 Timing of Output delay operation

In this case the count clock is set to $f_{SPCLK0}/2$.

13.6 Explanation of Basic Operation

(1) Overview of the mode settings

The Timer Gn includes 2 channels of 16-bit counters (TMGn0/TMGn1), which can operate as independently timebases. TMGn0 (TMGn1) can be set by CCSGn0 bit (CCSGn5 bit) in the following modes:

- free-run mode
- match and clear mode

When a timer output (TOGnm) or INTCCGnm interrupt is used, one of the two counters can be selected by setting the TBGnm bit (m = 1 to 4) of the TMGCMHn register.

The tables below indicate the interrupt output and timer output states dependent on the register setting values.

Table 13-8 Interrupt output and timer output states dependent on the register setting values

Register setting value				State of each output pin			
CCSGn0	**TBGnm**	**SWFGnm**	**CCSGnm**	**INTTMGn0**	**INTCCGn0**	**INTCCGnm**	**TOGnm**
0 Free-run mode	0	0	0	Overflow interrupt	TI0 edge detection	TIm edge detection	Tied to inactive level
			1			GCCnm match	
		1	0			TIm edge detection	
			1			CMPGm match	PWM (free run)
1 Match and clear mode		0	0	Overflow interrupt[Note 1]	GCCn0 match[Note 2]	TIm edge detection	Tied to inactive level
			1			GCCnm match	
		1	0			TIm edge detection	
			1			CMPGm match	PWM (match and clear)

Note

1. An interrupt is generated only when the value of the GCCn0 register is FFFFH.
2. An interrupt is generated only when the value of the GCCn0 register is not FFFFH.
3. The setting of the CCSGnm bit in combination with the SWFGnm bit sets the mode for the timing of the actualization of new compare values.
    - In compare mode the new compare value will be immediately active.
    - In PWM mode the new compare value will be active first after the next overflow or match & clear of the assigned counter (TMGn0, TMGn1).

Table 13-9 Interrupt output and timer output states dependent on the register setting values

Register setting value				State of each output pin			
CCSGn5	**TBGnm**	**SWFGnm**	**CCSGnm**	**INTTMGn1**	**INTCCGn5**	**INTCCGnm**	**TOGnm**
0 Free-run mode	1	0	0	Overflow interrupt	TI5 edge detection	TIm edge detection	Tied to inactive level
			1			GCCnm match	
		1	0			TIm edge detection	
			1			CMPGm match	PWM (free run)
1 Match and clear mode		0	0	Overflow interrupt[Note 1]	GCCn5 match[Note 2]	TIm edge detection	Tied to inactive level
			1			GCCnm match	
		1	0			TIm edge detection	
			1			CMPGm match	PWM (match and clear)

Note
1. An interrupt is generated only when the value of the GCCn5 register is FFFFH.
2. An interrupt is generated only when the value of the GCCn5 register is not FFFFH.
3. The setting of the CCSGnm bit in combination with the SWFGnm bit sets the mode for the timing of the actualization of new compare values.
   - In compare mode the new compare value will be immediately active.
   - In PWM mode the new compare value will be active first after the next overflow or match & clear of the assigned counter (TMGn0, TMGn1).

13.7 Operation in Free-Run Mode

This operation mode is the standard mode for Timer Gn operations. In this mode the 2 counter TMGn0 and TMGn1 are counting up from 0000H to FFFFH, generates an overflow and start again. In the match and clear mode, which is described in Chapter *13.8 on page 462* the fixed assigned register GCCn0 (GCCn5) is used to reduce the bit-size of the counter TMGn0 (TMGn1).

(1) Capture operation (free run)

Basic settings:

Bit	Value	Remark
CCSGn0	0	free run mode
CCSGn5	0	
SWFGnm	0	disable TOGnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Pulse width or period measurement of the TIGny input signal (free run)

Capture setting method:

(1) When using one of the TOGn1 to TOGn4 pins, select the corresponding counter with the TBGnm bit. When TIGn0 is used, the corresponding counter is TMGn0. When TIGn5 is used, the corresponding counter is TMGn1.

(2) Select a count clock cycle with the CSE12 to CSE10 bits (TMGn1) or CSE02 to CSE02 bits (TMGn0).

(3) Select a valid TIGny edge with the IEGny1 and IEGny0 bits. A rising edge, falling edge, or both edges can be selected.

(4) Start timer operation by setting POWERn bit and TMGn0E bit for TMGn0 or TMGn1E bit for TMGn1.

Capture operation:

(1) When a specified edge is detected, the value of the counter is stored in GCCny and an edge detection interrupt (INTCCGny) is output.

(2) When the counter overflows, an overflow interrupt (INTTMGn0 or INTTMGn1) is generated.

(3) If an overflow has occurred between capture operations, the CCFGny flag is set when GCCny is read. Correct capture data by checking the value of CCFGny.

Using CCFGny:

When using GCCny as a capture register, use the procedure below.

<1> After INTCCGny (edge detection interrupt) generation, read the corresponding GCCny register.

<2> Check if the corresponding CCFGny bit of the TMGSTn register is set.

<3> If the CCFGny bit is set, the counter was cleared from the previous captured value.

CCFGny is set when GCCny is read. So, after GCCny is read, the value of CCFGny should be read. Using the procedure above, the value of CCFGny corresponding to GCCny can be read normally.

Caution If two or more overflows occur between captures, a software-based measure needs to be taken to count overflow interrupts (INTTMGn0, INTTMGn1).

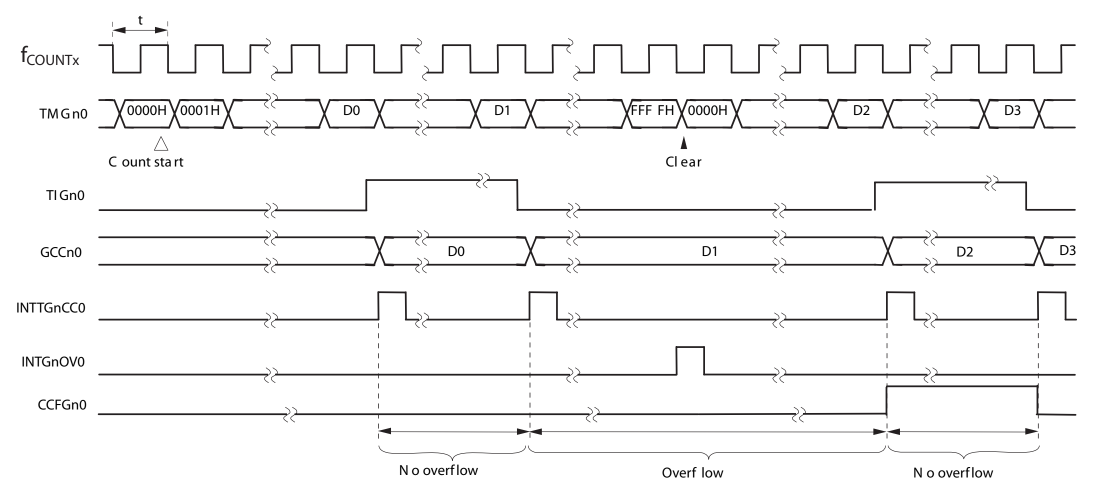


Figure 13-3 Timing when both edges of TIGn0 are valid (free run)

Note The figure above shows an image. In actual circuitry, 3 to 4 periods of the count-up signal are required from the input of a waveform to TIGn0 until a capture interrupt is output.

(b) Timing of capture trigger edge detection

The Tin inputs are fitted with an edge-detection and noise-elimination circuit.

Because of this circuit, 3 periods to less than 4 periods of the count clock are required from edge input until an interrupt signal is output and capture operation is performed. The timing chart is shown below.

Basic settings (x = 0, 1 and y = 0 to 5):

Bit	Value	Remark
CSEx2	0	Count clock = $f_{SPCLK0}/4$
CSEx1	1	
CSEx0	0	
IEGny1	1	detection of both edges
IEGny0	1	

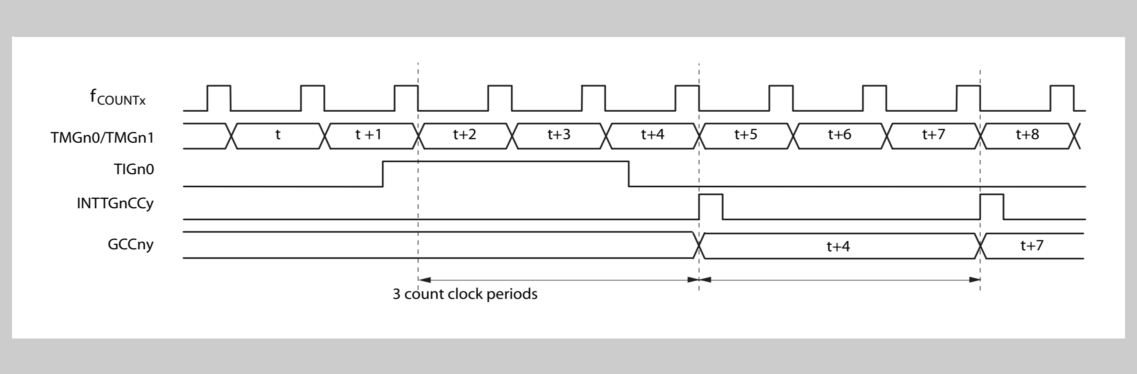


Figure 13-4 Timing of capture trigger edge detection (free run)

(c) Timing of starting capture trigger edge detection

A capture trigger input signal (TIGny) is synchronized in the noise eliminator for internal use.

Edge detection starts when 1 count clock period (f_{COUNT}) has been input after timer count operation starts. (This is because masking is performed to prevent the initial TIGny level from being recognized as an edge by mistake.). The timing chart for starting edge detection is shown below.

Basic settings (x = 0, 1 and y = 0 to 5):

Bit	Value	Remark
CSEx2	0	Count clock = $f_{SPCLK0}/4$
CSEx1	1	
CSEx0	0	
IEGny1	1	detection of both edges
IEGny0	1	

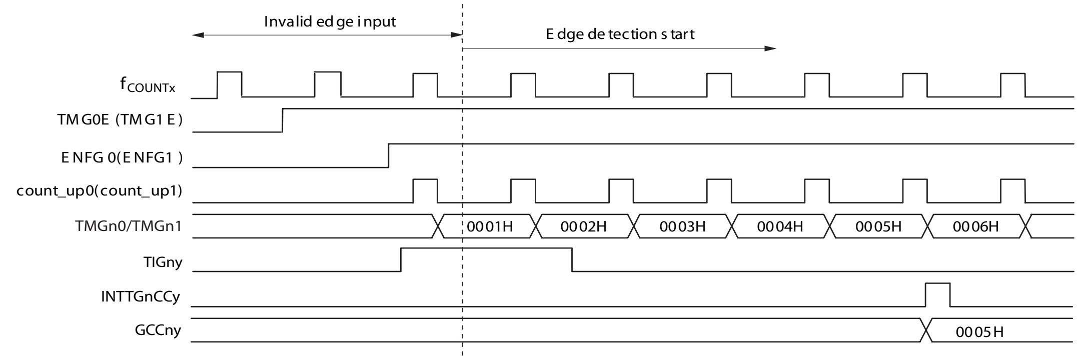


Figure 13-5 Timing of starting capture trigger edge detection

(2) Compare operation (free run)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	0	free run mode
CCSGn5	0	
SWFGnm	0	disable TOGnm
CCSGnm	1	Compare mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Interval timer (free run)

Setting method interval timer:

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counter (TMGn0 or TMGn1) must be selected with the TBGnm bit.

(2) Select a count clock cycle with the CSE12 to CSE10 bits (TMGn1 register) or CSE02 to CSE00 bits (TMGn0 register).

(3) Write data to GCCnm.

(4) Start timer operation by setting POWERn and TMGn0E (or TMGn1E).

Compare Operation:

(1) When the value of the counter matches the value of GCCnm (m = 0 to 4), a match interrupt (INTCCGnm) is output.

(2) When the counter overflows, an overflow interrupt (INTTMGn0/ INTTMGn1) is generated.

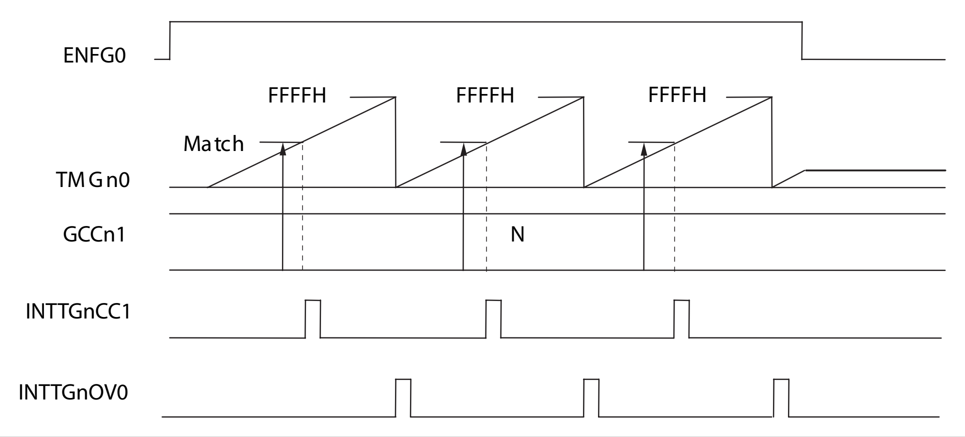


Figure 13-6 Timing of compare mode (free run)

Data N is set in GCCn1, and the counter TMGn0 is selected.

(b) When the value 0000H is set in GCCnm

INTCCGnm is activated when the value of the counter becomes 0001H.

INTTMGn0/INTTMGn1 is activated when the value of the counter changes from FFFFH to 0000H.

Note, however, that even if no data is set in GCCnm, INTCCGnm is activated immediately after the counter starts.

(c) When the value FFFFH is set in GCCnm

INTCCGnm and INTTMGn0/INTTMGn1 are activated when the value of the counter changes from FFFFH to 0000H.

(d) When GCCnm is rewritten during operation

When GCCn1 is rewritten from 5555H to AAAAH. TMGn0 is selected as the counter.

The following operation is performed:

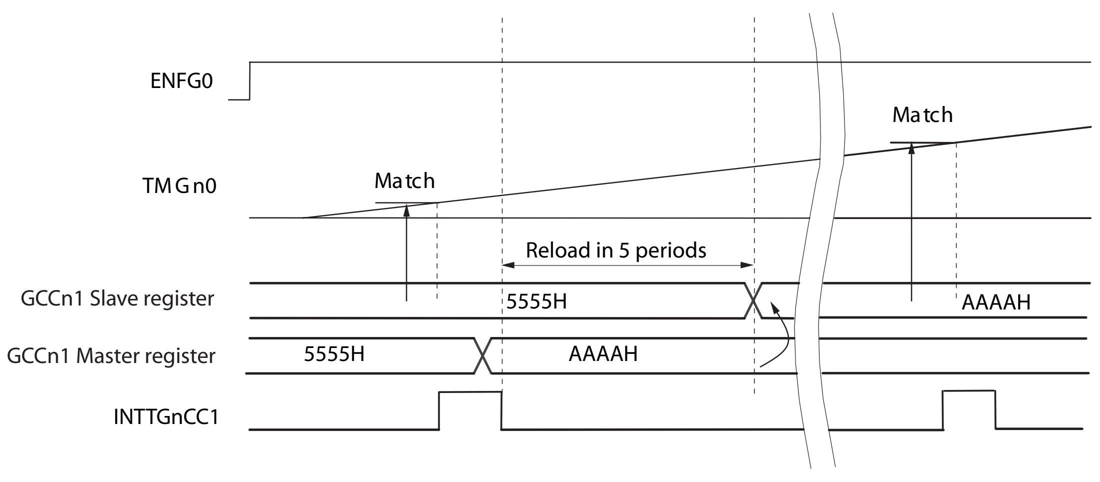


Figure 13-7 Timing when GCCn1 is rewritten during operation (free run)

Caution To perform successive write access during operation, for rewriting the GCCny register (n = 1 to 4), you have to wait for minimum 7 peripheral clocks periods (f_{SPCLK0}).

(3) PWM output (free run)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	0	free run mode
CCSGn5	0	
SWFGnm	1[Note]	enable TOGnm
CCSGnm	1[Note]	Compare mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

Note The PWM mode is activated by setting the SWFGnm and the CCSGnm bit to "1".

PWM setting method:

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counter must be selected with the TBGnm bit.

(2) Select a count clock cycle with the CSE12 to CSE10 bits (TMGn1 register) or CSE02 to CSE00 bits (TMGn0 register).

(3) Specify the active level of a timer output (TOGnm pin) with the ALVGnm bit.

(4) When using multiple timer outputs, the user can prevent TOGnm from becoming active simultaneously by setting the OLDEn bit of TMGMHn register to provide step-by-step delays for TOGnm. (This capability is useful for reducing noise and current.)

(5) Write data to GCCnm.

(6) Start timer operation by setting POWERn bit and TMGn0E bit (or TMGn1E bit).

PWM operation:

(1) When the value of the counter matches the value of GCCnm, a match interrupt (INTCCGnm) is output.

(2) When the counter overflows, an overflow interrupt (INTTMGn0 or INTTMGn1) is generated.

(3) TOGnm does not make a transition until the first overflow occurs. (Even if the counter is cleared by software, TOGnm does not make a transition until the next overflow occurs. After the first overflow occurs, TOGnm is activated.

(4) When the value of the counter matches the value of GCCnm, TOGnm is deactivated, and a match interrupt (INTCCGnm) is output. The counter is not cleared, but continues count-up operation.

(5) The counter overflows, and INTTMGn0 or INTTMGn1 is output to activate TOGnm. The counter resumes count-up operation starting with 0000H.

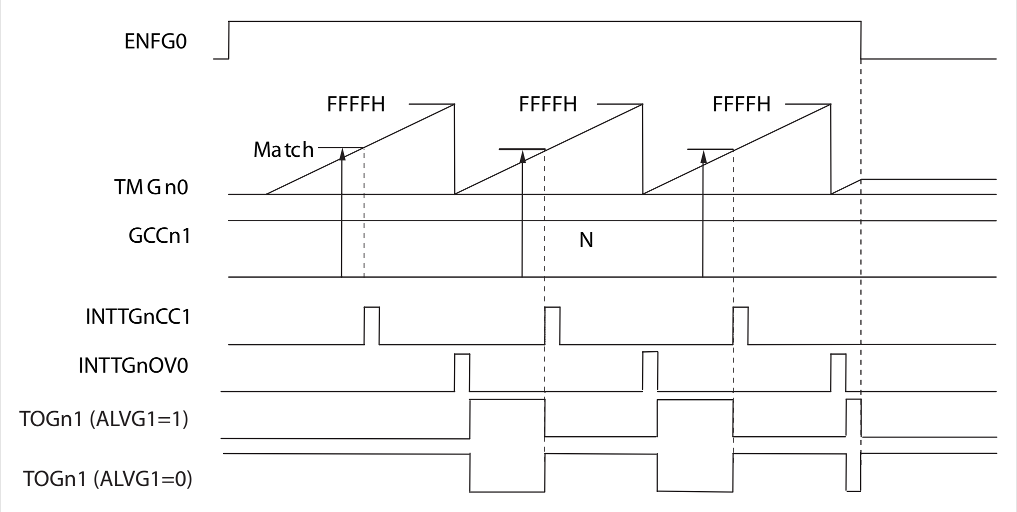


Figure 13-8 Timing of PWM operation (free run)

Data N is set in GCCn1, counter TMGn0 is selected.

(a) When 0000H is set in GCCnm (m = 1 to 4)

When 0000H is set in GCCnm, TOGnm is tied to the inactive level.

The figure below shows the state of TOGn1 when 0000H is set in GCCn1, and TMGn0 is selected.

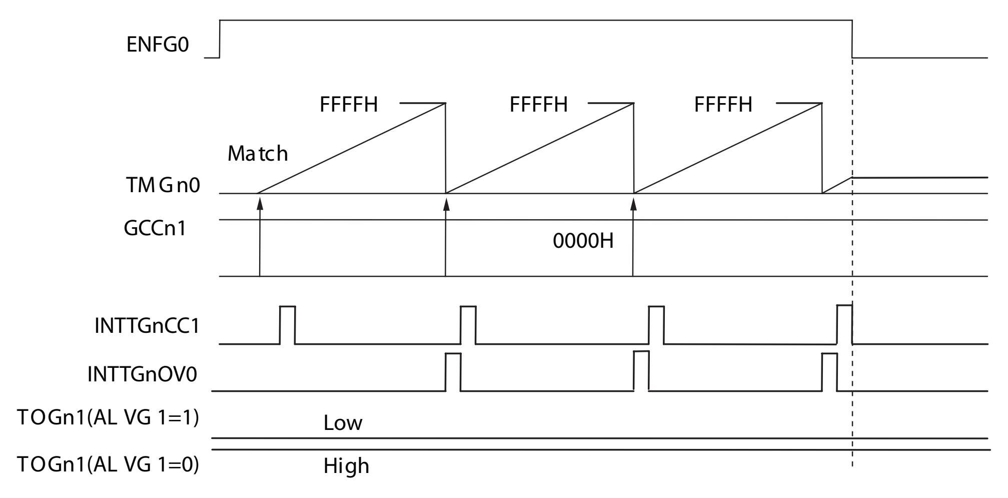


Figure 13-9 Timing when 0000H is set in GCCnm (free run)

GCCn1 and TMGn0 are selected.

(b) When FFFFH is set in GCCnm (m = 1 to 4)

When FFFFH is set in GCCnm, TOGnm outputs the inactive level for one clock period immediately after each counter overflow (except the first overflow).

The figure shows the state of TOGn1 when FFFFH is set in GCCn1, and TMGn0 is selected.

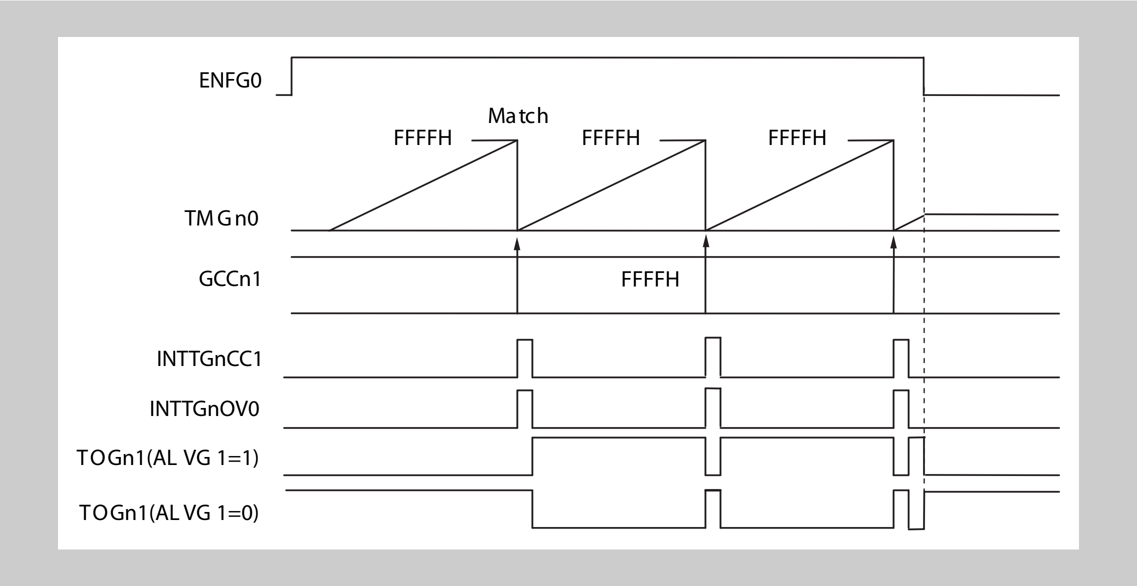


Figure 13-10 Timing when FFFFH is set in GCCnm (free run)

GCCn1 and TMGn0 are selected.

(c) When GCCnm is rewritten during operation (m = 1 to 4)

When GCCn1 is rewritten from 5555H to AAAAH, the operation shown below is performed.

The figure below shows a case where TMGn0 is selected for GCCn1.

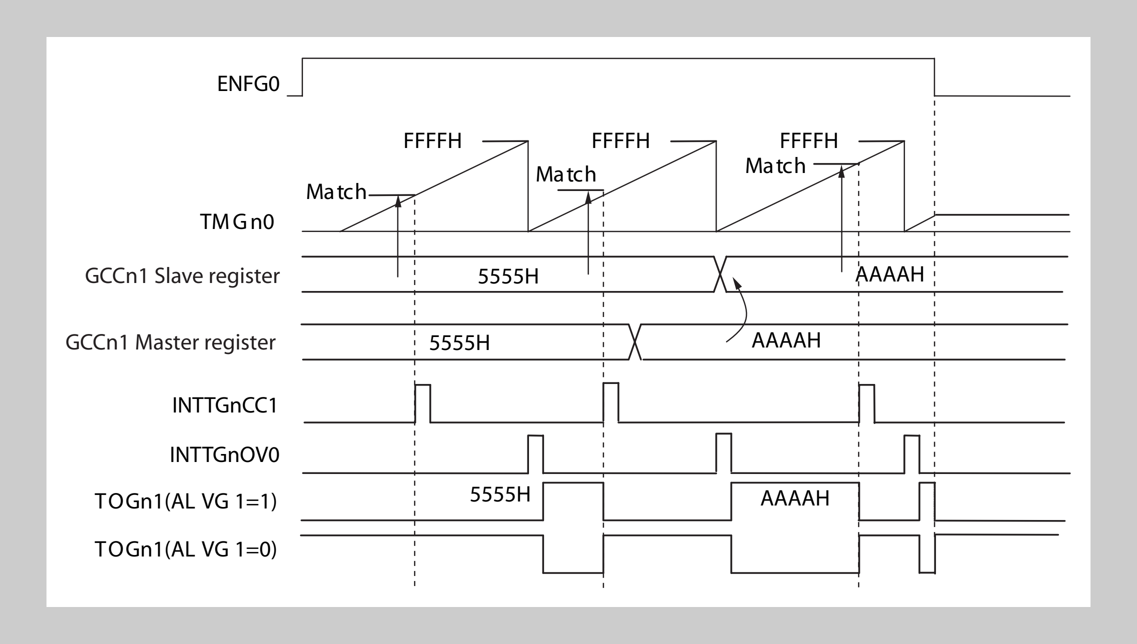


Figure 13-11 Timing when GCCnm is rewritten during operation (free run)

GCCn1 and TMGn0 are selected.

If GCCn1 is rewritten to AAAAH after the second INTCCGn1 is generated as shown in the figure above, AAAAH is reloaded to the GCCn1 register when the next overflow occurs.

The next match interrupt (INTCCGn1) is generated when the value of the counter is AAAAH. The pulse width also matches accordingly.

13.8 Match and Clear Mode

The match and clear mode is mainly used reduce the number of valid bits of the counters (TMGn0, TMGn1).

Therefore the fixed assigned register GCCn0 (GCCn1) is used to compare its value with the counter TMGn0 (TMGn1). If the values match, than an interrupt is generated and the counter is cleared. Than the counter starts up counting again.

(1) Capture operation (match and clear)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	1	match and clear mode
CCSGn5	1	
SWFGnm	0	disable TOGnm
CCSGnm	0	Capture mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Pulse width measurement or period measurement of the TIGnm input signal

Setting method:

(1) When using one of TOGn1 to TOGn4-pin, select the corresponding counter with the TBGnm bit. When CCSGn0 = 1, TI0 cannot be used. When CCSGn5 = 1, TIGn5 cannot be used.

(2) Select a count clock cycle with the CSE12 to CSE10 (TMGn1) bits or CSE02 to CSE00 (TMGn0) bits.

(3) Select a valid TIGnm edge with the IEGnm1 and IEGnm0 bit. A rising edge, falling edge, or both edges can be selected.

(4) Set an upper limit on the value of the counter in GCCn0 or GCCn5.

(5) Start timer operation by setting POWERn bit and TMGn0E bit (or TMGn1E bit).

Operation:

(1) When a specified edge is detected, the value of the counter is stored in GCCnm, and an edge detection interrupt (INTCCGnm) is output.

(2) When the value of GCCn0 or GCCn5 matches the value of the counter, INTCCGn0 (INTCCGn5) is output, and the counter is cleared. This operation is referred to as "match and clear".

(3) If a match and clear event has occurred between capture operations, the CCFGny flag is set when GCCny is read. Correct capture data by checking the value of CCFGny.

(b) Example: Capture where both edges of TIGnm are valid (match and clear)

For the timing chart TMGn0 is selected as the counter corresponding to TOGn1, and 0FFFH is set in GCCn0.

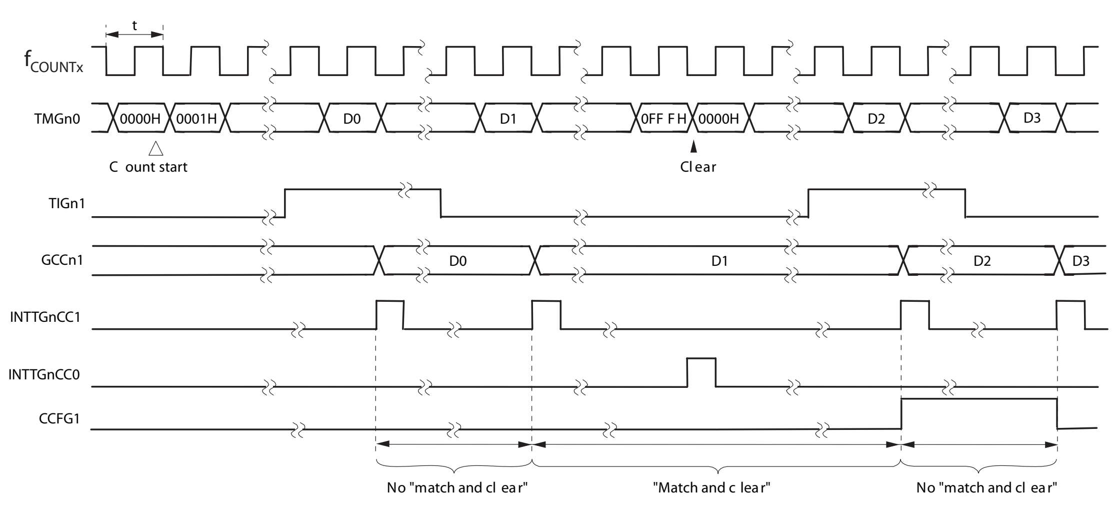


Figure 13-12 Timing when both edges of TIGnm are valid (match and clear)

Note The figure above shows an image. In actual circuitry, 3 to 4 periods of the count-up signal (f_{COUNT}) are required from the input of a waveform to TOGn1 until a capture interrupt is output. (See *Figure 13-4 on page 454*.)

Caution If two or more match and clear events occur between captures, a software-based measure needs to be taken to count INTCCGn0 or INTCCGn5.

(c) When 0000H is set in GCCn0 or GCCn5 (match and clear)

When 0000H is set in GCCn0 (GCCn5), the value of the counter is fixed at 0000H, and does not operate. Moreover, INTCCGn0 (INTCCGn5) continues to be active.

(d) When FFFFH is set in GCCn0 or GCCn5 (match and clear)

When FFFFH is set in GCCn0 (GCCn5), operation equivalent to the free-run mode is performed. When an overflow occurs, INTTMGn0 (INTTMGn1) is generated, but INTCCGn0 (INTCCGn5) is not generated.

(2) Compare operation (match and clear)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	1	match and clear mode
CCSGn5	1	
SWFGnm	0	disable TOGnm
CCSGnm	1	Compare mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Interval timer (match and clear)

Setting Method

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counter must be selected with the TBGnm bit.

(2) Select a count clock cycle with the CSE12 to CSE10 bits (TMGn1) or CSE02 to CSE00 bits (TMGn0).

(3) Set an upper limit on the value of the counter in GCCn0 or GCCn5.

(4) Write data to GCCnm.

(5) Start timer operation by setting the POWERn bit and TMGxE bit (x = 0, 1).

Operation:

(1) When the value of the counter matches the value of GCCnm, a match interrupt (INTCCGnm) is output.

(2) When the value of GCCn0 or GCCn5 matches the value of the counter, INTCCGn0 (or INTCCGn5) is output, and the counter is cleared. This operation is referred to as "match and clear".

(3) The counter resumes count-up operation starting with 0000H.

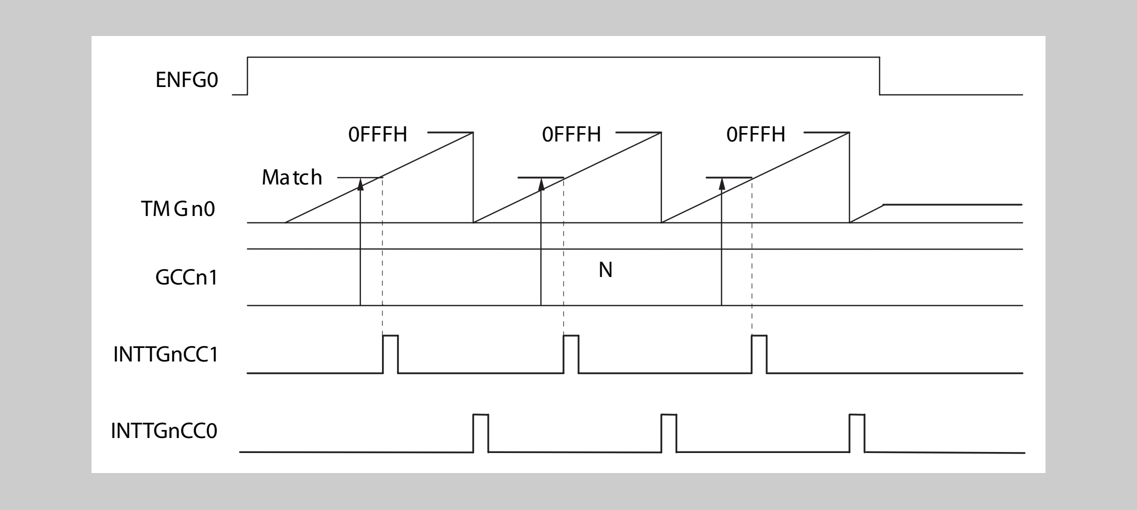


Figure 13-13 Timing of compare operation (match and clear)

In this example, the data N is set in GCCn1, and TMGn0 is selected.

0FFFH is set in GCCn0. Here, N < 0FFFH.

(b) When 0000H is set in GCCn0 or GCCn5 (match and clear)

When 0000H is set in GCCn0 or GCCn5, the value of the counter is fixed at 0000H, and does not operate. Moreover, INTCCGn0 (or INTCCGn5) continues to be active.

(c) When FFFFH is set in GCCn0 or GCCn5 (match and clear)

When FFFFH is set in GCCn0 or GCCn5, operation equivalent to the free-run mode is performed. When an overflow occurs, INTTMGn0 (or INTTMGn1) is generated, but INTCCGn0 (or INTCCGn5) is not generated.

(d) When 0000H is set in GCCnm (m = 1 to 4) (match and clear)

INTCCGnm is activated when the value of the counter becomes 0001H.

Note, however, that even if no data is set in GCCnm, INTCCGnm is activated immediately after the counter starts.

(e) When a value exceeding the value of GCCn0 or GCCn5 is set in GCCnm (m = 1 to 4) (match and clear)

INTCCGnm is not generated.

(f) When GCCnm (m = 1 to 4) is rewritten during operation (match and clear)

When the value of GCCn1 is changed from 0555H to 0AAAH, the operation described below is performed.

TMGn0 is selected as the counter, and 0FFFH is set in GCCn0.

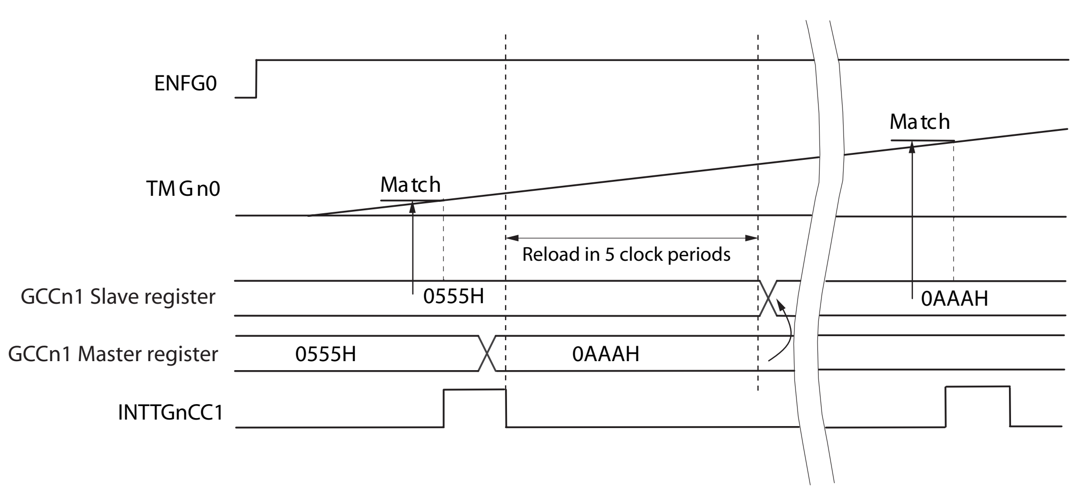


Figure 13-14 Timing when GCCnm is rewritten during operation (match and clear)

Caution To perform successive write access during operation, for rewriting the GCCny register, you have to wait for minimum 7 peripheral clocks periods (f_{SPCLK0}).

(3) PMW output (match and clear)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	1	match and clear mode
CCSGn5	1	
SWFGnm	1^{Note}	enable TOGnm
CCSGnm	1^{Note}	Compare mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

Note The PWM mode is activated by setting the SWFGnm and the CCSGnm bit to "1".

Setting Method:

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counters TMGn0 or TMGn1 must be selected with the TBGnm bit (m = 1 to 4).

(2) Select a count clock cycle with the CSE12 to CSE10 (TMGn1) bits or CSE02 to CSE00 (TMGn0) bits.

(3) Specify the active level of a timer output (TOGnm) with the ALVGnm bit.

(4) When using multiple timer outputs, the user can prevent TOGnm from making transitions simultaneously by setting the OLDEn bit of TMGMHn register. (This capability is useful for reducing noise and current.)

(5) Set an upper limit on the value of the counter in GCCn0 or GCCn5. (Timer Dn 0000H is forbidden)

(6) Write data to GCCnm.

(7) Start count operation by setting POWERn bit and TMGn0E bit (or TMGn1E bit).

Operation of PWM (match and clear):

(1) When the value of the counter matches the value of GCCnm, a match interrupt (INTCCGnm) is output.

Caution Do not set 0000H in GCCn0 or GCCn5 in match and clear modus.

(2) When the value of GCCn0 (GCCn5) matches the value of the counter, INTCCGn0 (INTCCGn5) is output, and the counter is cleared. This operation is referred to as "match and clear".

(3) TOGnm does not make a transition until the first match and clear event.

(4) TOGnm makes a transition to the active level after the first match and clear event.

(5) When the value of the counter matches the value of GCCnm, TOGnm makes a transition to the inactive level, and a match interrupt (INTCCGnm) is output.

(6) When the next match and clear event occurs, INTCCGn0 (INTCCGn5) is output, and the counter is cleared. The counter resumes count-up operation starting with 0000H.

Example where the data N is set, and the counter TMGn0 is selected.

0FFFH is set in GCCn0 and N < 0FFFH.

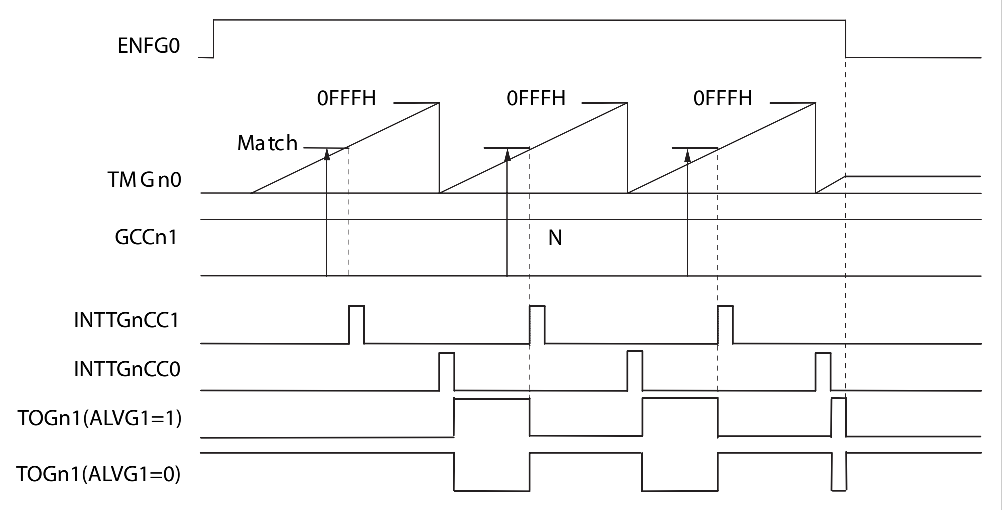


Figure 13-15 Timing of PWM operation (match and clear)

When 0000H is set in GCCn0 (GCCn5), the value of the counter is fixed at 0000H, and the counter does not operate. The waveform of INTCCGn0 (INTCCGn5) varies, depending on whether the count clock is the reference clock or the sampling clock.

(a) When FFFFH is set in GCCn0 or GCCn5 (match and clear)

When FFFFH is set in GCCn0 (GCCn5), operation equivalent to the free-run mode is performed. When an overflow occurs, INTTMGn0 (INTTMGn1) is generated, but INTCCGn0 (INTCCGn5) is not generated.

(b) When 0000H is set in GCCnm (match and clear)

When 0000H is set in GCCnm, TOGnm is tied to the inactive level.

The figure below shows the state of TOGn1 when 0000H is set in GCCn1, and TMGn0 is selected.

Note, however, that 0FFFH is set in GCCn0.

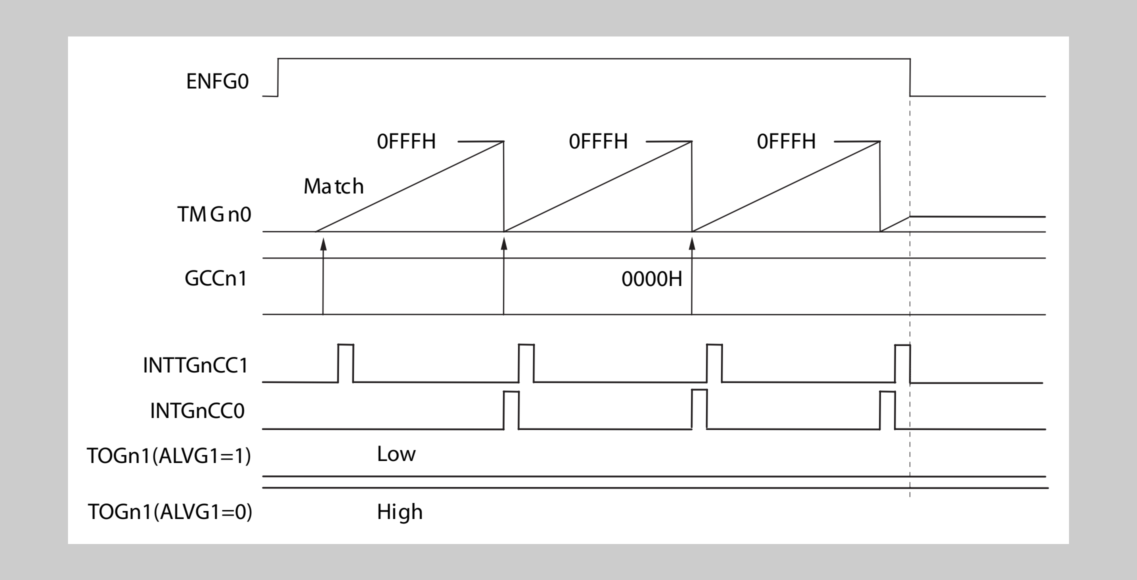


Figure 13-16 Timing when 0000H is set in GCCnm (match and clear)

(c) When the same value as set in GCCn0 or GCCn5 is set in GCCnm (match and clear)

When the same value as set in GCCn0 (GCCn5) is set in GCCnm, TOGnm outputs the inactive level for only one clock period immediately after each match and clear event (excluding the first match and clear event).

The figure below shows the state of TOGn1 when 0FFFH is set in GCCn0 and GCCn1, and TMGn0 is selected.

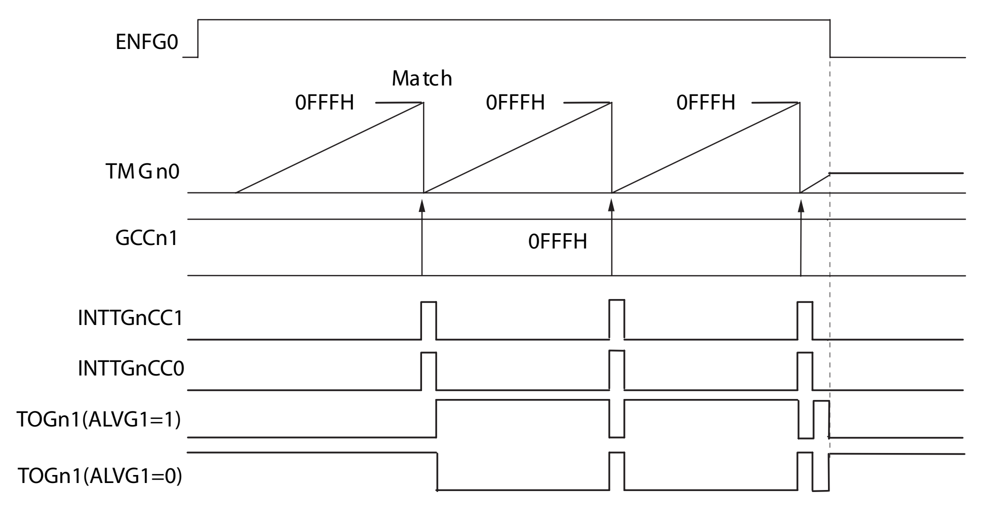


Figure 13-17 Timing when the same value as set in GCCn0/GCCn5 is set in GCCnm (match and clear)

(d) When a value exceeding the value set in GCCn0 or GCCn5 is set in GCCnm (match and clear)

When a value exceeding the value set in GCCn0 (GCCn5) is set in GCCnm, TOGnm starts and continues outputting the active level immediately after the first match and clear event (until count operation stops.)

The figure shows the state of TOGn1 when 0FFFH is set in GCCn0, 1FFFH is set in GCCn1, and TMGn0 is selected.

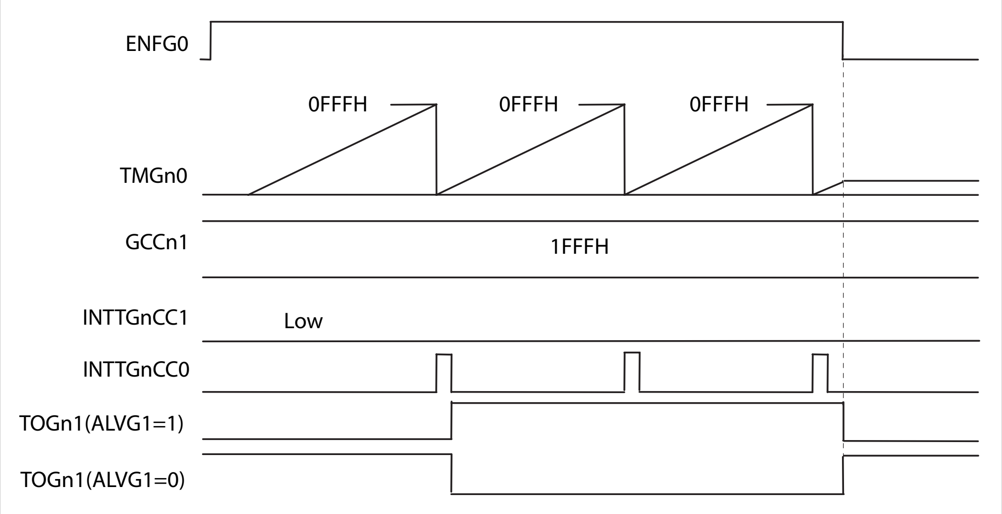


Figure 13-18 Timing when the value of GCCnm exceeding GCCn0 or GCCn5 (match and clear)

(e) When GCCnm is rewritten during operation (match and clear)

When GCCn1 is rewritten from 0555H to 0AAAH, the operation shown below is performed.

The figure below shows a case where 0FFFH is set in GCCn0, and TMGn0 is selected for GCCn1.

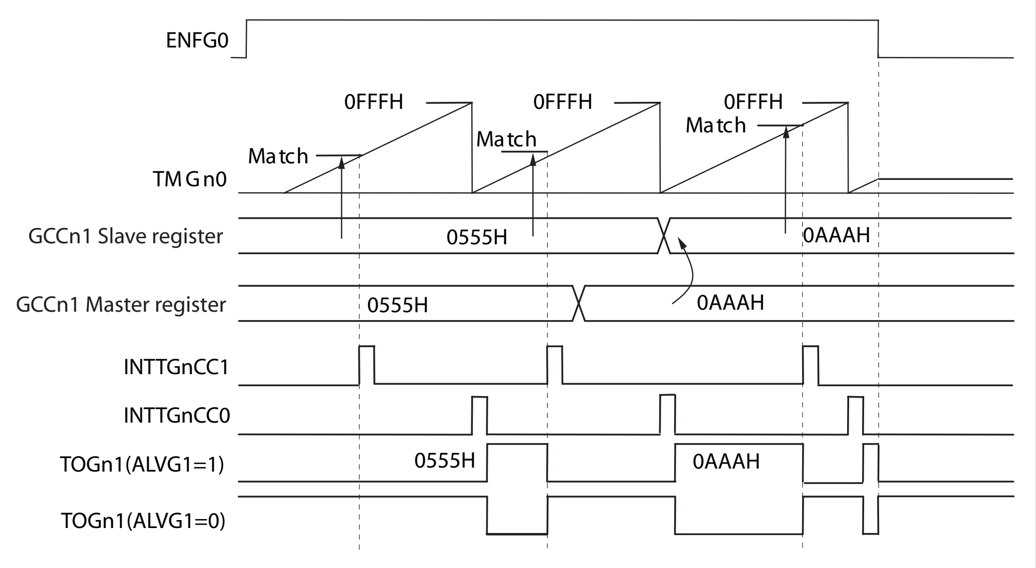


Figure 13-19 Timing when GCCnm is rewritten during operation (match and clear)

If GCCn1 is rewritten to 0AAAH after the second INTCCGn1 is generated as shown in the figure above, 0AAAH is reloaded to the GCCn1 register when the next overflow occurs.

The next match interrupt (INTCCGn1) is generated when the value of the counter is 0AAAH. The pulse width also matches accordingly.

13.9 Edge Noise Elimination

The edge detection circuit has a noise elimination function. This function regards:

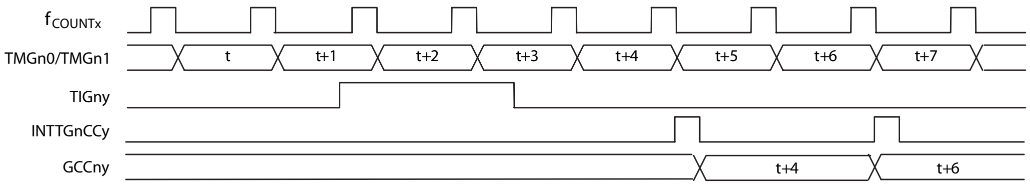
- a pulse **not wider than 1 count clock** period as a **noise**, and does not detect it as an edge.
- a pulse **not shorter than 2 count clock** periods is detected normally as an **edge**.
- a pulse **wider than 1 count clock period but shorter than 2 count clock** periods may be **detected as an edge or may be eliminated as noise**, depending on the timing.

(This is because the count-up signal of the counter is used for sampling timing.) The upper figure below shows the timing chart for performing edge detection. The lower figure below shows the timing chart for not performing edge detection.

Basic settings (x = 0, 1 and y = 0 to 5):

Bit	Value	Remark
CSEx2	0	Count clock = $f_{SPCLK0}/4$
CSEx1	1	
CSEx0	0	
IEGny1	1	detection of both edges
IEGny0	1	

<Timing chart for performing edge detection>



<Timing chart for noise elimination>

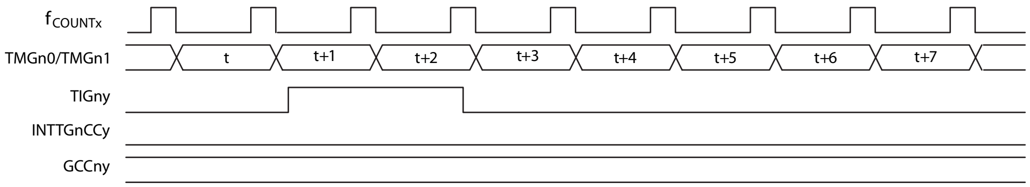


Figure 13-20 Timing of edge detection noise elimination

13.10 Precautions Timer Gn

(1) When POWERn bit of TMGMHn register is set

The rewriting of the CSEn2 to CSEn0 bits of TMGMHn register is prohibited.

These bits set the prescaler for the Timer Gn counter.

The rewriting of the CCSGny bits (y = 0 to 5) is prohibited.

This bits (OCTLGnL and OCTLGnH registers) set the capture mode or the compare mode to the GCCy register. For the GCCn0 register and the GCCn5 register these bits (TMGMLn register) set the “free run” or “match and clear” mode of the TMGn0 and TMGn1 counter.

The rewriting of the TMGCMnL and the TMGCMHn register is prohibited.

These registers configure the counter (TMGn0 or TMGn1) for the GCCnm register (m = 1 to 4) and define the edge detection for the TIGnm input pins (falling, rising, both).

Even when POWERn bit is set, TOGnm output is switched by switching the ALVGnm bit of OCTLGnL and OCTLGnH registers.

These bits configure the active level of the TOGnm-pins (m = 1 to 4).

(2) When POWERn bit and TMGxE bit are set (x = 0, 1)

The rewriting of ALVGnm is prohibited (m = 1 to 4).

These bits configure the active level of the TOGnm-pins (m = 1 to 4).

When in compare-mode the rewriting of the GCCn0 or GCCn5 register is prohibited.

In compare mode these registers set the value for the “match and clear” mode of the TMGn0 and TMGn1 counter.

(3) Functionality

When the POWERn bit is set to “0”, regardless of the SWFGnm bit (OCTLGnL and OCTLGnH registers), the TOGnm pins are tied to the inactive level.

The SWFGnm bit enables or disables the output of the TOGnm pins. This bit can be rewritten during timer operation.

The CLRGx bit (x = 0, 1) is a flag. If this bit is read, a "0" is read at all times.

This bit clears the corresponding counter (TMGn0 or TMGn1)

When GCCnm register (m = 1 to 4) are used in capture operation:

If two or more overflows of TMGn0 or TMGn1 occur between captures, a software-based measure needs to be taken to count overflow interrupts (INTTMGn0 or INTTMGn1).

If only one overflow is necessary, the CCFGny bits (y = 0 to 5) can be used for overflow detection.

Only the overflow of the TMGn0 or TMGn1counter clears the CCFGny bit (TMGSTn register). The software-based clearing via CLRGn0 or CLRGn1 bit (TMGMLn register) doesn't affect these bits.

The CCFGny bit is set if a TMGn0 (TMGn1) overflow occurs. This flag is only updated if the corresponding GCCny register was read, so first read the GCCny register and then read this flag if necessary.

(4) Timing

The delay of each timer output TOGnm (m = 1 to 4) varies according to the setting of the count clock with the CSEx2 to CSEx0 bits (x = 0, 1).

In capture operation 3 to 4 periods of the count-clock (f_{COUNT}) signal are required from the TIGny pin (y = 0 to 5) until a capture interrupt is output.

When TMGxE (x = 0, 1) is set earlier or simultaneously with POWERn bit, than the Timer Gn needs 7 peripheral clocks periods (f_{SPCLK0}) to start counting.

When TMGxE (x = 0, 1) is set later than POWERn bit, than the Timer Gn needs 4 peripheral clocks periods (f_{SPCLK0}) to start counting.

When a capture register (GCCny) is read, the capturing is disable during read operation. This is intended to prevent undefined data during reading. So, if a contention occurs between an external trigger signal and the read operation, capture operation may be cancelled, and old data may be read.

GCCnm register (m = 1 to 4) in Compare mode:

After setting the POWERn bit you have to wait for 10 peripheral clocks periods (f_{SPCLK0}) to perform write access to the GCCnm register (m = 1 to 4).

To perform successive write access during operation, for rewriting the GCCnm register, you have to wait for minimum 7f peripheral clocks periods (f_{SPCLK0}).

Chapter 14 Watch Timer (WT)

The Watch Timer (WT) generates interrupts at regular time intervals. These interrupts are generally used as ticks for updating the internal daytime and calendar.

The Watch Timer includes two identical counters. Throughout this chapter, the counters are identified as WTn, where n = 0 to 1.

14.1 Overview

The Watch Timer consists of two 16-bit down-counters, WT0 and WT1, and includes the Watch Calibration Timer WCT.

WT0 The load value that must be set for WT0 depends on the chosen clock frequency and the desired time interval between two interrupts.

For example, WT0 can be set up to generate an interrupt every second (INTWT0UV).

During normal operation, the clock of WT0 (WTCLK) is directly derived from the precision main oscillator. It bypasses the PLL and SSCG.

However, the WTCLK can also be derived from the sub or ring oscillator. This is useful when the main oscillator is switched off in order to save power.

WT1 WT1 is clocked by the interrupts generated by WT0. It can, for example, generate an interrupt every hour (or whatever wake-up time is required).

This interrupt (INTWT1UV) can be used to escape from Sub-WATCH mode and hence to revive the main oscillator if necessary.

WCT The sub or ring oscillators used in Sub_WATCH mode are not as stable as the main oscillator. The time between two WT0 interrupts may be slightly shorter or longer than desired.

Therefore a third timer - the Watch Calibration Timer (WCT) - can be used occasionally to measure the time between two interrupts INTWT0UV.

WCT requires the main oscillator clock for this measurement. Its clock, WCTCLK, always stops if the main oscillator stops, that means if STOP mode or Sub-WATCH mode are entered.

Based on the measurement result, a new load value for WT0 can be calculated. This is the solution to regain precise intervals between WT0 interrupts. After the adjustment of WT0, the system can return to Sub-WATCH mode where the main oscillator is stopped.

Features summary

Special features of the Watch Timer are:

- Periodic interrupts (clock ticks) generated by two down-counters
- Two reload registers, one for each counter
- Choice of oscillators to reduce power consumption in stand-by mode
- Can operate in all power save modes
- Clock correction in stand-by mode by means of the Watch Calibration Timer
- In debug mode, the counters WT0 and WT1 can be stopped at breakpoint

Special features of the Watch Calibration Timer are:

- 16-bit counter register TM00
- 16-bit capture / compare register CR000
- Capture / trigger input for INTWT0UV with edge specification for INTWT0UV interval measurement
- Capture / match interrupt request signal INTTM00

14.1.1 Description

The following figure shows the structure of the Watch Timer and its connection to the Watch Calibration Timer.

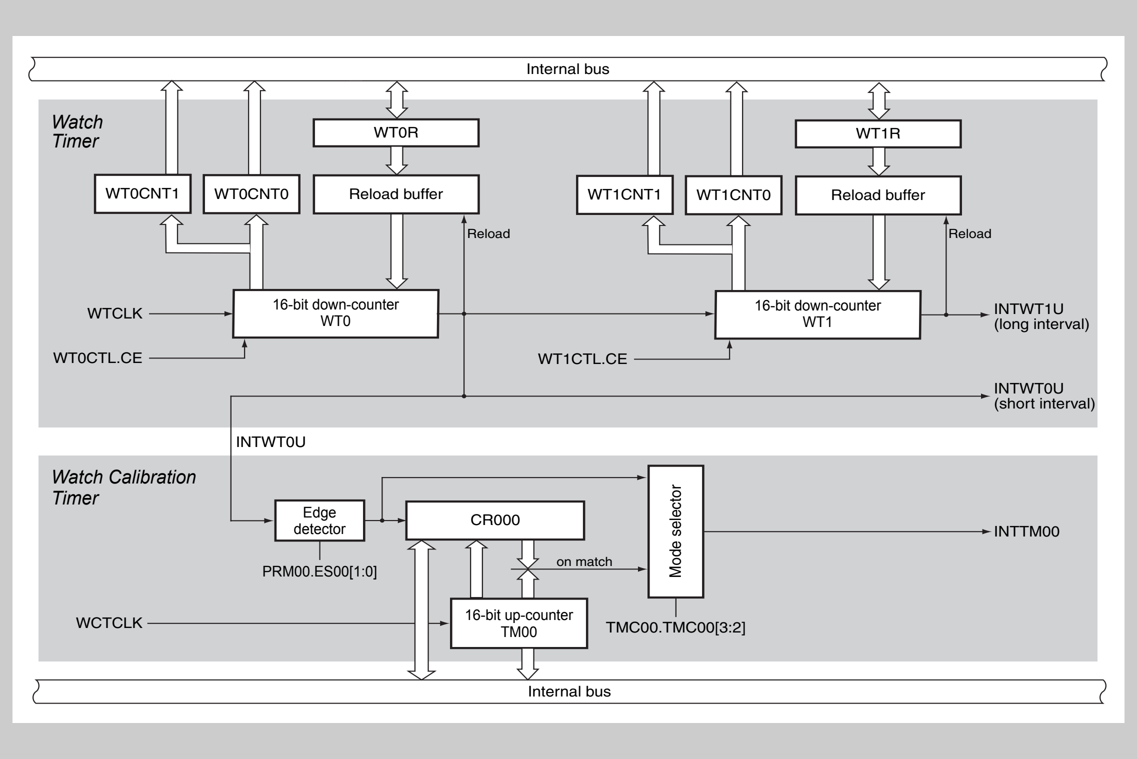


Figure 14-1 Watch Timer configuration

As shown in the figure, WT0 is clocked by WTCLK, a clock generated by the Clock Generator. When WT0 counts down to zero, it generates the INTWT0UV interrupt.

WT1 is clocked by the interrupts INTWT0UV. When WT1 reaches zero, it generates the interrupt INTWT1UV.

Two control registers WTnCTL are used to enable the counters. This is done by setting WTnCTL.WTCE to 1.

As soon as the counters are enabled, it is possible to write a start value to the reload registers WT0R and WT1R.

WCT is a capture/compare timer. In this application, it measures the time between two INTWT0UV interrupts. It is clocked by WCTCLK, another clock generated by the Clock Generator.

14.1.2 Principle of operation

In order to generate an interrupt every one or two seconds, WTCLK is usually set to a frequency around 30 KHz. Then, a load value around 2^{15} will yield a running time of about 1 s.

(1) Operation control of WT0

The source and frequency of WTCLK are specified in the Clock Generator register TCC.

The Clock Generator contains a programmable frequency divider that makes it possible to scale down the selected clock source.

Note WTCLK uses the same clock source and clock divider as the LCD Controller/ Driver clock LCDCLK. The frequency f_{WTCLK} can be the same as f_{LCDCLK} or f_{LCDCLK} / 2. For details refer to *“Clock Generator“ on page 129*.

Typical settings and the resulting maximum time interval between two interrupts are listed in the table below.

Table 14-1 Typical Settings of WTCLK

Clock source	Clock divider setting	WTCLK Frequency	Max. period of INTWT0UV[a]
4 MHz main osc.	1 / 128	31.25 KHz	2.097 s
32 KHz sub osc.	1	32.768 KHz (typ.)	2.0 s
240 KHz ring osc.	1 / 8	30 KHz (typ.)	2.184533 s

[a] The maximum period corresponds to a counter load value of $2^{16} - 1$.

Note that you can double the maximum period by setting TCC.WTSEL1 to 1.

The clock input can be disabled (WT0CTL.WTCE = 0). This stops the Watch Timer. After reset, the timer is also stopped.

When WT0 is enabled and a non-zero reload value is specified, the counter decreases with every rising edge of WTCLK. When the counter reaches zero, the interrupt INTWT0UV is active high for one clock cycle. Upon undeflow, i.e. with the next clock, the timer reloads its start value and resumes down-counting. The load value can be freely chosen

(2) Operation of WT1

Once WT1 is enabled and a non-zero reload value is specified, its counter decreases with every interrupt INTWT0UV.

When WT1 reaches zero, it generates the interrupt INTWT1UV. Upon undeflow, i.e. with the next clock, the timer reloads its load value and restarts down-counting. The load value can be freely chosen.

Starting WT1 requires some attention. For further details refer to *“Watch Timer start-up“ on page 486*.

(3) Operation of WCT

The third counter WCT is used for clock correction. This counter is connected to PCLK1 (8 MHz) or directly to the 4 MHz main oscillator. It is used to measure the time between two INTWT0UV requests.

For this measurement, WCT is configured as a capture timer.

Once it is enabled, the WCT counter is increased with every rising edge of its clock. When the value $FFFF_H$ is reached, the counter sets a flag and restarts with 0000_H.

The interrupt INTWT0UV from counter WT0 triggers the capture operation. At every INTWT0UV, the count value is captured, and the interrupt INTTM00 is generated. From the counter difference between two consecutive capture events, the accuracy of the WTCLK can be measured, and WT0 or WT1, respectively, can be corrected.

The WCT can be programmed to restart counting after the capture operation.

Note The WCT detects the valid edge of INTWT0UV by sampling its input signal (the INTWT0UV interrupt line) with WCTCLK. The capture operation is only performed if the same level after a valid edge is detected two times in series.

As a consequence, the time interval measurement will only work correctly if $f_{WTCLK} < f_{WCTCLK} / 2$.

14.2 Watch Timer Registers

The Watch Timer counters WT0 and WT1 are controlled and operated by means of the following registers:

Table 14-2 WTn registers overview

Register name	Shortcut	Address
Watch timer synchronized read register	WTnCNT0	<base>
Watch timer non-synchronized read register	WTnCNT1	<base> + 2_H
Watch timer reload register	WTnR	<base> + 4_H
Watch timer control register	WTnCTL	<base> + 6_H

Table 14-3 WTn register base addresses

Timer	Base address
WT0	FFFF $F560_H$
WT1	FFFF $F570_H$

(1) WTnCTL - WTn timer control register

The 8-bit WTnCTL register controls the operation of the timer WTn.

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

Address <base> + 6_H

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

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

Table 14-4 WTnCTL register contents

Bit position	Bit name	Function
7	WTCE	Watch timer counter enable: 0: Disable count operation (the timer stops immediately with the count value 0000_H and does not operate). 1: Enable count operation.

Note
1. When WTnCTL.WTCE is 1, the counter starts after the counter's load value has been written to the reload register WTnR. As long as WTnR is zero, no counting is performed, and no interrupts INTWTnUV are generated.
2. The first interval from counter start to the first underflow takes at least four clock cycles more than the following intervals. For details refer to *"Watch Timer start-up" on page 486*.

(2) WTnCNT0 - WTn synchronized counter register

The WTnCNT0 register is the synchronized register that can be used to read the present value of the 16-bit counter.

“Synchronized” means that the read access via the internal bus is synchronized with the counter clock. The synchronization process causes a delay, but the resulting value is reliable.

Access This register is read-only, in 16-bit units.

Address <base> of WTn

Initial Value 0000_H. This register is cleared by any reset and when WTnCTL.WTCE = 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Updated counter value (synchronized)															
R															

Note Due to the low frequencies of the counter clocks, the synchronization can take about up to two WTCLK. For a quick response, it is recommended to read the non-synchronized counter register WTnCNT1.

(3) WTnCNT1 - WTn non-synchronized counter read register

The WTnCNT1 register is the non-synchronized register that can be used to read the present value of the corresponding 16-bit counter.

“Non-synchronized” means that the read access via the internal bus is not synchronized with the counter clock. It returns the instantaneous value immediately, with the risk that this value is just being updated by the counter and therefore in doubt.

Access This register is read-only, in 16-bit units.

Address <base> + 2_H

Initial Value 0000_H. This register is cleared by any reset and when WTnCTL.WTCE = 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Instantaneous counter value (non-synchronized)															
R															

Note The value read from this register can be incorrect, because the read access is not synchronized with the counter clock.

Therefore, this register shall be read multiple times within one period of the counter clock cycle.

If the difference between the first and the second value is not greater than one, you can consider the second value to be correct. If the difference between the two values is greater than one, you have to read the register a third time and compare the third value with the second. Again, the difference must not be greater than one.

If the read accesses do not happen within one period of the counter clock cycle, the difference between the last two values will be greater than one. In this case, you can only repeat the procedure or estimate the updated counter value.

Reading the counter value via WTnCNT1 instead of WTnCNT0 is only reasonable when the CPU clock is remarkably higher than WTCLK and the overhead of multiple reading WTnCNT1 is justifiable.

(4) WTnR - WTn reload register

The WTnR register is a dedicated register for setting the reload value of the corresponding counter.

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

Address <base> + 4_H

Initial Value 0000_H. This register is cleared by any reset and when WTnCTL.WTCE = 0.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">Down-counter load value</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

Note

1. WTnR can only be written when WTnCTL.WTCE = 1 (counter enabled).
2. The load value must be non-zero (0001_H …$FFFF_H$).
3. The contents of this register is automatically copied to the reload buffer. The counters load their values from the respective buffers at underflow. To ensure correct operation, this register shall not be written twice within three cycles of the counter clock. A second write attempt within that time span is ignored.
4. The value read from WTnR is the target start value. It is not necessarily identical with the current start value that is stored in the reload buffer. The buffer may not yet be updated.

14.3 Watch Timer Operation

This section describes the operation of the Watch Timer counters in detail.

14.3.1 Timing of steady operation

The contents of the reload registers WTnR can be changed at any time, provided the corresponding counter is enabled. The contents is then copied to the reload buffer. The counters WTn reload their start value from the buffer upon underflow.

This is illustrated in the following figure.

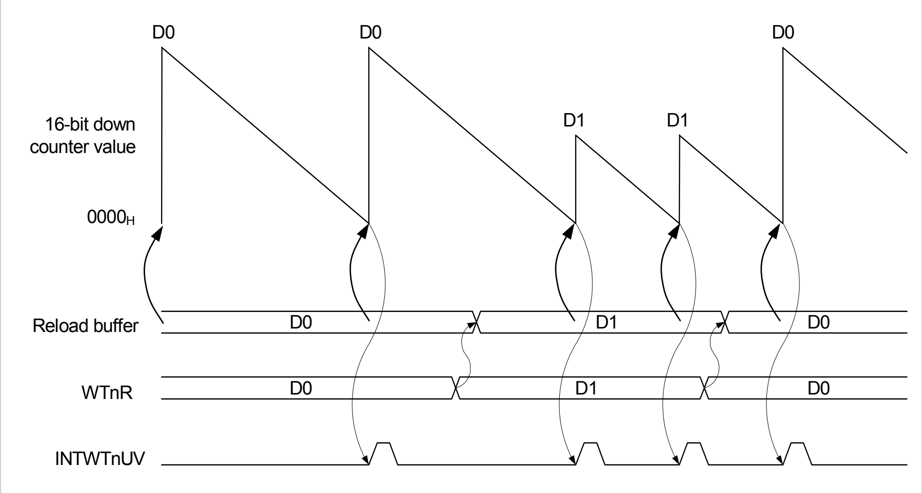


Figure 14-2 Reload timing and interrupt generation

D0 and D1 are two different reload values.

Note also that there is a delay between writing to WTnR and making the data available in the reload buffer, depending on the previous reload value and the chosen count clock.

14.3.2 Watch Timer start-up

The first interval after starting WT0 and WT1 until their first underflow takes at least four additional input clock cycles. At this point in time, the values of the counter registers WTnCNT are not correct.

After the first automatic reload of the WTnR value, the counter registers WTnCNT hold the correct number of clock cycles since the last underflow.

In the following, the start-up procedure of WT1 is described, because of its higher relevance from an application point of view. However, all statements refer also to WT0.

Start-up timing Starting WT1 correctly requires some attention in order to avoid wrong calculation of the watch time.

If WT1 is used as an extended Watch Timer counter, two steps in the following order are required:

- The counter has to be enabled by setting WT1CTL.WTCE = 1.
- The counter's reload register WT1R has to be set to a non-zero value.

Both actions consume a different amount of input clock cycles to become effective, as shown in the following diagram.

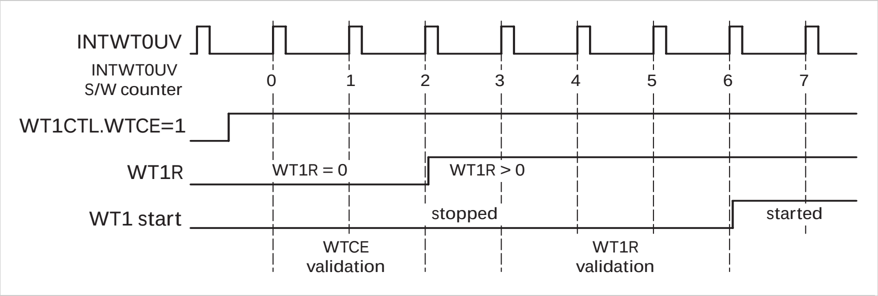


Figure 14-3 WT1 start-up timing

To start the counter in a deterministic way, the above actions have to be synchronized to the WT1 input clock, which is INTWT0UV. For that purpose it is recommended to maintain a software counter that is increased inside the INTWT0UV interrupt service routine. By this means, it is ensured that the actions are performed at the correct point in time.

Setting WT1CTL.WTCE to 1 enables WT1. The write access can happen at any time. Due to internal clock synchronization, it takes at least two complete input clock cycles, that means two INTWT0UV intervals (WTCE validation time 0 –> 1 –> 2) to become effective. After that, WT1 is prepared to acknowledge the reload value.

S/W counter state “2” indicates that the reload value can be written now (WT1R > 0). This time, at least three complete input clock cycles (WTR1 validation time 3 –> 4 –> 5 –> 6) are required to take over the reload value from WT1R to the reload buffer and to start counting. At S/W counter state 6 the counter WT1CNT is preloaded with the WT1R contents.

As a consequence, register WT1CNT does not show the correct number of INTWT0UV events after WT1R > 0, but a value of four less:

- 1 INTWT0UV cycle 2 –> 3 taken for the cycle WT1R is written
- 3 INTWT0UV cycles 3 –> 4 –> 5 –> 6 for WT1R validation time

The above calculation assumes that WT1R is written within one INTWT0UV cycle, which is highly probable, considering INTWT0UV to be the "one second tick".

However, it may happen that the write to WT1R is delayed because of other circumstances (nested interrupts, DMA transfers, etc.) and may happen after S/W counter state 3.

Thus, WT1 would start later, since the 3 clock WTR1 validation time is maintained.

In order to recognize that situation, read the WT1CNT1 register and compare its contents with the value written to WT1R. If both are equal, WTR1 has been written before S/W counter state 3, add four when reading WT1CNT. If they are not equal check again at next INTWT0UV and add the proper number of correction cycles.

14.4 Watch Calibration Timer Registers

The Watch Calibration Timer is controlled by means of the following registers:

Table 14-5 WCT registers overview

Register name	Shortcut	Address
WCT timer / counter read register	TM00	<base>
WCT capture / compare register	CR000	<base> + 2_H
WCT mode control register	TMC00	<base> + 6_H
WCT prescaler mode register	PRM00	<base> + 7_H
WCT capture / compare control register	CRC00	<base> + 8_H

Table 14-6 WCT register base address

Timer	Base address
WCT	FFFF $F5E0_H$

(1) TMC00 - WCT mode control register

The 8-bit TMC00 register controls the operation of the WCT.

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

Address <base> + 6_H

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

7	6	5	4	3	2	1	0
0	0	0	0	TMC003	TMC002	0	OVF00
R	R	R	R	R/W	R/W	R	R/W

Table 14-7 TMC00 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3 to 2</td><td>TMC00[3:2]</td><td>Operation mode selection:
<table>
<tr><th>TMC003</th><th>TMC002</th><th>Operating mode</th></tr>
<tr><td>0</td><td>0</td><td>Stop mode</td></tr>
<tr><td>0</td><td>1</td><td>Free-running mode. Generates interrupt on match between TM00 and CR000.</td></tr>
<tr><td>1</td><td>0</td><td>INTWT0UV interval measurement mode with restart of TM00. Generates interrupt with valid edge of INTWT0UV.</td></tr>
<tr><td>1</td><td>1</td><td>Setting prohibited.</td></tr>
</table>
</td></tr>
<tr><td>0</td><td>OVF00</td><td>Counter overflow indicator:
0: No overflow
1: Overflow occurred</td></tr>
</table>

Note

1. If an attempt is made to change the setting of TMC00[3:2] while the timer is running, these bits are cleared and the timer is stopped. When the timer is stopped, you can change the operation mode.
2. The OVF00 bit is set when the counter reaches $FFFF_H$ and once more when the counter continues with 0000_H. Clearing OVF00 within that time has no effect.

(2) PRM00 - WCT prescaler mode register

The 8-bit PRM00 register is used to select the “valid edge” of INTWT0UV for interval measurements.

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

Address <base> + 7_H

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

7	6	5	4	3	2	1	0
0	0	ES001	ES000	0	0	0	0
R	R	R/W	R/W	R	R	R	R

Table 14-8 PRM00 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>5 to 4</td><td>ES00[1:0]</td><td>Edge selection:
<table>
<tr><th>ES001</th><th>ES000</th><th>Valid edge</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Setting prohibited</td></tr>
<tr><td>1</td><td>1</td><td>Both rising and falling edges</td></tr>
</table></td></tr>
</table>

All other bits are initialized as zero and must not be changed.

Note
1. If both edges of INTWT0UV are specified as valid, INTWT0UV interval measurement is not possible.
2. Stop the timer before changing ES00[1:0].

(3) CRC00 - WCT capture / compare control register

The 8-bit CRC00 register controls the operation of the capture/compare register CR000.

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

Address <base> + 8_H

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

7	6	5	4	3	2	1	0
0	0	0	0	0	X	CRC001	CRC000
R	R	R	R	R	R/W	R/W	R/W

Table 14-9 CRC00 register contents

Bit position	Bit name	Function
1	CRC001	Setting of the CR000 capture trigger signal: 0: No capture signal 1: INTWT0UV. capture signal **Caution:** CRC001 must be set to 1 after reset.
2	CRC000	Selects the operation mode of CR000: 0: Operates as a compare register. 1: Operates as capture register.

Note

1. Stop the timer before changing the contents of this register.
2. If both the rising edge and falling edge are specified as valid for the INTWT0UV signal, the interval measurement does not work.
3. Be sure to set bits 7 to 3 to 0.

(4) CR000 - WCT capture / compare register 1

The 16-bit CR000 register can be used as a capture register or as a compare register. Whether it is used as a capture register or compare register is specified in bit CRC00.CRC000.

- Compare mode:
  In compare mode, the value written to CR000 is continually compared with the count value of TM00. When the two values match, the interrupt request INTTM00 is generated.
- Capture mode:
  In capture mode, the count value of TM00 is copied to CR000 upon a valid edge of INTWT0UV. Then, the interrupt INTTM00 is generated.

  The valid edge of INTWT0UV can be selected as a capture trigger. The valid edge is specified in PRM00.ES00[1:0].
  In capture mode, a read access to register CR000 is not synchronized with the counter operation. Read access and register update can occur simultaneously. If that happens, CR000 holds the correct value, but the value read is undefined.

Access In compare mode, this register can be read/written in 16-bit units. In capture mode, it cannot be written.

Address <base> + 4_H

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

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">Compare or captured value (captured from TM00)</td></tr>
<tr><td colspan="16">R(W)</td></tr>
</table>

Note Stop the timer before changing the contents of this register.

(5) TM00 - WCT timer / counter read register

The TM00 register can be used to read the present value of the 16-bit up-counter.

The counter is increased with every rising edge of the input clock. If the counter value is read while the counter is running, input of the count clock is temporarily stopped, and the counter value at this point is read.

Access This register is read-only, in 16-bit units.

Address <base>

Initial Value 0000_H. This register is cleared by any reset, when the counter is stopped (TMC00.TMC00[3:2] = 0), and when the counter is in INTWT0UV interval measurement mode and a valid edge of INTWT0UV is detected.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">Counter value</td></tr>
<tr><td colspan="16">R</td></tr>
</table>

When the counter overflows, it sets the flag TMC00.OVF00 to 1 and continues with 0000_H.

14.5 Watch Calibration Timer Operation

The Watch Calibration Timer WCT is used to measure the time between two successive occurrences of the Watch Timer WT0 underflow interrupt INTWT0UV.

The WCT is supplied with the stable clock WCTCLK:

- WCTCLK = 4 MHz main oscillator, if PSM.CMODE = 1
- WCTCLK = 8 MHz PCLK1, if PSM.CMODE = 0

For further details refer to *“Clock Generator“ on page 129*.

The measured INTWT0UV interval time gives an indication about the accuracy of the sub or ring oscillator. A correction value can be calculated to calibrate WT0 and WT1 by changing their reload values.

The interval measurement can be performed by both WCT operating modes:

- Free-running mode
- Restart mode

If a timer overflow can occur during the interval measurement, take care for regarding also the overflow flag TMC00.OVF00 for calculating the interval correctly.

14.5.1 INTWT0UV interval measurement with free-running counter

When the timer is used as a free-running counter (see register TMC00) and it detects the valid edge of INTWT0UV, it

- copies the present counter value of register TM00 to CR000,
- generates the interrupt request INTTM00.

The valid edge (rising edge, falling edge) is specified in register PRM00. If both edges are specified, CR000 cannot perform a capture operation.

Setup example

TMC00 = 0000 0100$_B$:	Free running mode
CRC00 = 0000 0x11$_B$:	CR000 as capture register with INTWT0UV as capture signal
PRM00.ES00[1:0] = 01$_B$:	Rising edge

The following figure is not to scale but illustrates the operation.

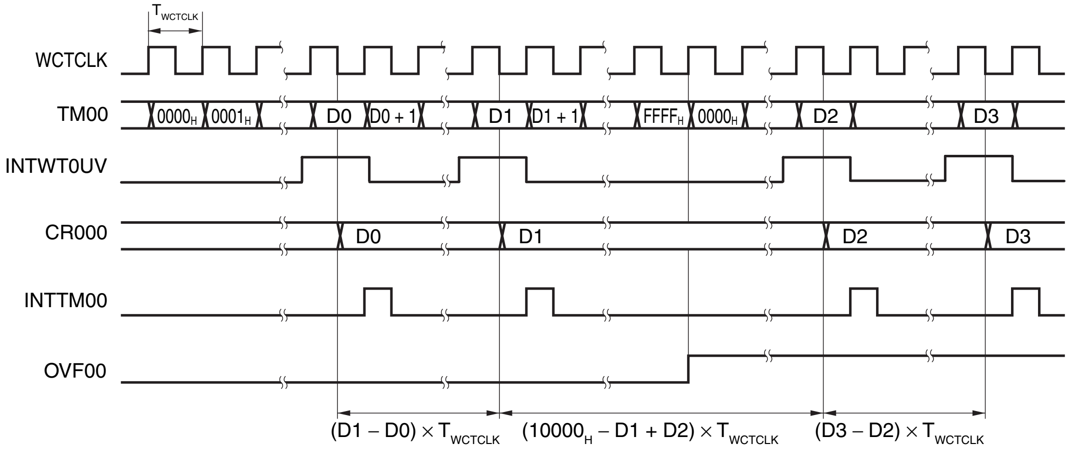


Figure 14-4 Timing in free-running mode

As shown in the figure, the interrupt INTTM00 can be used as a trigger for reading the register CR000.

The interval duration must be calculated from the difference between the present and the previous value of CR000.

Note If TM00 overflows between two occurrences of INTWT0UV, that means between two capture triggers, the overflow flag TMC00.OVF00 is set. Therefore, check also TMC00.OVF00 when reading the second capture value in order to calculate the interval correctly, because an overflow may happen during the measurement.

Consider the chosen periods for INTWT0UV and of WCTCLK.

14.5.2 INTWT0UV interval measurement by restarting the counter

When the timer is in restart mode (see register TMC00) and it detects the valid edge of INTWT0UV, it

- copies the present counter value of register TM00 to CR000,
- clears TM00 (restarts counting),
- generates the interrupt request INTTM00.

The valid edge (rising edge, falling edge) is specified in register PRM00. If both edges are specified, CR000 cannot perform a capture operation.

Setup example

TMC00 = 0000 1000$_B$:	Restart mode
CRC00 = 0000 0x11$_B$:	CR000 as capture register with INTWT0UV as capture signal
PRM00.ES00[1:0] = 0100 0000$_B$:	Rising edge

The following figure is not to scale but illustrates the operation.

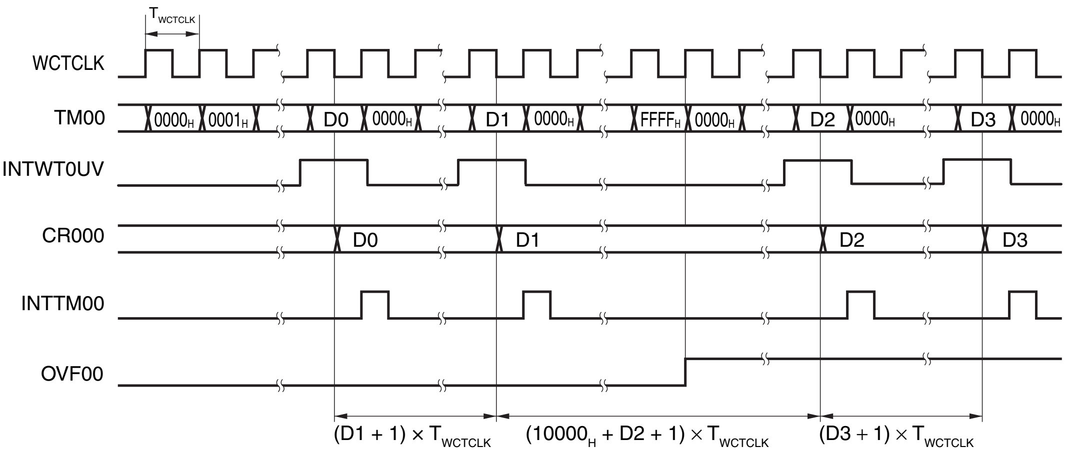


Figure 14-5 Timing in restart mode

As shown in the figure, the present value of CR000 is directly related to the duration of the previous interval.

Note If TM00 overflows between two occurrences of INTWT0UV, that means between two capture triggers, the overflow flag TMC00.OVF00 is set. Therefore, check also TMC00.OVF00 when reading the second capture value in order to calculate the interval correctly, because an overflow may have happened during the measurement.

Chapter 15 Watchdog Timer (WDT)

The Watchdog Timer is used to escape from a system deadlock or program runaway. If it is not restarted within a certain time, the Watchdog Timer flows over and interrupts or even resets the microcontroller.

15.1 Overview

The Watchdog Timer contains an up-counter that is driven by the Watchdog Timer clock WDTCLK. This clock can be derived from the main oscillator, the ring oscillator, or the sub oscillator. It's frequency can be identical with the frequency of the source clock or a fraction thereof.

Features summary

The Watchdog Timer

- can generate the non-maskable interrupt NMIWDT
- can generate a hardware reset by means of the internal signal RESWDT
- has a programmable running time (set in terms of 2^n multiples of WDTCLK periods)
- is specially protected against inadvertent setup changes

15.1.1 Description

The following figure shows a simplified block diagram.

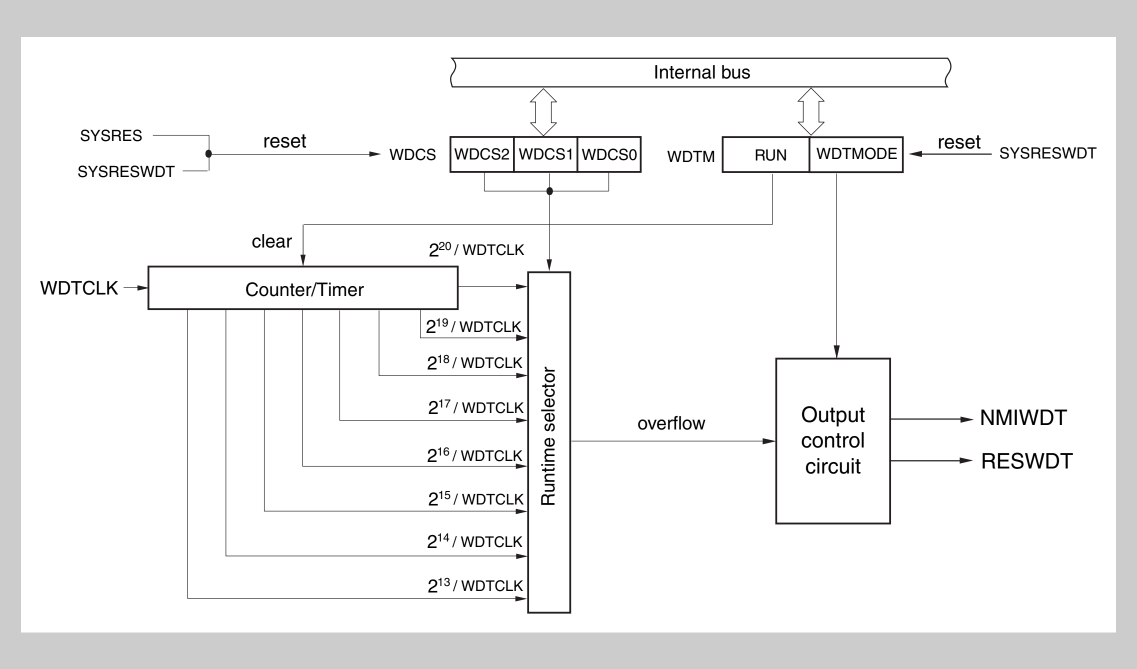


Figure 15-1 Block diagram of the Watchdog Timer

As shown in the figure, the WDCS register controls the running time and the WDTM register the operating mode.

The running time can be chosen between 2^{13} and 2^{20} times the period of the Watchdog Timer clock WDTCLK.

The figure shows also, that the run and mode settings of the WDTM register are only cleared by SYSRESWDT.

15.1.2 Principle of operation

Before the Watchdog Timer is started, its running time and mode have to be configured.

The Watchdog Timer has two operating modes:

- Mode 0 (generate non-maskable interrupt NMIWDT)
- Mode 1 (generate reset request RESWDT)

The mode is defined by the bit WDTM.WDTMODE. The mode can only be changed after SYSRESWDT, that means, after external $\overline{\text{RESET}}$ or Power-On Clear.

(1) Watchdog Timer mode 0 (generate non-maskable interrupt NMIWDT)

If WDTM.WDTMODE is 0, the Watchdog Timer is in interrupt-request mode. This is the default after initialization.

Setting bit WDTM.RUN to 1 starts the counter. Without intervention, the timer will now run until the specified time has elapsed and then generate the non-maskable interrupt NMIWDT. After that, the counter is reset to zero and starts counting again.

(2) Watchdog Timer mode 1 (generate reset request RESWDT)

If WDTM.WDTMODE is 1, the Watchdog Timer is in reset-request mode.

Setting bit WDTM.RUN to 1 starts the counter. Without intervention, the timer will now run until the specified time has elapsed and then generate the internal RESWDT signal. After that, the counter operation is stopped until the system reset SYSRES or SYSRESWDT occurs.

(3) Watchdog Timer running

Once it is running, the Watchdog Timer cannot be stopped by software. It can only be stopped by the reset signal SYSRESWDT. This signal is generated by the Reset module at power-on and external $\overline{\text{RESET}}$.

The way to prevent the timer from flowing over is writing to the register WDTM before the specified time has elapsed. The write access resets the counter to zero.

15.1.3 Watchdog Timer clock

The Watchdog Timer clock WDTCLK is generated by the Clock Generator. It can be derived from the main, ring or sub oscillator.

The generation of WDTCLK is controlled by the WCC register of the Clock Generator.

In this register, it is possible to choose the main, sub, or ring oscillator as the clock source (WCC.SOSCW, WCC.WDTSEL0).

You can also choose a suitable frequency divider between 1 and 128 (WCC.WPS[2:0]).

Please refer to *“Clock Generator“ on page 129* for further details.

Note Once the timer has been started, do not switch off the selected clock source of WDTCLK.

When the microcontroller is in HALT mode, the Watchdog Timer remains active.

The activity in the other power save modes depends on the availability of the WDTCLK clock source.

When the WDTCLK resumes operation, the Watchdog Timer is not reset but continues counting. To prevent a quick and unexpected overflow, it is recommended to write to WDTM and thus clear the Watchdog Timer counter before entering one of these power save modes.

15.1.4 Reset behavior

The reset of the Watchdog Timer is controlled by the two reset inputs SYSRES and SYSRESWDT. The respective signals are generated by the Reset module.

Every reset sets the WDCS register to the longest possible running time.

SYSRESWDT The watchdog reset SYSRESWDT is used to initialize the Watchdog Timer. This signal is generated at power-on and after external $\overline{\text{RESET}}$.

After SYSRESWDT, all registers are set to their reset values, and the timer is stopped. You have to write the required settings to the WDCS register and may start the counter. Once the counter has been started, it cannot be reprogrammed or stopped unless the next reset (SYSRES or SYSRESWDT) occurs.

SYSRES SYSRES is generated by all reset sources.

SYSRES does not reset the register WDTM. That means, the timer status (running or stopped) and mode (generate interrupt or reset request) remain unchanged.

If the Watchdog Timer was running before SYSRES was released, the counter is automatically cleared and restarts with the new timing.

Note

1. Every reset clears also the WCC register. That means, the WDTCLK has the frequency of the 240 KHz ring oscillator. In combination with the largest time factor (2^{20}), this yields a running time of 4.37 s.
2. After any reset, the write protection for WDCS is disabled. WDCS can be written once to specify a shorter time interval. After that, the WDCS register is write-protected.

15.2 Watchdog Timer Registers

The Watchdog Timer is controlled by means of the following registers:

Table 15-1 Watchdog Timer registers overview

Register name	Shortcut	Address
Watchdog Timer clock selection register	WDCS	<base>
Watchdog Timer command protection register	WCMD	<base> + 2_H
Watchdog Timer mode register	WDTM	<base> + 4_H
Watchdog Timer command status register	WPHS	<base> + 6_H

The registers WDCS and WDTM are protected against accidental changes. A special write procedure, employing the WCMD register, ensures that these registers are not easily rewritten in case of a program hang-up.

Their contents can only be changed after a reset.

In addition, the registers are write-protected when the timer is running. Their protection status is indicated in the WDTM register.

Table 15-2 WDT register base address

Timer	Base address
WDT	FFFF $F6C0_H$

Note Only byte access is supported for the registers WDCS, WCMD and WDTM. The registers are allocated at even addresses. Thus, they cannot be written by a consecutive byte write sequence or a consecutive half word or word write sequence.

(1) WDCS - WDT clock selection register

The 8-bit WDCS register is used to specify the running time of the Watchdog Timer.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *“WCMD - WDT command protection register“ on page 505* for details.

Address <base>

Initial Value 07_H. This register is initialized by SYSRESWDT and SYSRES.

7	6	5	4	3	2	1	0
					WDCS2	WDCS1	WDCS0
R	R	R	R	R	R/W	R/W	R/W

Table 15-3 WDCS register contents

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>2 to 0</td><td>WDCS[2:0]</td><td>Specifies the running time of the Watchdog Timer
<table>
<tr><th>WDCS2</th><th>WDCS1</th><th>WDCS0</th><th>Running time calculation</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>$2^{13} / f_{WDTCLK}$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$2^{14} / f_{WDTCLK}$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$2^{15} / f_{WDTCLK}$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$2^{16} / f_{WDTCLK}$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$2^{17} / f_{WDTCLK}$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>$2^{18} / f_{WDTCLK}$</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>$2^{19} / f_{WDTCLK}$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>$2^{20} / f_{WDTCLK}$</td></tr>
</table>
</td></tr>
</table>

Note The WDCS register must be considered in conjunction with the WCC register of the Clock Generator. The source and frequency of WDTCLK are defined in the WCC register.

The running time depends on the frequency of the chosen clock. The following table shows two examples for 4 MHz and 32 KHz.

Table 15-4 Running time examples

WDCS2	WDCS1	WDCS0	Calculation	Time until overflow	
				f_{WDTCLK} = 4 MHz (main oscillator)	f_{WDTCLK} = 32 KHz (sub oscillator)
0	0	0	$2^{13} / f_{WDTCLK}$	2 ms	256 ms
0	0	1	$2^{14} / f_{WDTCLK}$	4.1 ms	512 ms
0	1	0	$2^{15} / f_{WDTCLK}$	8.2 ms	1.02 s
0	1	1	$2^{16} / f_{WDTCLK}$	16.4 ms	2.05 s
1	0	0	$2^{17} / f_{WDTCLK}$	32.8 ms	4.10 s
1	0	1	$2^{18} / f_{WDTCLK}$	65.5 ms	8.20 s
1	1	0	$2^{19} / f_{WDTCLK}$	131 ms	16.38 s
1	1	1	$2^{20} / f_{WDTCLK}$	262 ms	32.77 s

These are just two examples for WDTCLK. The actual clock signal depends on the clock divider settings and the external oscillator resonators.

Note Every reset sets the WDCS register to 07_H, which means the longest time interval.

After SYSRESWDT, the timer is always stopped and initialized. You can write a smaller value to the register.

After SYSRES, the WDTM register is not cleared. If the Watchdog Timer was running before SYSRES occurred, it remains active. To specify a shorter interval:

1. Write one byte to the WCMD register (the value is ignored)
2. Immediately after that, write one byte with the desired value of WDCS[2:0] to the WDCS register

The write operation resets the watchdog counter to zero, and it continues with the new timing.

Note When the timer is active, WDCS can only be written once after reset. Then, the register is locked until the next reset occurs (WDTM.LOCK_CS = 1).

(2) WDTM - WDT mode register

This register sets the operating mode of the Watchdog Timer and enables or disables counting.

When the Watchdog Timer is running and shall not overflow, it is necessary to write to WDTM before the specified running time has elapsed.

Access This register can be read/written in 8-bit units. Once the Watchdog Timer is started (WDTM.RUN = 1), the contents of this register cannot be changed.

Writing to this register is protected by a special sequence of instructions. Please refer to *“WCMD - WDT command protection register“ on page 505* for details.

Address <base> + 4_H

Initial Value 00_H. This register is cleared by SYSRESWDT. This stops the timer and unlocks the registers. The register remains unchanged after SYSRES.

7	6	5	4	3	2	1	0
RUN	LOCK_TM	LOCK_CS	0	WDTMODE	0	0	0
R/W	R	R	R	R/W	R	R	R

Table 15-5 WDTM register contents

Bit Position	Bit Name	Function
7	RUN	Watchdog Timer running: 0: Timer stopped 1: Timer running (with the time interval specified in register WDCS)
6	LOCK_TM	WDTM register protection status: 0: Register unlocked 1: Register locked (write-protected)
5	LOCK_CS	WDCS register protection status: 0: Register unlocked 1: Register locked (write-protected)
3	WDTMODE	Watchdog Timer operation mode: 0: Mode 0: Generates the non-maskable interrupt NMIWDT upon overflow 1: Mode 1: Generates RESWDT upon overflow

Note After SYSRESWDT, the timer is always stopped and initialized. You can change the register contents by writing.

When the timer is running, you can also write to this register, but the write operation does not change the register contents (WDTM.LOCK_TM = 1). When the timer is running, the write access resets the counter.

To write to the WDTM register:

1. Write one byte to the WCMD register (the value is ignored).
2. Immediately after that, write one byte to the WDTM register (the value is ignored).

With this procedure, restarting the counter is always possible, regardless of the register’s write protection status.

(3) WCMD - WDT command protection register

The 8-bit WCMD register is write-only. It is used to protect the WDTM and WDCS registers from unintended writing.

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

Address <base> + 2_H

Initial Value Undefined

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

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

After writing to the WCMD register, you are permitted to write once to one of the protected registers. This must be done immediately after writing to the WCMD register. If the second write action does not follow immediately, the protected registers are write-locked again. See also *“Write Protected Registers“ on page 124*.

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

The following registers are protected:

- WDCS: Watchdog clock selection register
- WDTM: Watchdog mode control register

An invalid write attempt to one of the above registers sets the error flag WPHS.WPRERR. WPHS.WPRERR is also set, if a write access to WCMD is not followed by an access to one of the protected registers.

Data read from the WCMD register is undefined.

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

---

(4) WPHS - WDT command status register

The WPHS register monitors the success of a write instruction to the WDTM and WDCS registers.

If the write operation to WDTM or WDCS failed because WCMD was not written immediately before writing to WDTM or WDCS, the WPRERR flag is set.

Access This register can be read/written in 8-bit or 1-bit units. After a write access, the register is cleared.

Address <base> + 6_H

Initial Value 00_H. This register is cleared by SYSRESWDT and any write access.

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

Table 15-6 WPHS register contents

Bit Position	Bit Name	Function
0	WPRERR	Error flag: 0: No WDTM or WDCS register writing error has occurred 1: A WDTM or WDCS register writing error has occurred

Chapter 16 Asynchronous Serial Interface (UARTA)

The V850E/Dx3 microcontrollers have following instances of the universal Asynchronous Serial Interface UARTA:

UARTA	All devices
Instances	2
Names	UARTA0 to UARTA1

Throughout this chapter, the individual instances of UARTA are identified by “n”, for example, UARTAn, or UAnCTL0 for the UARTAn control register 0.

16.1 Features

- Transfer rate: 300 bps to 1000 kbps (using dedicated baud rate generator)
- Full-duplex communication:
  - Internal UARTA receive data register n (UAnRX)
  - Internal UARTA transmit data register n (UAnTX)
- 2-pin configuration:
  - TXDAn: Transmit data output pin
  - RXDAn: Receive data input pin
- Reception error output function
  - Parity error
  - Framing error
  - Overrun error
- Interrupt sources: 3
  - Reception complete interrupt (INTUAnR):

    This interrupt occurs upon transfer of receive data from the shift register to receive buffer register n after serial transfer completion, in the reception enabled status.
  - Transmission enable interrupt (INTUAnT):

    This interrupt occurs upon transfer of transmit data from the transmit buffer register to the shift register in the transmission enabled status.
  - Receive error interrupt (INTUAnRE):

    This interrupt occurs upon transfer of erroneous receive data.
- Character length: 7, 8 bits
- Parity function: Odd, even, 0, none
- Transmission stop bit: 1, 2 bits
- On-chip dedicated baud rate generator
- MSB-/LSB-first transfer selectable
- Transmit/receive data inverted input/output possible

- 13 to 20 bits selectable for the SBF (Sync Break Field) in the LIN (Local Interconnect Network) communication format
  - Recognition of 11 bits or more possible for SBF reception in LIN communication format
  - SBF reception flag provided
- DMA support
  Two different DMA trigger events in transmission mode (refer to *"DMA Controller (DMAC)" on page 309*)

16.2 Configuration

The block diagram of the UARTAn is shown below.

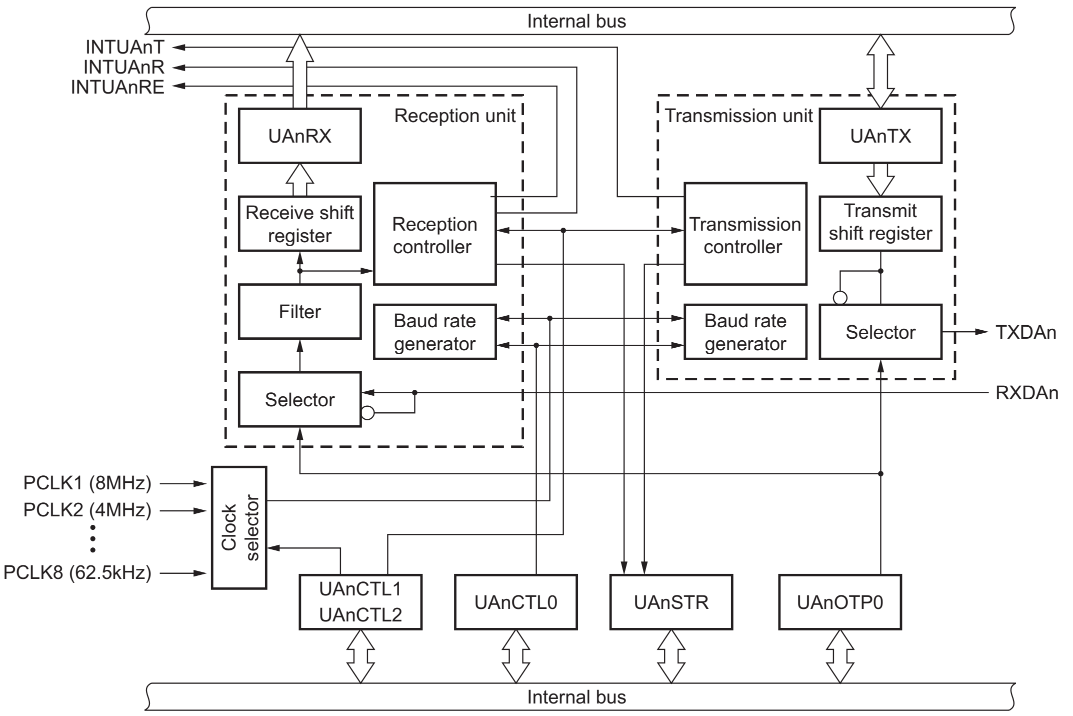


Figure 16-1 Block diagram of Asynchronous Serial Interface UARTAn

Note For the configuration of the baud rate generator, see *Figure 16-11 on page 533*.

UARTAn consists of the following hardware units.

(1) UARTAn control register 0 (UAnCTL0)

The UAnCTL0 register is an 8-bit register used to specify the UARTAn operation.

(2) UARTAn control register 1 (UAnCTL1)

The UAnCTL1 register is an 8-bit register used to select the input clock for the UARTAn.

(3) UARTAn control register 2 (UAnCTL2)

The UAnCTL2 register is an 8-bit register used to control the baud rate for the UARTAn.

(4) UARTAn option control register 0 (UAnOPT0)

The UAnOPT0 register is an 8-bit register used to control serial transfer for the UARTAn.

(5) UARTAn status register (UAnSTR)

The UAnSTRn register consists of flags indicating the error contents when a reception error occurs. Each one of the reception error flags is set (to 1) upon occurrence of a reception error and is reset (to 0) by reading the UAnSTR register.

(6) UARTAn receive shift register

This is a shift register used to convert the serial data input to the RXDAn pin into parallel data. Upon reception of 1 byte of data and detection of the stop bit, the receive data is transferred to the UAnRX register.

This register cannot be manipulated directly.

(7) UARTAn receive data register (UAnRX)

The UAnRX register is an 8-bit register that holds receive data. When 7 characters are received, 0 is stored in the highest bit (when data is received LSB first).

In the reception enabled status, receive data is transferred from the UARTAn receive shift register to the UAnRX register in synchronization with the completion of shift-in processing of 1 frame.

Transfer to the UAnRX register also causes the reception complete interrupt request signal (INTUAnR) to be output.

(8) UARTAn transmit shift register

The transmit shift register is a shift register used to convert the parallel data transferred from the UAnTX register into serial data.

When 1 byte of data is transferred from the UAnTX register, the shift register data is output from the TXDAn pin.

This register cannot be manipulated directly.

(9) UARTAn transmit data register (UAnTX)

The UAnTX register is an 8-bit transmit data buffer. Transmission starts when transmit data is written to the UAnTX register. When data can be written to the UAnTX register (when data of one frame is transferred from the UAnTX register to the UARTAn transmit shift register), the transmission enable interrupt request signal (INTUAnT) is generated.

16.3 UARTA Registers

The asynchroneous serial interfaces UARTAn are controlled and operated by means of the following registers:

Table 16-1 UARTAn registers overview

Register name	Shortcut	Address
UARTAn control register 0	UAnCTL0	<base>
UARTAn control register 1	UAnCTL1	<base> + 1_H
UARTAn control register 2	UAnCTL2	<base> + 2_H
UARTAn option control register 0	UAnOPT0	<base> + 3_H
UARTAn status register	UAnSTR	<base> + 4_H
UARTAn receive data register	UAnRX	<base> + 6_H
UARTAn transmit data register	UAnTX	<base> + 7_H

Table 16-2 UARTAn register base address

Timer	Base address
UARTA0	FFFF $FA00_H$
UARTA1	FFFF $FA10_H$

(1) UAnCTL0 - UARTAn control register 0

The UAnCTL0 register is an 8-bit register that controls the UARTAn serial transfer operation.

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

Address <base>

Initial Value 10_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
UAnPWR	UAnTXE	UAnRXE	UAnDIR	UAnPS1	UAnPS0	UAnCL	UAnSL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 16-3 UAnCTL0 register contents (1/2)

Bit position	Bit name	Function
7	UAnPWR	UARTAn operation disable/enable: 0: Disable UARTAn operation and reset the UARTAn asynchroneously 1: Enable UARTAn operation The UARTAn operation is controlled by the UAnPWR bit. The TXDAn pin output is fixed to high level by clearing the UAnPWR bit to 0 (fixed to low level if UAnOPT0.UAnTDL bit = 1).
6	UAnTXE	Transmit operation disable/enable: 0: Disable transmit operation 1: Enable transmit operation • To start transmission, set the UAnPWR bit to 1 and then set the UAnTXE bit to 1. • To stop transmission, clear the UAnTXE bit to 0 and then the UAnPWR bit to 0. • To initialize the transmission unit, clear UAnTXE bit to 0, wait for two cycles of the base clock, and then set UAnTXE bit to 1 again. Otherwise, initialization may not be executed.
5	UAnRXE	Receive operation disable/enable: 0: Disable receive operation 1: Enable receive operation • To start reception, set the UAnPWR bit to 1 and then set the UAnRXE bit to 1. • To stop reception, clear the UAnRXE bit to 0 and then the UAnPWR bit to 0. • To initialize the reception unit, clear UAnRXE bit to 0, wait for two cycles of the base clock, and then set UAnRXE bit to 1 again. Otherwise, initialization may not be executed.
4	UAnDIR	Transfer direction mode specification (MSB/LSB): 0: MSB first transfer 1: LSB first transfer

Table 16-3 UAnCTL0 register contents (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3, 2</td><td>UAnPS[1:0]</td><td>Parity selection
<table>
<tr><th rowspan="2">UAnPS1</th><th rowspan="2">UAnPS0</th><th colspan="2">Parity selection during</th></tr>
<tr><th>transmission</th><th>reception</th></tr>
<tr><td>0</td><td>0</td><td>No parity output</td><td>Reception with no parity</td></tr>
<tr><td>0</td><td>1</td><td>0 parity output</td><td>Reception with 0 parity</td></tr>
<tr><td>1</td><td>0</td><td>Odd parity output</td><td>Odd parity check</td></tr>
<tr><td>1</td><td>1</td><td>Even parity output</td><td>Even parity check</td></tr>
</table>

• This register is rewritten only when the UAnPWR bit = 0 or the UAnTXE bit = the UAnRXE bit = 0.
• If “Reception with 0 parity” is selected during reception, a parity check is not performed.
Therefore, since the UAnSTR.UAnPE bit is not set, no error interrupt is output.
• When transmission and reception are performed in the LIN format, clear the UAnPS1 and UAnPS0 bits to 00.</td></tr>
<tr><td>1</td><td>UAnCL</td><td>Specification of data character length of 1 frame of transmit/receive data
0: 7 bits
1: 8 bits
This register can be rewritten only when the UAnPWR bit = 0 or the UAnTXE bit = the UAnRXE bit = 0.</td></tr>
<tr><td>0</td><td>UAnSL</td><td>Specification of length of stop bit for transmit data
0: 1 bit
1: 2 bits
This register can be rewritten only when the UAnPWR bit = 0 or the UAnTXE bit = the UAnRXE bit = 0.</td></tr>
</table>

Note For details of parity, see *“Parity types and operations“ on page 530*.

(2) UAnCTL1 - UARTAn control register 1

The UAnCTL1 register is an 8-bit register used to select the input clock for the UARTAn.

For details, see *“UAnCTL1 - UARTAn control register 1“ on page 534*.

(3) UAnCTL2 - UARTAn control register 2

The UAnCTL2 register is an 8-bit register used to control the baud rate for the UARTAn.

For details, see *“UAnCTL2 - UARTAn control register 2“ on page 535*.

(4) UAnOPT0 - UARTAn option control register 0

The UAnOPT0 register is an 8-bit register that controls the serial transfer operation of the UARTAn register.

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

Address <base> + 3_H

Initial Value 14_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
UAnSRF	UAnSRT	UAnSTT	UAnSBL2	UAnSBL1	UAnSBL0	UAnTDL	UAnRDL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 16-4 UAnOPT0 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>UAnSRF</td><td>SBF reception flag:
0: When the UAnCTL0.UAnPWR bit = UAnCTL0.UAnRXE bit = 0 are set. Also upon normal end of SBF reception.
1: During SBF reception
• SBF (Sync Brake Field) reception is judged during LIN communication.
• The UAnSRF bit is held at 1 when an SBF reception error occurs, and then SBF reception is started again.</td></tr>
<tr><td>6</td><td>UAnSRT</td><td>SBF reception trigger:
0: –
1: SBF reception trigger
• This is the SBF reception trigger bit during LIN communication, and when read, “0” is always read. For SBF reception, set the UAnSRT bit (to 1) to enable SBF reception.
• Set the UAnSRT bit after setting the UAnPWR bit = UAnRXE bit = 1.</td></tr>
<tr><td>5</td><td>UAnSTT</td><td>SBF transmission trigger:
0: –
1: SBF transmission trigger [a]
• This is the SBF transmission trigger bit during LIN communication, and when read, “0” is always read.
• Set the UAnSTT bit after setting the UAnPWR bit = UAnTXE bit = 1.</td></tr>
<tr><td>4 to 2</td><td>UAnSBL[2:0]</td><td>SBF transmission length selection:
<table>
<tr><th>UAnSBL2</th><th>UAnSBL1</th><th>UAnSBL0</th><th>SBF transmission length</th></tr>
<tr><td>1</td><td>0</td><td>1</td><td>13-bit output (default value)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>14-bit output</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>15-bit output</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>16-bit output</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>17-bit output</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>18-bit output</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>19-bit output</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>20-bit output</td></tr>
</table>
This register can be set when the UAnPWR bit = 0 or when the UAnTXE bit = 0.</td></tr>
<tr><td>1</td><td>UAnTDL</td><td>Transmit data level bit
0: Normal output of transfer data
1: Inverted output of transfer data
• The output level of the TXDAn pin can be inverted using the UAnTDL bit.
• This register can be set when the UAnPWR bit = 0 or when the UAnTXE bit = 0.</td></tr>
<tr><td>0</td><td>UAnRDL</td><td>Receive data level bit
0: Normal input of transfer data
1: Inverted input of transfer data
• The output level of the RXDAn pin can be inverted using the UAnRDL bit.
• This register can be set when the UAnPWR bit = 0 or the UAnRXE bit = 0.</td></tr>
</table>

a) Before starting the SBF transmission by UAnOPT0.UAnSTT = 1 make sure that no data transfer is ongoing, that means check that UAnSTR.UAnTSF = 0.

(5) UAnSTR - UARTAn status register

The UAnSTR register is an 8-bit register that displays the UARTAn transfer status and reception error contents.

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

Address <base> + 4_H

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

The initialization conditions are shown below.

Register/Bit	Initialization conditions
UAnSTR register	• Reset • UAnCTL0.UAnPWR = 0
UAnTSF bit	• UAnCTL0.UAnTXE = 0
UAnPE, UAnFE, UAnOVE bits	• 0 write • UAnCTL0.UAnRXE = 0

7	6	5	4	3	2	1	0
UAnTSF	0	0	0	0	UAnPE	UAnFE	UAnOVE
R	R/W	R/W	R/W	R/W	R/W[a]	R/W[a]	R/W[a]

[a] These bits can only be cleared by writing, They cannot be set by writing 1 (even if 1 is written, the value is retained).

Table 16-5 UAnSTR register contents (1/2)

Bit position	Bit name	Function
7	UAnTSF	Transfer status flag: 0: – When the UAnPWR bit = 0 or the UAnTXE bit = 0 has been set. – When, following transfer completion, there was no next data transfer from UAnTX register 1: Write to UAnTXB bit The UAnTSF bit is always 1 when performing continuous transmission. When initializing the transmission unit, check that the UAnTSF bit = 0 before performing initialization. The transmit data is not guaranteed when initialization is performed while the UAnTSF bit = 1.

Table 16-5 UAnSTR register contents (2/2)

Bit position	Bit name	Function
2	UAnPE	Parity error flag: 0: – When the UAnPWR bit = 0 or the UAnRXE bit = 0 has been set. – When 0 has been written 1: When parity of data and parity bit do not match during reception. • The operation of the UAnPE bit is controlled by the settings of the UAnCTL0.UAnPS1 and UAnCTL0.UAnPS0 bits. • The UAnPE bit can be read and written, but it can only be cleared by writing 0 to it, and it cannot be set by writing 1 to it. When 1 is written to this bit, the value is retained.
1	UAnFE	Framing error flag 0: – When the UAnPWR bit = 0 or the UAnRXE bit = 0 has been set – When 0 has been written 1: When no stop bit is detected during reception • Only the first bit of the receive data stop bits is checked, regardless of the value of the UAnCTL0.UAnSL bit. • The UAnFE bit can be both read and written, but it can only be cleared by writing 0 to it, and it cannot be set by writing 1 to it. When 1 is written to this bit, the value is retained.
0	UAnOVE	Overrun error flag 0: – When the UAnPWR bit = 0 or the UAnRXE bit = 0 has been set. – When 0 has been written 1: When receive data has been set to the UAnRXB register and the next receive operation is completed before that receive data has been read • When an overrun error occurs, the data is discarded without the next receive data being written to the receive buffer. • The UAnOVE bit can be both read and written, but it can only be cleared by writing 0 to it. When 1 is written to this bit, the value is retained.

(6) UAnRX - UARTAn receive data register

The UAnRX register is an 8-bit buffer register that stores parallel data converted by the receive shift register.

The data stored in the receive shift register is transferred to the UAnRX register upon completion of reception of 1 byte of data.

During LSB-first reception when the data length has been specified as 7 bits, the receive data is transferred to bits 6 to 0 of the UAnRX register and the MSB always becomes 0. During MSB-first reception, the receive data is transferred to bits 7 to 1 of the UAnRX register and the LSB always becomes 0.

When an overrun error (UAnOVE) occurs, the receive data at this time is not transferred to the UAnRX register and is discarded.

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

Address <base> + 6_H

Initial Value FF_H. This register is cleared by any reset.
In addition to reset input, the UAnRX register can be set to FF_H by clearing the UAnCTL0.UAnPWR bit to 0.

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">Receive data</td></tr>
<tr><td colspan="8">R</td></tr>
</table>

(7) UAnTX - UARTAn transmit data register

The UAnTX register is an 8-bit register used to set transmit data.

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

Address <base> + 7_H

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

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">Transmit data</td></tr>
<tr><td colspan="8">R/W</td></tr>
</table>

16.4 Interrupt Request Signals

The following three interrupt request signals are generated from UARTAn:

- Reception complete interrupt request signal (INTUAnR)
- Receive error interrupt request signal (INTUAnRE)
- Transmission enable interrupt request signal (INTUAnT)

(1) Reception complete interrupt request signal (INTUAnR)

A reception complete interrupt request signal is output when data is shifted into the receive shift register and transferred to the UAnRX register in the reception enabled status.

In case of erroneous reception, the reception error interrupt INTUanRE is generated instead of INTUAnR.

No reception complete interrupt request signal is generated in the reception disabled status.

(2) Receive error interrupt request signal (INTUAnRE)

A receive error interrupt request is generated if an error condition occurred during reception, as reflected by UAnSTR.UAnPE (parity error flag), UAnSTR.UAnFE (framing error flag), UAnSTR.UAnOVE (overrun error flag).

Note that INTUAnR and INTUAnRE do exclude each other: upon correct reception of data only INTUAnR is generated. In case of a reception error INTUAnRE is generated only.

(3) Transmission enable interrupt request signal (INTUAnT)

If transmit data is transferred from the UAnTX register to the UARTAn transmit shift register with transmission enabled, the transmission enable interrupt request signal is generated.

16.5 Operation

16.5.1 Data format

Full-duplex serial data reception and transmission is performed.

As shown in the figures below, one data frame of transmit/receive data consists of a start bit, character bits, parity bit, and stop bit(s).

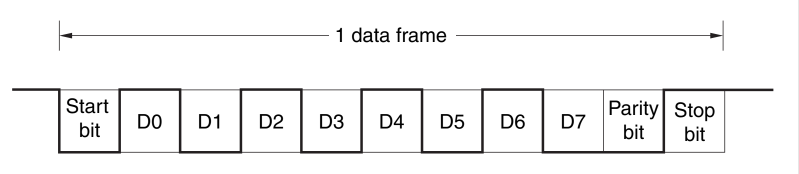
Specification of the character bit length within 1 data frame, parity selection, specification of the stop bit length, and specification of MSB/LSB-first transfer are performed using the UAnCTL0 register.

Moreover, control of UART output/inverted output for the TXDAn bit is performed using the UAnOPT0.UAnTDL bit.

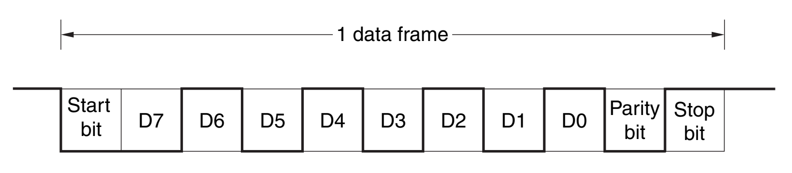
- Start bit..........................1 bit
- Character bits................7 bits/8 bits
- Parity bit ....Even parity/odd parity/0 parity/no parity
- Stop bit..........................1 bit/2 bits

(1) UARTA transmit/receive data format

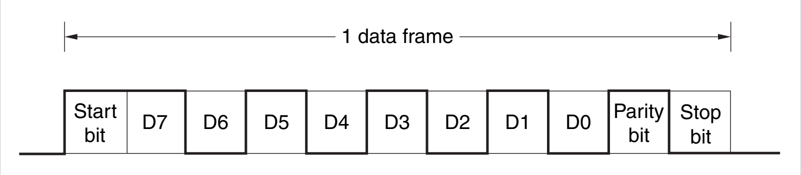
(a) 8-bit data length, LSB first, even parity, 1 stop bit, transfer data: 55H



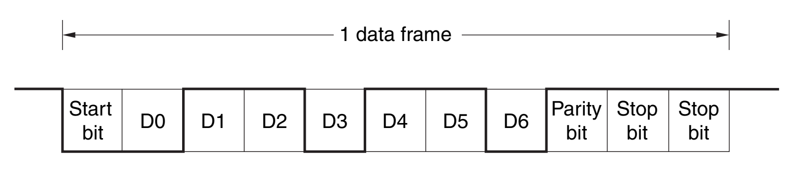
(b) 8-bit data length, MSB first, even parity, 1 stop bit, transfer data: 55H



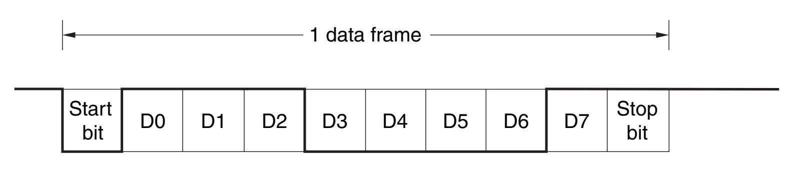
(c) 8-bit data length, MSB first, even parity, 1 stop bit, transfer data: 55H, TXDAn inversion



(d) 7-bit data length, LSB first, odd parity, 2 stop bits, transfer data: 36H



(e) 8-bit data length, LSB first, no parity, 1 stop bit, transfer data: 87H



16.5.2 SBF transmission/reception format

The UARTA has an SBF (Sync Break Field) transmission/reception control function to enable use of the LIN function.

About LIN LIN stands for Local Interconnect Network and is a low-speed (1 to 20 kbps) serial co

mmunication protocol intended to aid the cost reduction of an automotive network.

LIN communication is single-master communication, and up to 15 slaves can be connected to one master.

The LIN slaves are used to control the switches, actuators, and sensors, and these are connected to the LIN master via the LIN network.

Normally, the LIN master is connected to a network such as CAN (Controller Area Network).

In addition, the LIN bus uses a single-wire method and is connected to the nodes via a transceiver that complies with ISO9141.

In the LIN protocol, the master transmits a frame with baud rate information and the slave receives it and corrects the baud rate error. Therefore, communication is possible when the baud rate error in the slave is ±15% or less.

Figure 16-2 and *Figure 16-3* outline the transmission and reception manipulations of LIN.

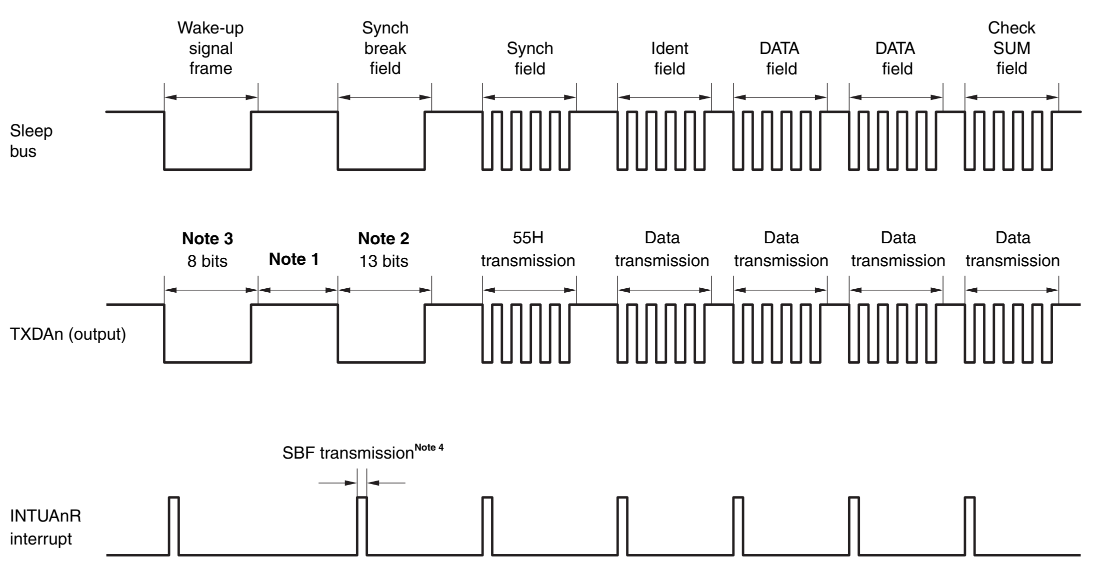


Figure 16-2 LIN transmission manipulation outline

Note

1. The interval between each field is controlled by software.
2. SBF output is performed by hardware. The output width is the bit length set by the UAnOPT0.UAnSBL2 to UAnOPT0.UAnSBL0 bits. If even finer output width adjustments are required, such adjustments can be performed using the UAnCTLn.UAnBRS7 to UAnCTLn.UAnBRS0 bits.
3. 80H transfer in the 8-bit mode is substituted for the wakeup signal frame.

4. A transmission enable interrupt request signal (INTUAnT) is output at the start of each transmission. The INTUAnT signal is also output at the start of each SBF transmission.

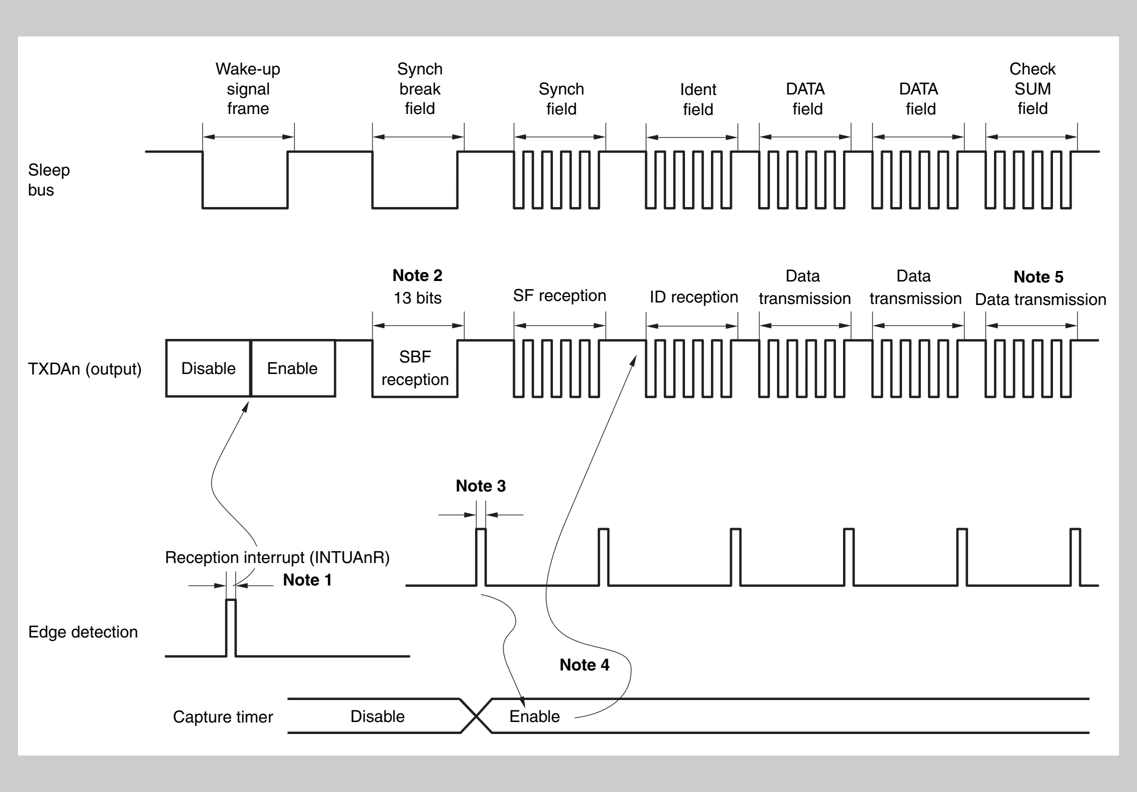


Figure 16-3 LIN reception manipulation outline

Note
1. The wakeup signal is sent by the pin edge detector, UARTAn is enabled, and the SBF reception mode is set.
2. The receive operation is performed until detection of the stop bit. Upon detection of SBF reception of 11 or more bits, normal SBF reception end is judged, and an interrupt signal is output. Upon detection of SBF reception of less than 11 bits, an SBF reception error is judged, no interrupt signal is output, and the mode returns to the SBF reception mode.
3. If SBF reception ends normally, an interrupt request signal is output. The timer is enabled by an SBF reception complete interrupt. Moreover, error detection for the UAnSTR.UAnOVE, UAnSTR.UAnPE, and UAnSTR.UAnFE bits is suppressed and UART communication error detection processing and UARTAn receive shift register and data transfer of the UAnRX register are not performed. The UARTAn receive shift register holds the initial value, FFH.
4. The RXDAn pin is connected to TI (capture input) of the timer, the transfer rate is calculated, and the baud rate error is calculated. The value of the UAnCTL2 register obtained by correcting the baud rate error after dropping UARTA enable is set again, causing the status to become the reception status.
5. Check-sum field distinctions are made by software. UARTAn is initialized following CSF reception, and the processing for setting the SBF reception mode again is performed by software.

16.5.3 SBF transmission

When the UAnCTL0.UAnPWR bit = UAnCTL0.UAnTXE bit = 1, the transmission enabled status is entered, and SBF transmission is started by setting (to 1) the SBF transmission trigger (UAnOPT0.UAnSTT bit).

Thereafter, a low level the width of bits 13 to 20 specified by the UAnOPT0.UAnSBL2 to UAnOPT0.UAnSBL0 bits is output. A transmission enable interrupt request signal (INTUAnT) is generated upon SBF transmission start. Following the end of SBF transmission, the UAnSTT bit is automatically cleared. Thereafter, the UART transmission mode is restored.

Transmission is suspended until the data to be transmitted next is written to the UAnTX register, or until the SBF transmission trigger (UAnSTT bit) is set.

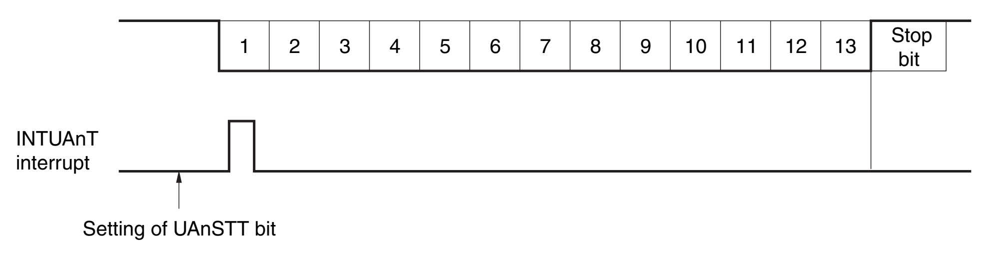


Figure 16-4 SBF transmission

16.5.4 SBF reception

The reception enabled status is achieved by setting the UAnCTL0.UAnPWR bit to 1 and then setting the UAnCTL0.UAnRX bit to 1.

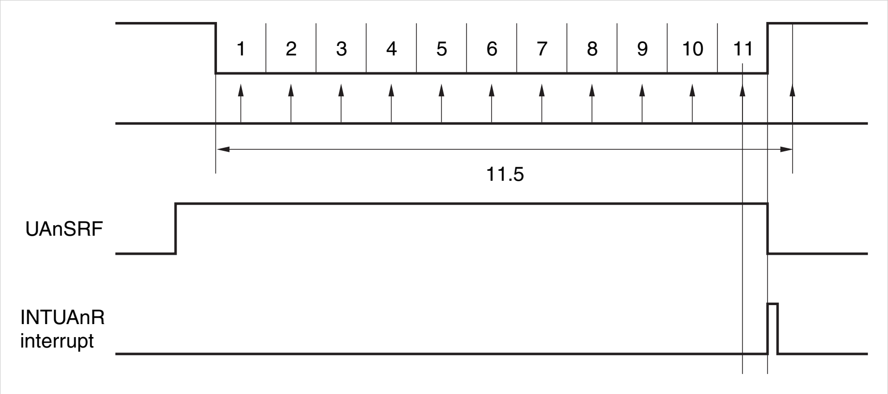
The SBF reception wait status is set by setting the SBF reception trigger (UAnOPT0.UAnSTR bit) to 1.

In the SBF reception wait status, similarly to the UART reception wait status, the RXDAn pin is monitored and start bit detection is performed.

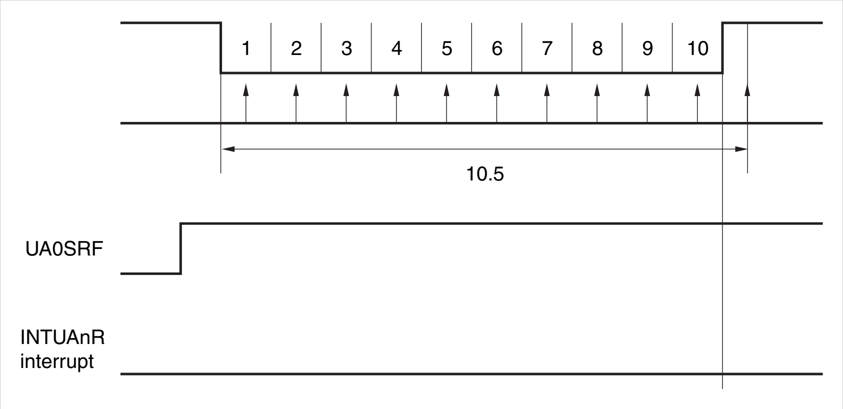
Following detection of the start bit, reception is started and the internal counter counts up according to the set baud rate.

When a stop bit is received, if the SBF width is 11 or more bits, normal processing is judged and a reception complete interrupt request signal (INTUAnR) is output. The UAnOPT0.UAnSRF bit is automatically cleared and SBF reception ends. Error detection for the UAnSTR.UAnOVE, UAnSTR.UAnPE, and UAnSTR.UAnFE bits is suppressed and UART communication error detection processing is not performed. Moreover, data transfer of the UARTAn reception shift register and UAnRX register is not performed and FFH, the initial value, is held. If the SBF width is 10 or fewer bits, reception is terminated as error processing without outputting an interrupt, and the SBF reception mode is returned to. The UAnSRF bit is not cleared at this time.

(a) Normal SBF reception (detection of stop bit in more than 10.5 bits)



(b) SBF reception error (detection of stop bit in 10.5 or fewer bits)



16.5.5 UART transmission

A high level is output to the TXDAn pin by setting the UAnCTL0.UAnPWR bit to 1.

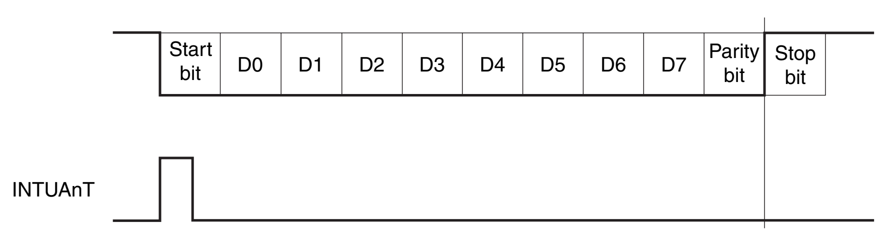
Next, the transmission enabled status is set by setting the UAnCTL0.UAnTXE bit to 1, and transmission is started by writing transmit data to the UAnTX register. The start bit, parity bit, and stop bit are automatically added.

Since the CTS (transmit enable signal) input pin is not provided in UARTAn, use a port to check that reception is enabled at the transmit destination.

The data in the UAnTX register is transferred to the UARTAn transmit shift register upon the start of the transmit operation.

A transmission enable interrupt request signal (INTUAnT) is generated upon completion of transmission of the data of the UAnTX register to the UARTAn transmit shift register, and thereafter the contents of the UARTAn transmit shift register are output to the TXDAn pin.

Write of the next transmit data to the UAnTX register is enabled by generating the INTUAnT signal.



Note LSB first

16.5.6 Continuous transmission procedure

UARTAn can write the next transmit data to the UAnTX register when the UARTAn transmit shift register starts the shift operation. The transmit timing of the UARTAn transmit shift register can be judged from the transmission enable interrupt request signal (INTUAnT).

An efficient communication rate is realized by writing the data to be transmitted next to the UAnTX register during transfer.

Caution During continuous transmission execution, perform initialization after checking that the UAnSTR.UAnTSF bit is 0. The transmit data cannot be guaranteed when initialization is performed while the UAnTSF bit is 1.

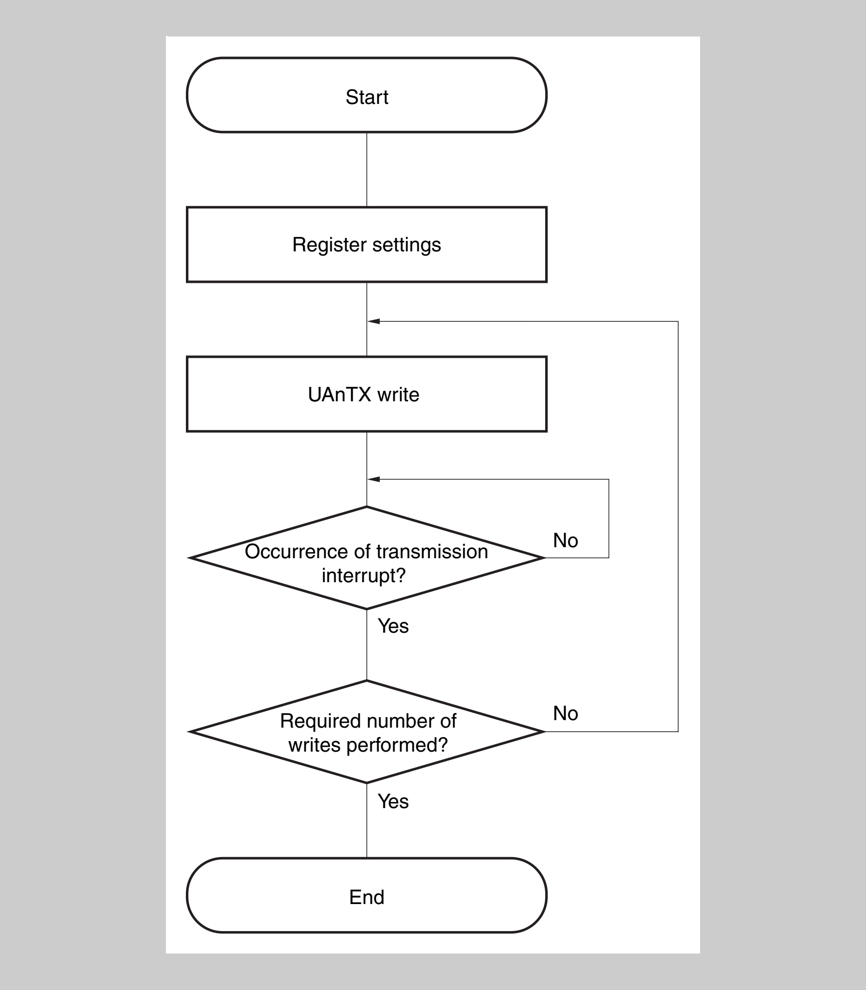


Figure 16-5 Continuous transmission processing flow

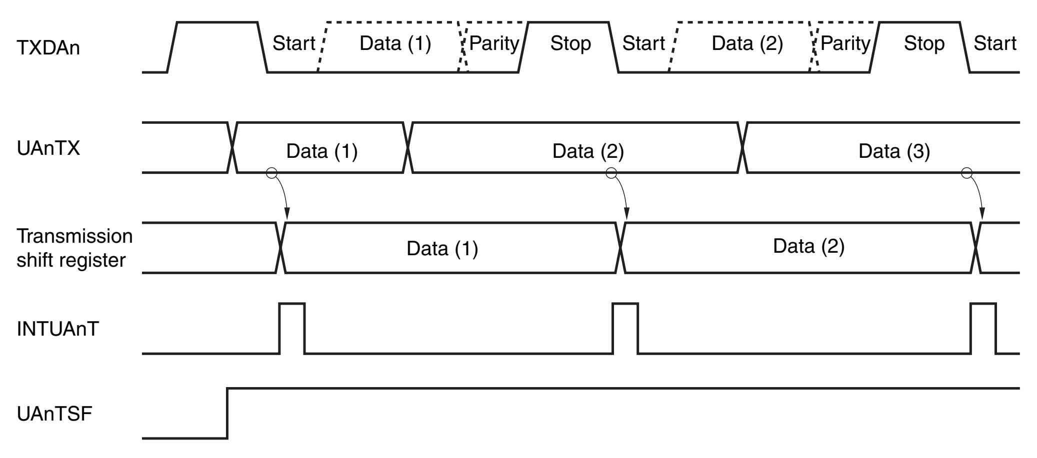


Figure 16-6 Continuous transmission operation timing —transmission start

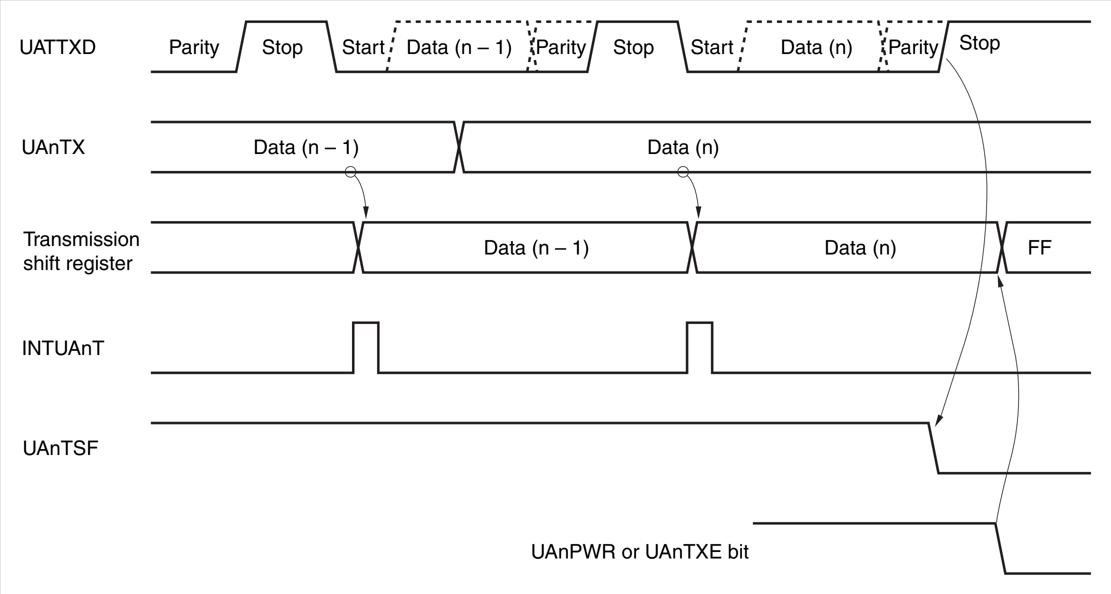


Figure 16-7 Continuous transmission operation timing—transmission end

16.5.7 UART reception

The reception wait status is set by setting the UAnCTL0.UAnPWR bit to 1 and then setting the UAnCTL0.UAnRXE bit to 1. In the reception wait status, the RXDAn pin is monitored and start bit detection is performed.

Start bit detection is performed using a two-step detection routine.

First the rising edge of the RXDAn pin is detected and sampling is started at the falling edge. The start bit is recognized if the RXDAn pin is low level at the start bit sampling point. After a start bit has been recognized, the receive operation starts, and serial data is saved to the UARTAn receive shift register according to the set baud rate.

When the reception complete interrupt request signal (INTUAnR) is output upon reception of the stop bit, the data of the UARTAn receive shift register is written to the UAnRX register. However, if an overrun error (UAnSTR.UAnOVE bit) occurs, the receive data at this time is not written to the UAnRX register and is discarded.

Even if a parity error (UAnSTR.UAnPE bit) or a framing error (UAnSTR.UAnFE bit) occurs during reception, reception continues until the reception position of the first stop bit, and INTUAnR is output following reception completion.

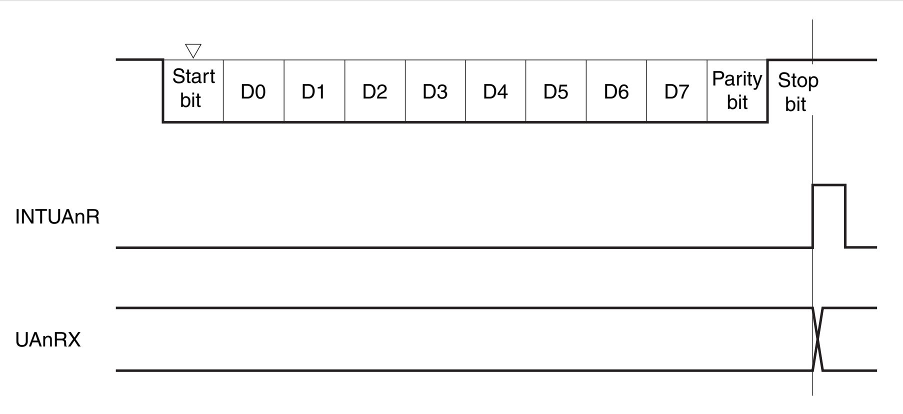


Figure 16-8 UART reception

Caution

1. Be sure to read the UAnRX register even when a reception error occurs. If the UAnRX register is not read, an overrun error occurs during reception of the next data, and reception errors continue occurring indefinitely.
2. The operation during reception is performed assuming that there is only one stop bit. A second stop bit is ignored.
3. When reception is completed, read the UAnRX register after the reception complete interrupt request signal (INTUAnR) has been generated, and clear the UAnPWR or UAnRXE bit to 0. If the UAnPWR or UAnRXE bit is cleared to 0 before the INTUAnR signal is generated, the read value of the UAnRX register cannot be guaranteed.
4. If receive completion processing (INTUAnR signal generation) of UARTAn and the UAnPWR bit = 0 or UAnRXE bit = 0 conflict, the INTUAnR signal may be generated in spite of these being no data stored in the UAnRX register.
   To complete reception without waiting INTUAnR signal generation, be sure to clear (0) the interrupt request flag (UAnRIF) of the UAnRIC register, after setting (1) the interrupt mask flag (UAnRMK) of the interrupt control register (UAnRIC) and then set (1) the UAnPWR bit = 0 or UAnRXE bit = 0.

16.5.8 Reception errors

Errors during a receive operation are of three types: parity errors, framing errors, and overrun errors. Data reception result error flags are set in the UAnSTR register and a reception error interrupt request signal (INTUAnRE) is output when an error occurs.

It is possible to ascertain which error occurred during reception by reading the contents of the UAnSTR register.

Clear the reception error flag by writing 0 to it after reading it.

Table 16-6 Reception error causes

Error flag	Reception error	Cause
UAnPE	Parity error	Received parity bit does not match the setting
UAnFE	Framing error	Stop bit not detected
UAnOVE	Overrun error	Reception of next data completed before data was read from receive buffer

Note Note that even in case of a parity or framing error, data is transferred from the receive shift register to the receive data register UAnRX. Consequently the data from UAnRX must be read. Otherwise an overrun error UAnSTR.UAnOVE will occur at reception of the next data.

In case of an overrun error, the receive shift register data is not transferred to UAnRX, thus the previous data is not overwritten.

16.5.9 Parity types and operations

Caution When using the LIN function, fix the UAnPS1 and UAnPS0 bits of the UAnCTL0 register to 00.

The parity bit is used to detect bit errors in the communication data. Normally the same parity is used on the transmission side and the reception side.

In the case of even parity and odd parity, it is possible to detect odd-count bit errors. In the case of 0 parity and no parity, errors cannot be detected.

(1) Even parity

- During transmission
  The number of bits whose value is “1” among the transmit data, including the parity bit, is controlled so as to be an even number. The parity bit values are as follows.
  - Odd number of bits whose value is “1” among transmit data:1
  - Even number of bits whose value is “1” among transmit data:0
- During reception
  The number of bits whose value is “1” among the reception data, including the parity bit, is counted, and if it is an odd number, a parity error is output.

(2) Odd parity

- During transmission
  Opposite to even parity, the number of bits whose value is “1” among the transmit data, including the parity bit, is controlled so that it is an odd number. The parity bit values are as follows.
  - Odd number of bits whose value is “1” among transmit data: 0
  - Even number of bits whose value is “1” among transmit data: 1
- During reception
  The number of bits whose value is “1” among the receive data, including the parity bit, is counted, and if it is an even number, a parity error is output.

(3) 0 parity

During transmission, the parity bit is always made 0, regardless of the transmit data.

During reception, parity bit check is not performed. Therefore, no parity error occurs, regardless of whether the parity bit is 0 or 1.

(4) No parity

No parity bit is added to the transmit data.

Reception is performed assuming that there is no parity bit. No parity error occurs since there is no parity bit.

16.5.10 Receive data noise filter

This filter samples the RXDAn pin using the base clock of the prescaler output.

When the same sampling value is read twice, the match detector output changes and the RXDAn signal is sampled as the input data. Therefore, data not exceeding 2 clock width is judged to be noise and is not delivered to the internal circuit (see *Figure 16-10*). See *“Base clock“ on page 533* regarding the base clock.

Moreover, since the circuit is as shown in *Figure 16-9*, the processing that goes on within the receive operation is delayed by 3 clocks in relation to the external signal status.

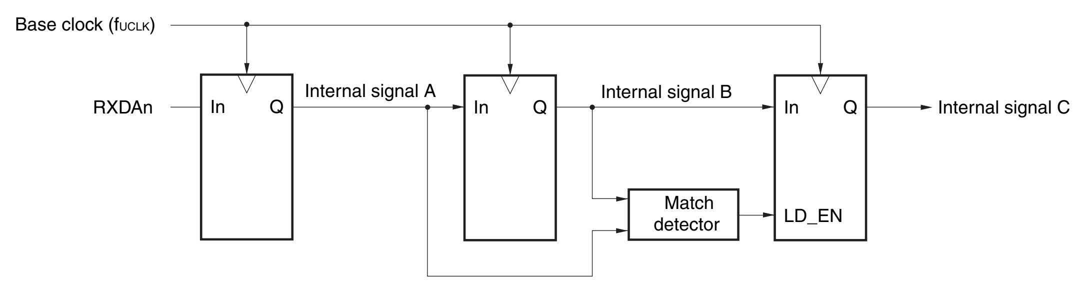


Figure 16-9 Noise filter circuit

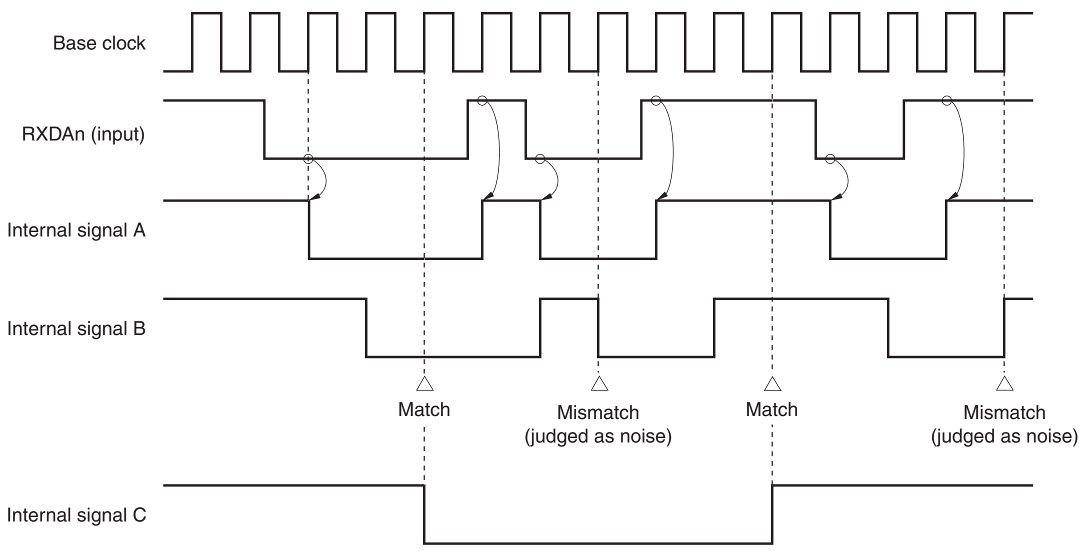


Figure 16-10 Timing of RXDAn signal judged as noise

16.6 Baud Rate Generator

The dedicated baud rate generator consists of a source clock selector block and an 8-bit programmable counter, and generates a serial clock during transmission and reception with UARTAn. Regarding the serial clock, a dedicated baud rate generator output can be selected for each channel.

There is an 8-bit counter for transmission and another one for reception.

16.6.1 Baud Rate Generator configuration

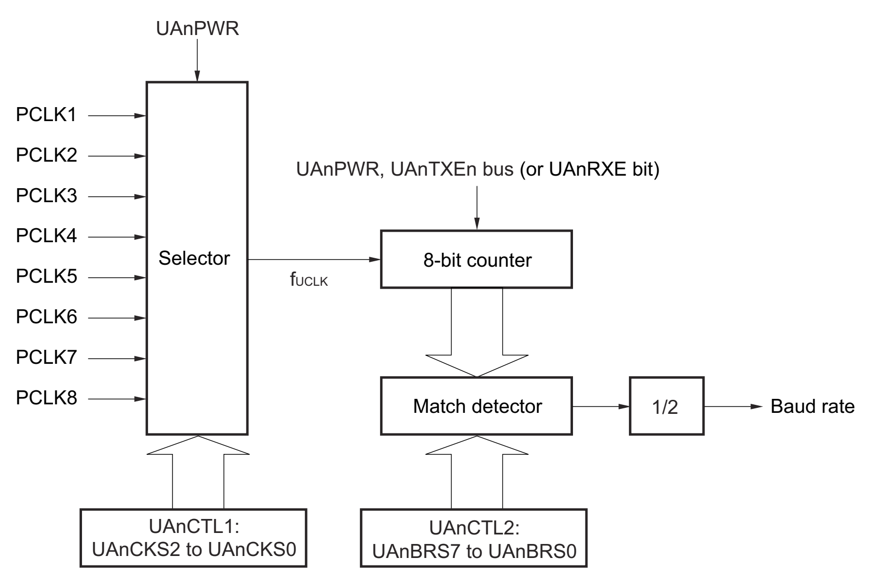


Figure 16-11 Configuration of baud rate generator

(a) Base clock

When the UAnCTL0.UAnPWR bit is 1, the clock selected by the UAnCTL1.UAnCKS[2:0] bits is supplied to the 8-bit counter. This clock is called the base clock. When the UAnPWR bit = 0, f_{UCLK} is fixed to the low level.

(b) Serial clock generation

A serial clock can be generated by setting the UAnCTL1 register and the UAnCTL2 register.

The base clock is selected by UAnCTL1.UAnCKS2 to UAnCTL1.UAnCKS0 bits.

The frequency division value for the 8-bit counter can be set using the UAnCTL2.UAnBRS[7:0] bits.

16.6.2 Baud Rate Generator registers

(1) UAnCTL1 - UARTAn control register 1

The UAnCTL1 register is an 8-bit register that selects the UARTAn base clock.

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

Address <base> + 1_H

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

7	6	5	4	3	2	1	0
0	0	0	0	0	UAnCKS2	UAnCKS1	UAnCKS0
R	R	R	R	R	R/W	R/W	R/W

Caution Clear the UAnCTL0.UAnPWR bit to 0 before rewriting the UAnCTL1 register.

Table 16-7 UAnCTL1 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>2 to 0</td><td>UAnCKS[2:0]</td><td>Base clock f_{UCLK} selection:
<table>
<tr><th>UAnCKS2</th><th>UAnCKS1</th><th>UAnCKS0</th><th>Base clock f_{UCLK}</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>PCLK1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>PCLK2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>PCLK3</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>PCLK4</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>PCLK5</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>PCLK6</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>PCLK7</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>PCLK8</td></tr>
</table>
</td></tr>
</table>

(2) UAnCTL2 - UARTAn control register 2

The UAnCTL2 register is an 8-bit register that selects the baud rate (serial transfer speed) clock of UARTAn.

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

Address <base> + 1_H

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

7	6	5	4	3	2	1	0
UAnBRS7	UAnBRS6	UAnBRS5	UAnBRS4	UAnBRS3	UAnBRS2	UAnBRS1	UAnBRS0
R	R	R	R	R	R/W	R/W	R/W

Caution Clear the UAnCTL0.UAnPWR bit to 0 or clear the UAnTXE and UAnRXE bits to 00 before rewriting the UAnCTL2 register.

Table 16-8 UAnCTL2 register contents

Bit position	Bit name	Function
2 to 0	UAnBRS[7:0]	Serial clock setting

UAn BRS7	UAn BRS6	UAn BRS5	UAn BRS4	UAn BRS3	UAn BRS2	UAn BRS1	UAn BRS0	k	Serial clock
0	0	0	0	0	0	x	x	x	Setting prohibited
0	0	0	0	0	1	0	0	4	$f_{UCLK}/4$
0	0	0	0	0	1	0	1	5	$f_{UCLK}/5$
0	0	0	0	0	1	1	0	6	$f_{UCLK}/6$
:	:	:	:	:	:	:	:	:	:
1	1	1	1	1	1	0	0	252	$f_{UCLK}/252$
1	1	1	1	1	1	0	1	253	$f_{UCLK}/253$
1	1	1	1	1	1	1	0	254	$f_{UCLK}/254$
1	1	1	1	1	1	1	1	255	$f_{UCLK}/255$

Note f_{UCLK}: clock frequency selected by UAnCTL1.UAnCKS[2:0]

16.6.3 Baud rate calculation

The baud rate is obtained by the following equation.

$$\text{Baud rate} = \frac{f_{UCLK}}{2 \times k} \text{ [bps]}$$

f_{UCLK} = Frequency of base clock selected by the UAnCTL1.UAnCKS[2:0]

k = Value set using the UAnCTL2.UAnBRS[7:0] bits
(k = 4, 5, 6, …, 255)

16.6.4 Baud rate error

The baud rate error is obtained by the following equation.

$$\text{Error (\%)} = \left(\frac{\text{Actual baud rate (baud rate with error)}}{\text{Target baud rate (correct baud rate)}} - 1\right) \times 100 \text{ [\%]}$$

Caution

1. The baud rate error during transmission must be within the error tolerance on the receiving side.
2. The baud rate error during reception must satisfy the range indicated in (7) Allowable baud rate range during reception.

Example Base clock frequency = 8MHz

Setting value of

- UAnDTL1.UAnCKS[2:0] = 001B (PCLK2 = 4MHz)

- UAnCTL2.UAnBRS[7:0] = 0000 1101B (k = 13)

Target baud rate = 153,600 bps

Baud rate = 4MHz/ (2 × 13) = 153,846 [bps]

Error = (153,846/153,600 – 1) × 100
= 0.160 [%]

16.6.5 Baud rate setting example

Table 16-9 Baud rate generator setting data

Target baud rate	UAnCTL1		UAnCTL2		Effective baud rate [bps]	Baud rate error (%)
[bps]	Selector	Divider	Divider k			
300	07H	128	68H	104	300.48	0.16
600	07H	128	34H	52	600.96	0.16
1,200	07H	128	1AH	26	1,201.92	0.16
2,400	07H	128	0DH	13	2,403.85	0.16
4,800	06H	64	0DH	13	4,807.69	0.16
9,600	05H	32	0DH	13	9,615.38	0.16
19,200	04H	16	0DH	13	19,230.77	0.16
31,250	05H	32	04H	4	31,250.00	0.00
38,400	03H	8	0DH	13	38,461.54	0.16
76,800	02H	4	0DH	13	76,923.08	0.16
153,600	01H	2	0DH	13	153,846.15	0.16
312,500	00H	1	0DH	13	307,692.31	–1.54

Note *Table 16-9* assumes normal operation mode, i.e. PCLK1=8MHz.

16.6.6 Allowable baud rate range during reception

The baud rate error range at the destination that is allowable during reception is shown below.

Caution The baud rate error during reception must be set within the allowable error range using the following equation.

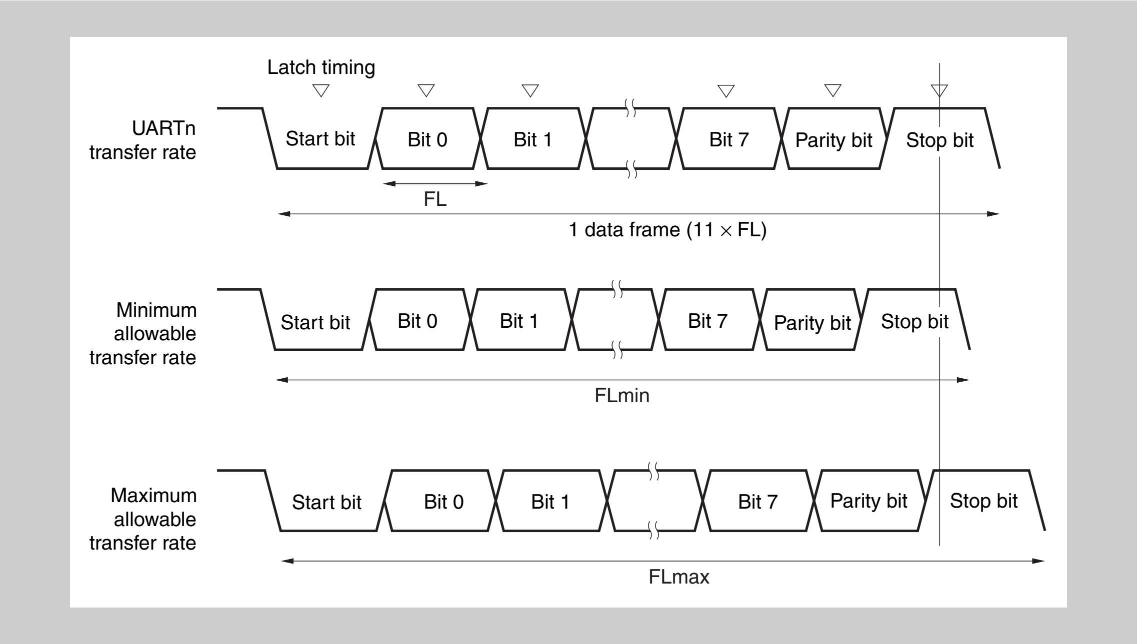


Figure 16-12 Allowable baud rate range during reception

As shown in *Figure 16-12*, the receive data latch timing is determined by the counter set using the UAnCTL2 register following start bit detection. The transmit data can be normally received if up to the last data (stop bit) can be received in time for this latch timing.

When this is applied to 11-bit reception, the following is the theoretical result.

$FL = (Brate)^{-1}$

Brate: UARTAn baud rate

k: Setting value of UAnCTL2.UAnBRS[7:0]

FL: 1-bit data length

Latch timing margin: 2 clocks

Minimum allowable transfer rate:

$$FL_{min} = 11 \times FL - \frac{k-2}{2k} \times FL = \frac{21k+2}{2k} \times FL$$

Therefore, the maximum baud rate that can be received by the destination is as follows.

$$BRmax = (FLmin/11)^{-1} = \frac{22k}{21k+2} \times Brate$$

Similarly, obtaining the following maximum allowable transfer rate yields the following.

$$\frac{10}{11} \times FLmax = 11 \times FL - \frac{k+2}{2k} \times FL = \frac{21k-2}{2k} \times FL$$

$$FLmax = \frac{21k-2}{20k} \times FL \times 11$$

Therefore, the minimum baud rate that can be received by the destination is as follows.

$$\mathrm{BRmin} = (\mathrm{FLmax}/11)^{-1} = \frac{20k}{21k-2} \times \mathrm{Brate}$$

Obtaining the allowable baud rate error for UARTAn and the destination from the above-described equations for obtaining the minimum and maximum baud rate values yields the following.

Table 16-10 Maximum/Minimum allowable baud rate error

Division ratio (k)	Maximum allowable baud rate error	Minimum allowable baud rate error
4	+2.32%	-2.43%
8	+3.52%	-3.61%
20	+4.26%	-4.30%
50	+4.56%	-4.58%
100	+4.66%	-4.67%
255	+4.72%	-4.72%

Note
1. The reception accuracy depends on the bit count in 1 frame, the input clock frequency, and the division ratio (k). The higher the input clock frequency and the larger the division ratio (k), the higher the accuracy.
2. k: Setting value of UAnCTL2.UAnBRS[7:0]

16.6.7 Baud rate during continuous transmission

During continuous transmission, the transfer rate from the stop bit to the next start bit is usually 2 base clocks longer. However, timing initialization is performed via start bit detection by the receiving side, so this has no influence on the transfer result.

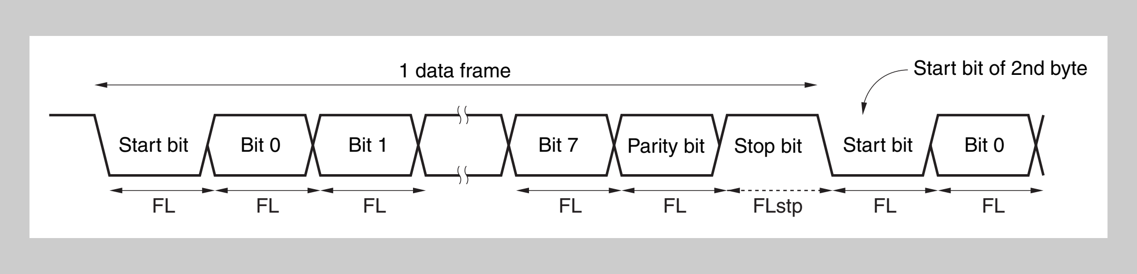


Figure 16-13 Transfer rate during continuous transfer

Assuming 1 bit data length: FL; stop bit length: FLstp; and base clock frequency: f_{UCLK}, we obtain the following equation.

$$FLstp = FL + 2/f_{UCLK}$$

Therefore, the transfer rate during continuous transmission is as follows.

$$\text{Transfer rate} = 11 \times FL + (2/f_{UCLK})$$

16.7 Cautions

- When the clock supply to UARTAn is stopped (for example, in IDLE or STOP mode), the operation stops with each register retaining the value it had immediately before the clock supply was stopped. The TXDAn pin output also holds and outputs the value it had immediately before the clock supply was stopped. However, the operation is not guaranteed after the clock supply is resumed. Therefore, after the clock supply is resumed, the circuits should be initialized by setting the UAnCTL0.UAnPWR, UAnCTL0.UAnRXEn, and UAnCTL0.UAnTXEn bits to 000.

Chapter 17 Clocked Serial Interface (CSIB)

The V850E/Dx3 microcontrollers have following instances of the clocked serial interface CSIB:

CSIB	**µPD70F3427, µPD70F3426 µPD70F3425, µPD70F3424**	**µPD70F3423, µPD70(F)3422 µPD70(F)3421, µPD70(F)3420**
Instances	3	2
Names	CSIB0 to CSIB2	CSIB0 to CSIB1

Throughout this chapter, the individual instances of clocked serial interface are identified by "n", for example CSIBn, or CBnCTL0 for the control register 0 of CSIBn.

17.1 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
- 3 interrupt request signals (INTCBnT, INTCBnR, INTCBnRE)
- Serial clock and data phase switchable
- Transfer data length selectable in 1-bit units between 8 and 16 bits
- Transfer data MSB-first/LSB-first switchable
- 3-wire transfer

3-wire transfer	SOBn:	Serial data output
	SIBn:	Serial data input
	SCKBn:	Serial clock input/output

  Transmission mode, reception mode, and transmission/reception mode specifiable
- DMA support
- Dedicated baud rate generator for each interface instance
- Modulated and stable clock sources available

17.2 Configuration

The following shows the block diagram of CSIBn.

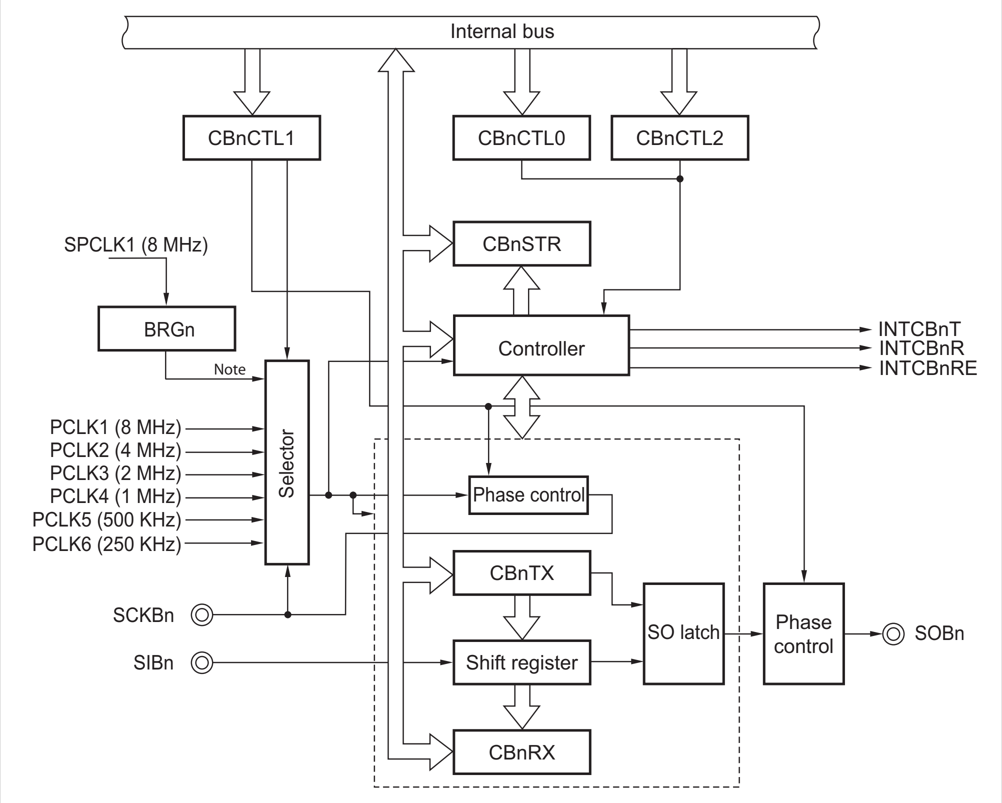


Figure 17-1 Block diagram of CSIBn

Note The clock is generated by the dedicated baud rate generator BRGn.

17.3 CSIB Control Registers

The clocked serial interfaces CSIBn are controlled and operated by means of the following registers:

Table 17-1 CSIBn registers overview

Register name	Shortcut	Address
CSIBn control register 0	CBnCTL0	<base>
CSIBn control register 1	CBnCTL1	<base> + 1_H
CSIBn control register 2	CBnCTL2	<base> + 2_H
CSIBn status register	CBnSTR	<base> + 3_H
CSIBn receive data register	CBnRX	<base> + 4_H
CSIBn transmit data register	CBnTX	<base> + 6_H

Table 17-2 CSIBn register base address

Timer	Base address
CSIB0	FFFF $DA00_H$
CSIB1	FFFF $FD10_H$
CSIB2	FFFF $FD20_H$

(1) CBnCTL0 - CSIBn control register 0

CBnCTL0 is a register that controls the CSIBn serial transfer operation.

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

Address <base>

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

7	6	5	4	3	2	1	0
CBnPWR	CBnTXE[a]	CBnRXE[a]	CBnDIR[a]	0	0	CBnTMS[a]	CBnSCE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

[a] These bits can only be rewritten when the CBnPWR bit = 0. However, CBnPWR bit = 1 can also be set at the same time as rewriting these bits.

Table 17-3 CBnCTL0 register contents (1/2)

Bit position	Bit name	Function
7	CBnPWR	CSIBn operation disable/enable: 0: Disable CSIBn operation and reset the CSIBn registers 1: Enable CSIBn operation The CBnPWR bit controls the CSIBn operation and resets the internal circuit.
6	CBnTXE	Transmit operation disable/enable: 0: Disable transmit operation 1: Enable transmit operation The SOBn output is low level when the CBnTXE bit is 0.
5	CBnRXE	Receive operation disable/enable: 0: Disable receive operation 1: Enable receive operation When the CBnRXE bit is cleared to 0, no reception complete interrupt is output even when the prescribed data is transferred in order to disable the receive operation, and the receive data (CBnRX register) is not updated.

Table 17-3 CBnCTL0 register contents (2/2)

Bit position	Bit name	Function
4	CBnDIR	Transfer direction mode specification (MSB/LSB): 0: MSB first transfer 1: LSB first transfer
1	CBnTMS	Transfer direction mode specification (MSB/LSB): 0: Single transfer mode 1: Continuous transfer mode
0	CBnSCE	Transfer direction mode specification (MSB/LSB): 0: Communication start trigger invalid 1: Communication start trigger valid • In master mode This bit enables or disables the communication start trigger. (a)In single transmission or transmission/reception mode, or continuous transmission or continuous transmission/reception mode A communication operation can be started only when the CBnSCE bit is 1. Set the CBnSCE bit to 1. (b)In single reception mode Clear the CBnSCE bit to 0 before reading the receive data (CBnRX register). If the CBnSCE bit is read while it is 1, the next communication operation is started. (c)In continuous reception mode Clear the CBnSCE bit to 0 one communication clock before reception of the last data is completed The CBnSCE bit is not cleared to 0 one communication clock before the completion of the last data reception, the next communication operation is automatically started. • In slave mode This bit enables or disables the communication start trigger. Set the CBnSCE bit to 1.

(2) CBnCTL1 - CSIBn control register 1

CBnCTL1 is an 8-bit register that controls the CSIBn serial transfer operation.

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

Address <base> + 1_H

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

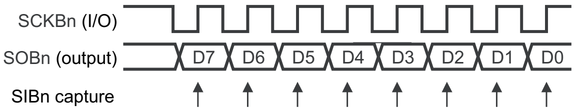
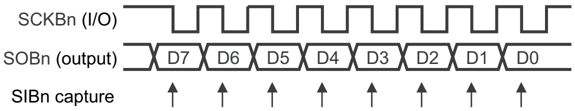
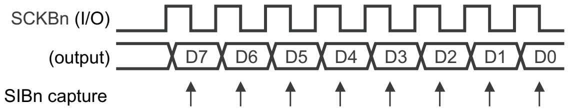
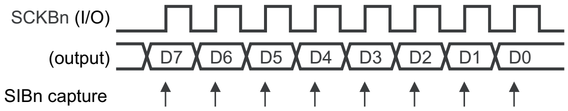
7	6	5	4	3	2	1	0
0	0	0	CBnCKP	CBnDAP	CBnCKS2	CBnCKS1	CBnCKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution The CBnCTL1 register can be rewritten only when the CBnCTL0.CBnPWR bit = 0.

Table 17-4 CBnCTL1 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>4
3</td><td>CBnCKP
CBnDAP</td><td>Specification of data transmission/reception timing in relation to SCKBn.
Refer to *Table 17-5*.</td></tr>
<tr><td>2 to 0</td><td>CBnCKS[2:0]</td><td>Communication clock setting
<table>
<tr><th>CBnCKS2</th><th>CBnCKS1</th><th>CBnCKS0</th><th>Communication clock</th><th>Mode</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>f_{BRGn}</td><td>Master</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>f_{PCLK1}</td><td>Master</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>f_{PCLK2}</td><td>Master</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>f_{PCLK3}</td><td>Master</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>f_{PCLK4}</td><td>Master</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>f_{PCLK5}</td><td>Master</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>f_{PCLK6}</td><td>Master</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>External clock SCKBn</td><td>Slave</td></tr>
</table>
</td></tr>
</table>

Table 17-5 Specification of data transmission/reception timing in relation to SCKBn

Communication type	CBnCKP	CBnDAP	SIBn/SOBN timing in relation to SCKBn
Communication type 1	0	0	
Communication type 2	0	1	
Communication type 3	1	0	
Communication type 4	1	1	

(3) CBnCTL2 - CSIBn control register 2

CBnCTL2 is an 8-bit register that controls the number of CSIBn serial transfer bits.

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

Address <base> + 2_H

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

7	6	5	4	3	2	1	0
0	0	0	0	CBnCL3	CBnCL2	CBnCL1	CBnCL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution The CBnCTL2 register can be rewritten only when the CBnCTL0.CBnPWR bit = 0 or when both the CBnTXE and CBnRXE bits = 0.

Table 17-6 CBnCTL2 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3 to 0</td><td>CBnCL[3:0]</td><td>Number of serial transfer bits
<table>
<tr><th>CBnCL3</th><th>CBnCL2</th><th>CBnCL1</th><th>CBnCL0</th><th>Number of serial transfer bits</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>8 bits</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>9 bits</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>10 bits</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>11 bits</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>12 bits</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>13 bits</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>14 bits</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>15 bits</td></tr>
<tr><td>1</td><td>x</td><td>x</td><td>x</td><td>16 bits</td></tr>
</table>
</td></tr>
</table>

Note If the number of transfer bits is other than 8 or 16, prepare and use data stuffed from the LSB of the CBnTX and CBnRX registers.

(a) Transfer data length change function

The CSIBn transfer data length can be set in 1-bit units between 8 and 16 bits using the CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits.

When the transfer bit length is set to a value other than 16 bits, set the data to the CBnTX or CBnRX register starting from the LSB, regardless of whether the transfer start bit is the MSB or LSB. Any data can be set for the higher bits that are not used, but the receive data becomes 0 following serial transfer.

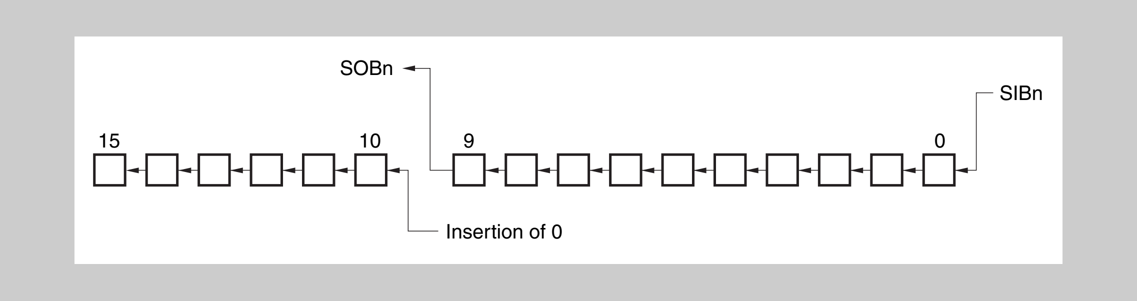


Figure 17-2 (i) Transfer bit length = 10 bits, MSB first

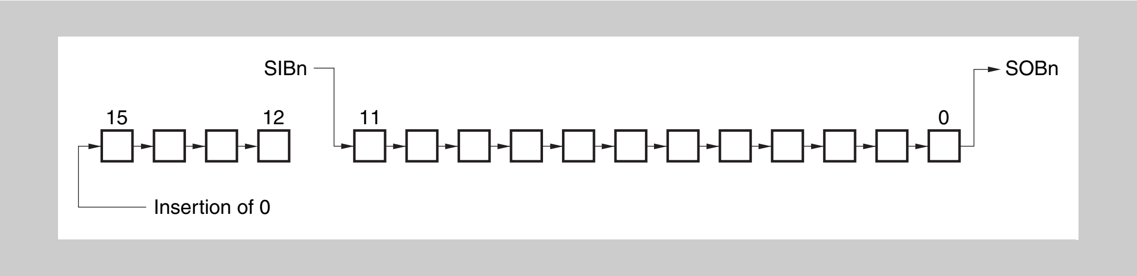


Figure 17-3 (ii) Transfer bit length = 12 bits, LSB first

(4) CBnSTR - CSIBn status register

CBnSTR is an 8-bit register that displays the CSIBn status.

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

Address <base> + 3_H

Initial Value 00_H. This register is cleared by any reset.
In addition to reset input, the CBnSTR register can be initialized by clearing the CBnCTL0.CBnPWR bit to 0.

7	6	5	4	3	2	1	0
CBnTSF	0	0	0	0	0	0	CBnOVE
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 17-7 CBnSTR register contents

Bit position	Bit name	Function
7	CBnTSF	Communication status flag 0: Communication stopped 1: Communicating During transmission, this register is set when data is prepared in the CBnTX register, and during reception, it is set when a dummy read of the CBnRX register is performed. When transfer ends, this flag is cleared to 0 at the last edge of the clock.
0	CBnOVE	Overrun error flag 0: No overrun 1: Overrun • An overrun error occurs when the next reception starts without performing a CPU read of the value of the receive buffer, upon completion of the receive operation. The CBnOVE flag displays the overrun error occurrence status in this case. • The CBnOVE flag is cleared by writing 0 to it. It cannot be set even by writing 1 to it.

Note In case of an overrun error, the reception error interrupt INTCBnRE behaves different, depending on the transfer mode:

- Continuous transfer mode
  The reception error interrupt INTCBnRE is generated instead of the reception completion interrupt INTCBnR.
- Single transfer mode
  No interrupt is generated.

In either case the overflow flag CBnSTR.CBnOVE is set to 1 and the previous data in CBnRX will be overwritten with the new data.

(5) CBnRX - CSIBn receive data register

The CBnRX register is a 16-bit buffer register that holds receive data.

Access This register can be read-only in 16-bit units.
If the transfer data length is 8 bits, the lower 8 bits of this register are read-only in 8-bit units as the CBnRXL register.

Address <base> + 4_H

Initial Value 0000_H. This register is cleared by any reset.
In addition to reset input, the CBnRX register can be initialized by clearing (to 0) the CBnPWR bit of the CBnCTL0 register.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">Receive data</td></tr>
<tr><td colspan="16">R</td></tr>
</table>

The receive operation is started by reading the CBnRX register in the reception enabled status.

(6) CBnTX - CSIB transmit data register

The CBnTX register is a 16-bit buffer register used to write the CSIBn transfer data.

Access This register can be read/written in 16-bit units.
If the transfer data length is 8 bits, the lower 8 bits of this register are read/write in 8-bit units as the CBnTXL register.

Address <base> + 6_H

Initial Value 0000_H. This register is cleared by any reset.
In addition to reset input, the CBnTX register can be initialized by clearing (to 0) the CBnPWR bit of the CBnCTL0 register.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">Transmit data</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

The transmit operation is started by writing data to the CBnTX register in the transmission enabled status.

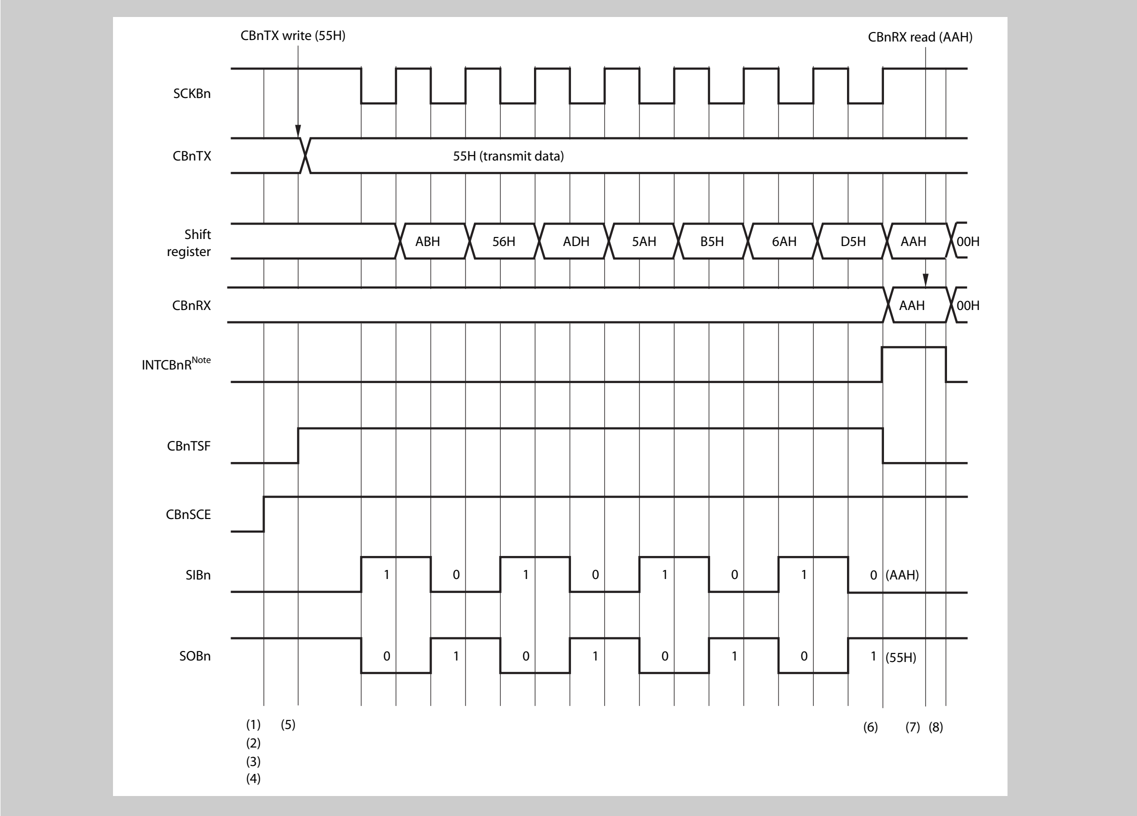
Note The communication start conditions are shown below:

- Transmission mode (CBnTXE bit = 1, CBnRXE bit = 0):
  Write to CBnTX register
- Transmission/reception mode (CBnTXE bit = 1, CBnRXE bit = 1):
  Write to CBnTX register
- Reception mode (CBnTXE bit = 0, CBnRXE bit = 1):
  Read from CBnRX register

17.4 Operation

17.4.1 Single transfer mode (master mode, transmission/reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 1 (see 16.4 (2) CSIBn control register 1 (CBnCTL1), transfer data length = 8 bits (CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnTXE, CBnRXE, and CBnSCE bits of the CBnCTL0 register to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the transmission/reception enabled status.

(4) Set the CBnPWR bit to 1 to enable the CSIBn operation.

(5) Write transfer data to the CBnTX register (transmission start).

(6) The reception complete interrupt request signal (INTCBnR) is output.

(7) Read the CBnRX register before clearing the CBnPWR bit to 0.

(8) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop operation of CSIBn (end of transmission/reception).

To continue transfer, repeat steps (5) to (7) before (8).

In transmission mode or transmission/reception mode, communication is not started by reading the CBnRX register.

Note

1. In single transmission or single transmission/reception mode, the INTCBnT signal is not generated. When communication is complete, the INTCBnR signal is generated.
2. The processing of steps (3) and (4) can be set simultaneously.

Caution In case the CSIB interface is operating in

- single transmit/reception mode (CBnCTL0.CBnTMS = 0)
- communication type 2 respectively type 4 (CBnCTL1.CBnDAP = 1)

pay attention to following effect:

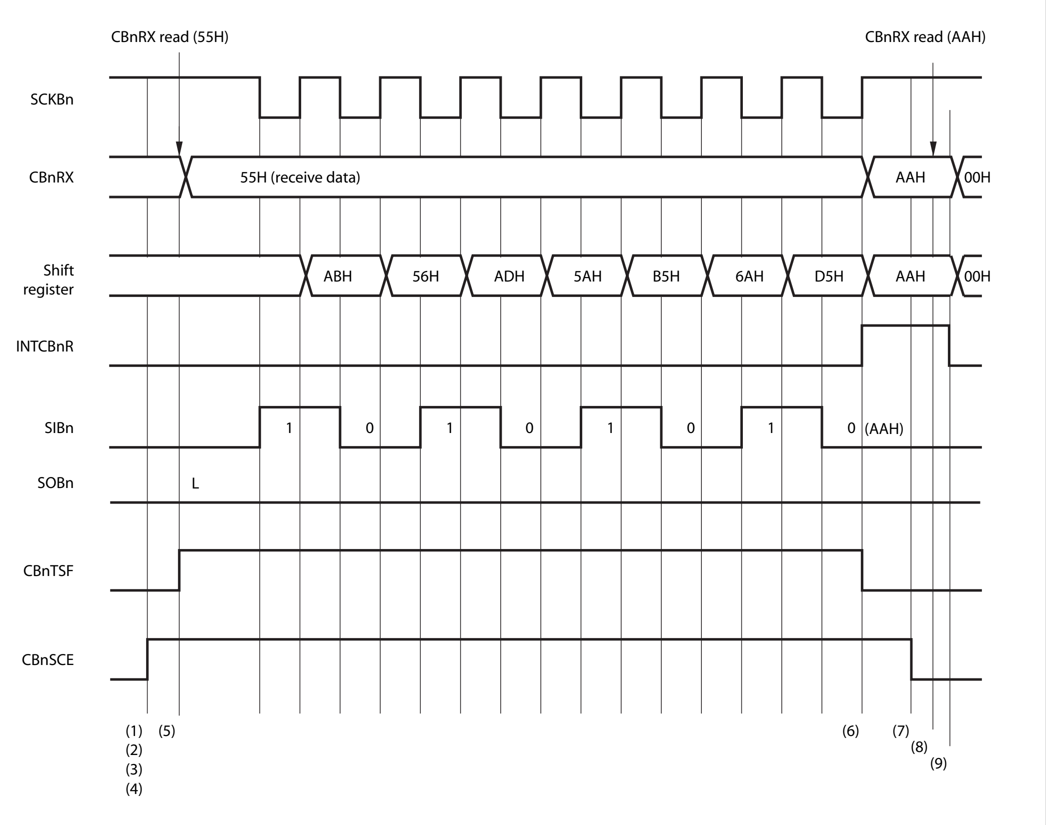
In case the next transmit should be initiated immediately after the occurrence of the reception completion interrupt INTCBnR any write to the CBnTX register is ignored as long as the communication status flag is still reflecting an ongoing communication (CBnTSF = 1). Thus the new transmission will not be started.

For transmitting data continuously use one of the following options:

- Use continuous transfer mode (CBnCTL0.CBnTMS = 1). 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.

17.4.2 Single transfer mode (master mode, reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 1 (see 16.4 (2) CSIBn control register 1 (CBnCTL1), transfer data length = 8 bits (CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnCTL0.CBnRXE and CBnCTL0.CBnSCE bits to 1, CBnCTL0.TXE to 0, at the same time as specifying the transfer mode using the CBnDIR bit, to set the reception enabled status.

(4) Set the CBnPWR bit to 1 to enable the CSIBn operation.

(5) Perform a dummy read of the CBnRX register (reception start trigger).

(6) The reception complete interrupt request signal (INTCBnR) is output.

(7) Set the CBnSCE bit to 0 to set the final receive data status.

(8) Read the CBnRX register.

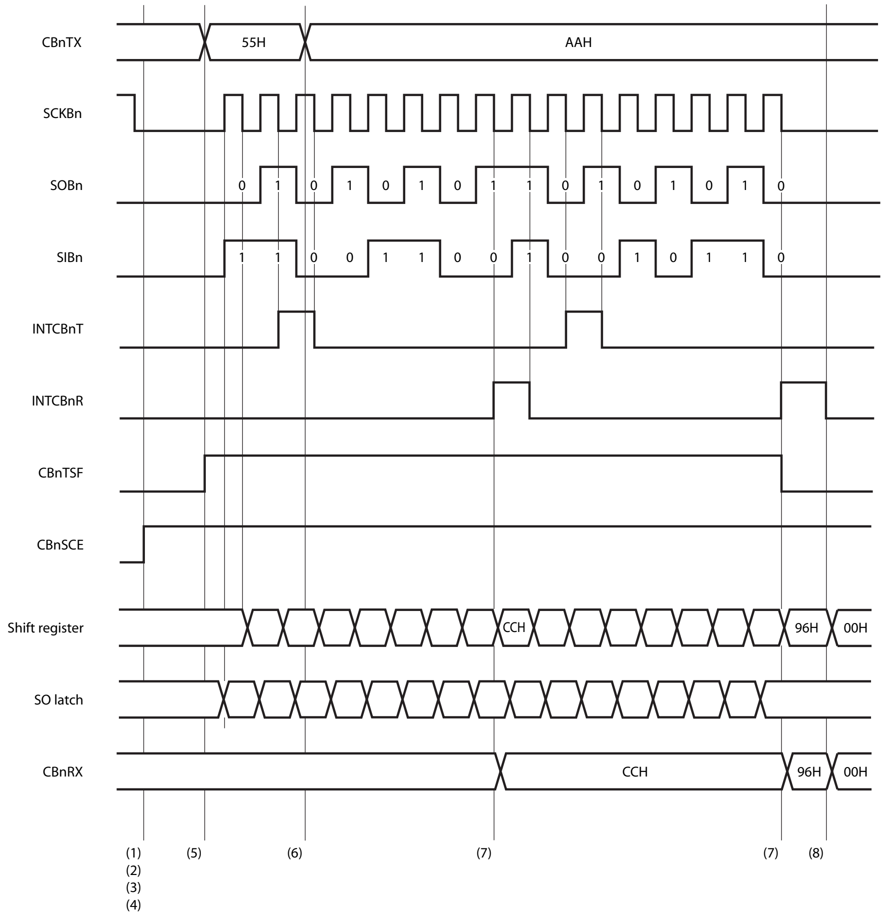
(9) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the CSIBn operation (end of reception).

To continue transfer, repeat steps (5) and (6) before (7). (At this time, (5) is not a dummy read, but a receive data read combined with the reception trigger.)

Note The processing of steps (3) and (4) can be set simultaneously.

17.4.3 Continuous mode (master mode, transmission/reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 3 (see 16.4 (2) CSIBn control register 1 (CBnCTL1)), transfer data length = 8 bits (CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnTXE, CBnRXE, and CBnSCE bits of the CBnCTL0 register to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the transmission/reception enabled status.

(4) Set the CBnPWR bit to 1 to enable the CSIBn operation.

(5) Write transfer data to the CBnTX register (transmission start).

(6) The transmission enable interrupt request signal (INTCBnT) is received and transfer data is written to the CBnTX register.

(7) The reception complete interrupt request signal (INTCBnR) is output.

Read the CBnRX register before the next receive data arrives or before the CBnPWR bit is cleared to 0.

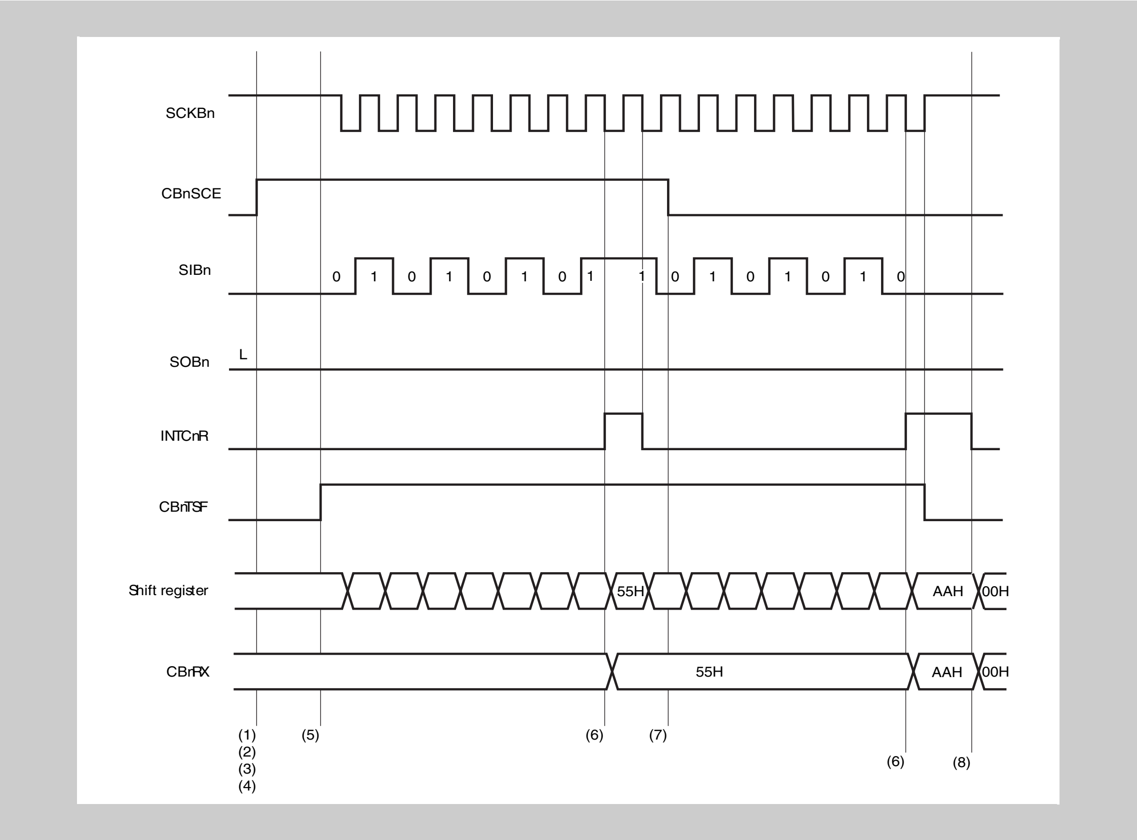
(8) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation of CSIBn (end of transmission/reception).

To continue transfer, repeat steps (5) to (7) before (8).

In transmission mode or transmission/reception mode, the communication is not started by reading the CBnRX register.

17.4.4 Continuous mode (master mode, reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 2 (see 16.4 (2) CSIBn control register 1 (CBnCTL1)), transfer data length = 8 bits (CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnCTL0.CBnRXE bit to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the reception enabled status.

(4) Set the CBnPWR bit to 1 to enable the CSIBn operation.

(5) Perform a dummy read of the CBnRX register (reception start trigger).

(6) The reception complete interrupt request signal (INTCBnR) is output.

Read the CBnRX register before the next receive data arrives or before the CBnPWR bit is cleared to 0.

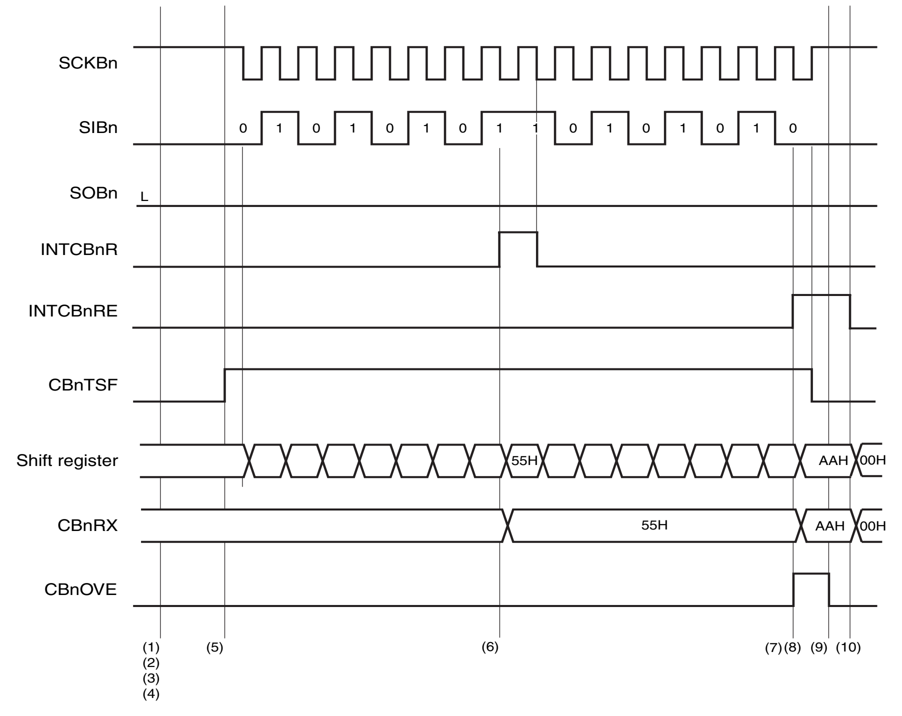
(7) Set the CBnCTL0.CBnSCE bit = 0 while the last data being received to set the final receive data status.

(8) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation of CSIBn (end of reception).

To continue transfer, repeat steps (5) and (6) before (7).

17.4.5 Continuous reception mode (error)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 2 (see 16.4 (2) CSIBn control register 1 (CBnCTL1)), transfer data length = 8 bits (CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnCTL0.CBnRXE bit to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the reception enabled status.

(4) Set the CBnPWR bit = 1 to enable CSIBn operation.

(5) Perform a dummy read of the CBnRX register (reception start trigger).

(6) The reception complete interrupt request signal (INTCBnR) is output.

(7) If the data could not be read before the end of the next transfer, the CBnSTR.CBnOVE flag is set to 1 upon the end of reception and the reception error interrupt INTCBnRE is output.

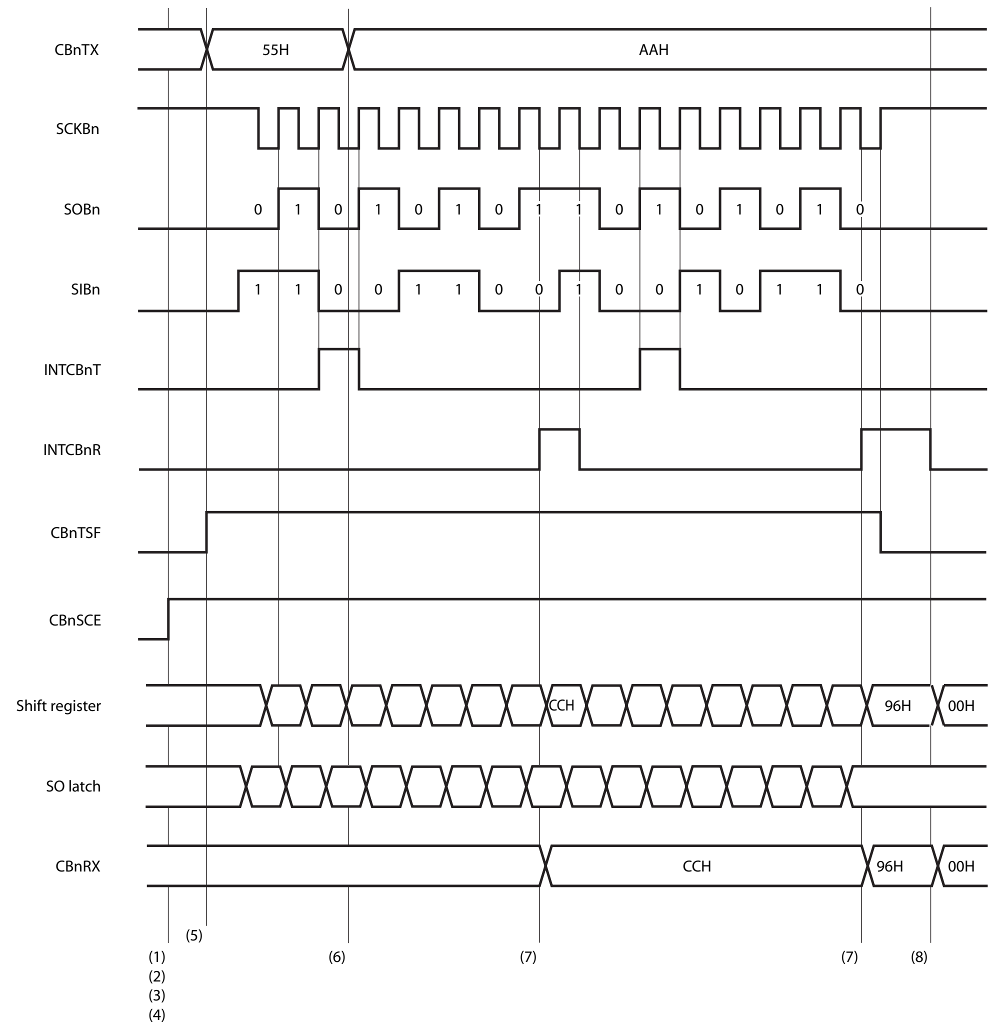
(8) Overrun error processing is performed after checking that the CBnOVE bit = 1 in the INTCBnRE interrupt servicing.

(9) Clear CBnOVE bit to 0.

(10)Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation CSIBn (end of reception).

17.4.6 Continuous mode (slave mode, transmission/reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 2 (see 16.4 (2) CSIBn control register 1 (CBnCTL1)), transfer data length = 8 bits (CBnCTL2.CSnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnTXE, CBnRXE and CBnSCE bits of the CBnCTL0 register to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the transmission/reception enabled status.

(4) Set the CBnPWR bit to 1 to enable supply of the CSIBn operation.

(5) Write the transfer data to the CBnTX register.

(6) The transmission enable interrupt request signal (INTCBnT) is received and the transfer data is written to the CBnTX register.

(7) The reception complete interrupt request signal (INTCBnR) is output.

Read the CBnRX register.

(8) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation of CSIBn (end of transmission/reception).

To continue transfer, repeat steps (5) to (7) before (8).

Note In order to start the entire data transfer the CBnTX register has to be written initially, as done in step (5) above. If this step is omitted also no data will be received.

Discontinued transmission In case the CSIB is operating in continuous slave transmission mode (CBnCTL0.CBnTMS = 1, CBnCTL1.CBnCKS[2:0] = 111_B) and new data is not written to the CBnTX register the SOBn pin outputs the level of the last bit.

Table 17-8 outlines this behaviour.

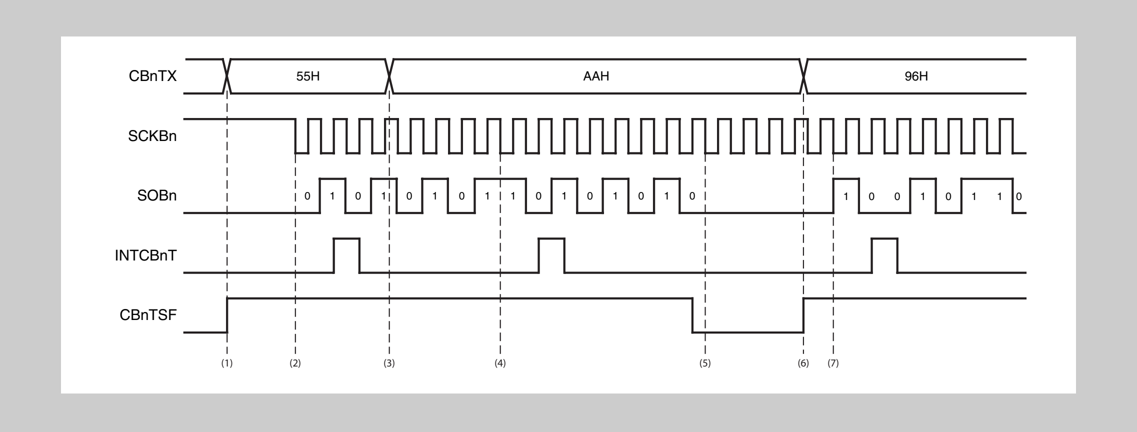


Table 17-8 **Discontinued slave transmission**

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.

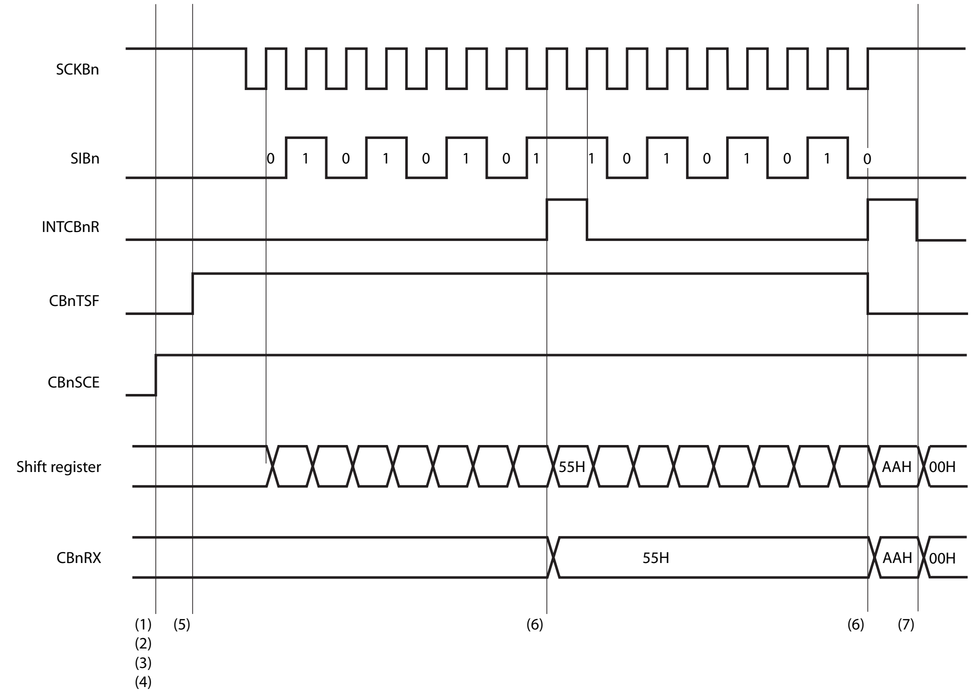
(6) New data (96_H) is written to CBnTX.

(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.

17.4.7 Continuous mode (slave mode, reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 1 (see 16.4 (2) CSIBn control register 1 (CBnCTL1)), transfer data length = 8 bits (CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnCTL0.CBnRXE and CBnCTL0.CBnSCE bits to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the reception enabled status.

(4) Set the CBnPWR bit = 1 to enable CSIBn operation.

(5) Perform a dummy read of the CBnRX register (reception start trigger).

(6) The reception complete interrupt request signal (INTCBnR) is output.
Read the CBnRX register.

(7) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation of CSIBn (end of reception).

To continue transfer, repeat steps (5) and (6) before (7).

17.4.8 Clock timing

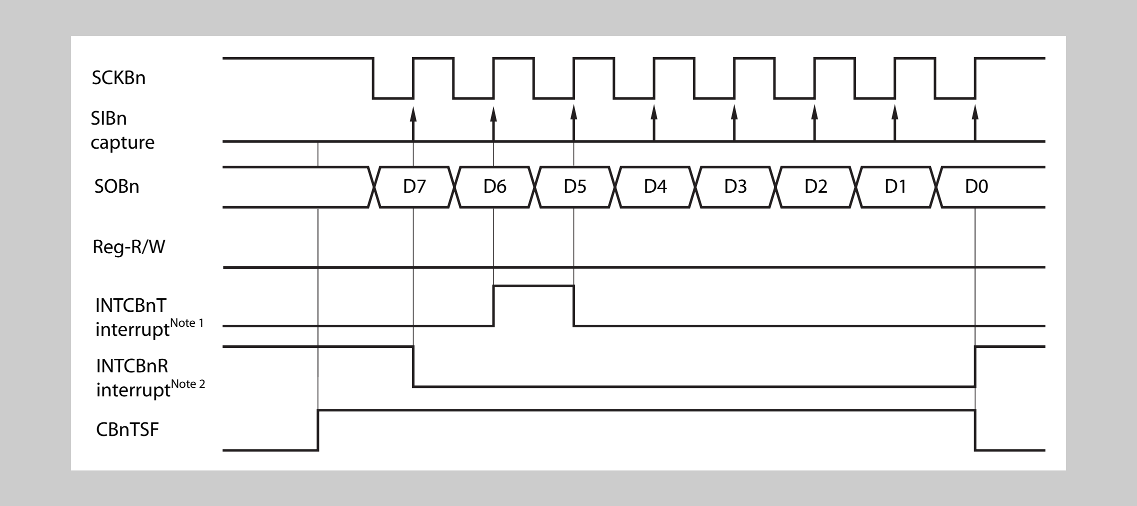


Figure 17-4 (i) Communication type 1 (CBnCKP = 0, CBnDAP = 0)

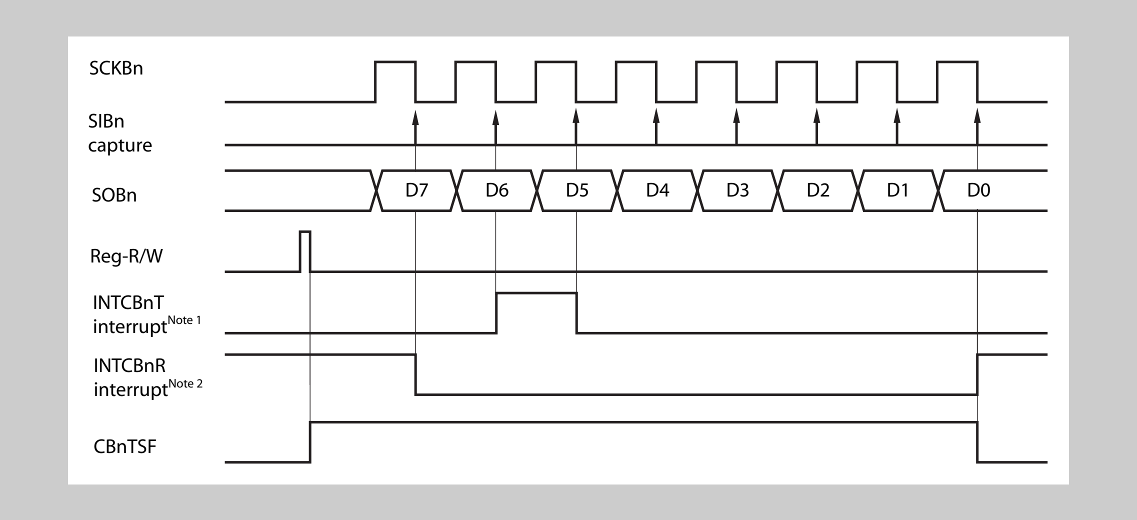


Figure 17-5 (ii) Communication type 3 (CBnCKP = 1, CBnDAP = 0)

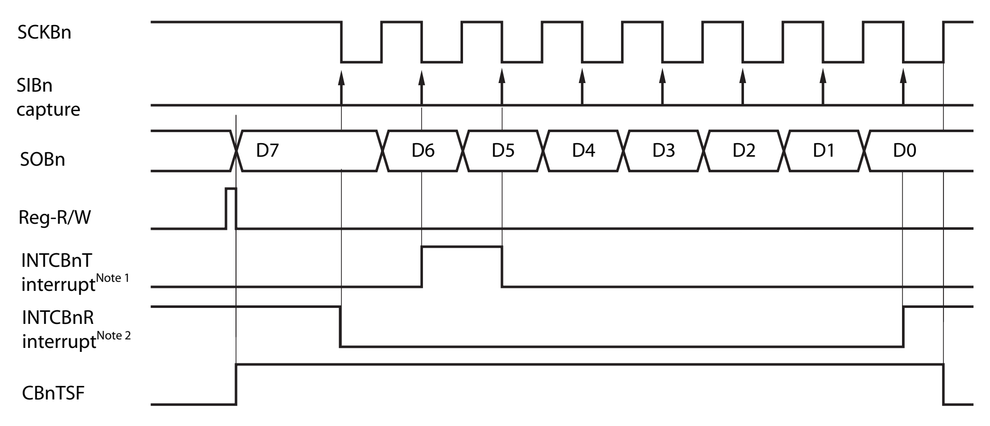


Figure 17-6 (iii) Communication type 2 (CBnCKP = 0, CBnDAP = 1)

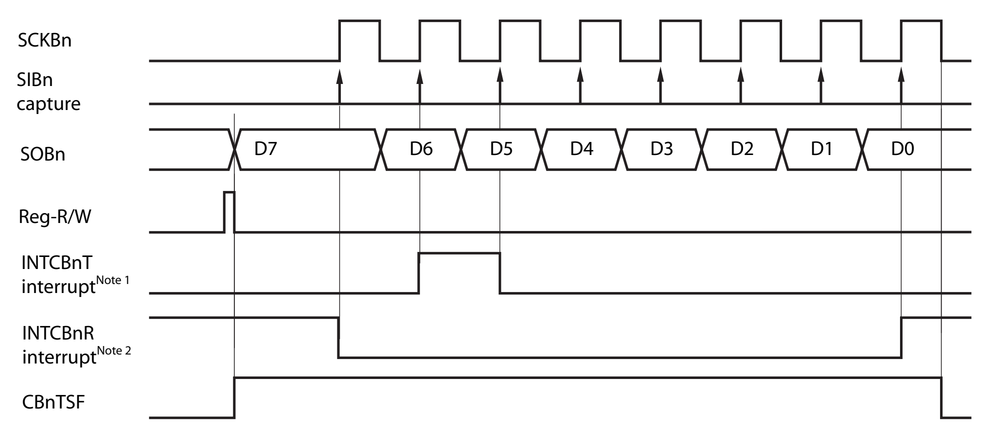


Figure 17-7 (iv) Communication type 4 (CBnCKP = 1, CBnDAP = 1)

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.

17.5 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.

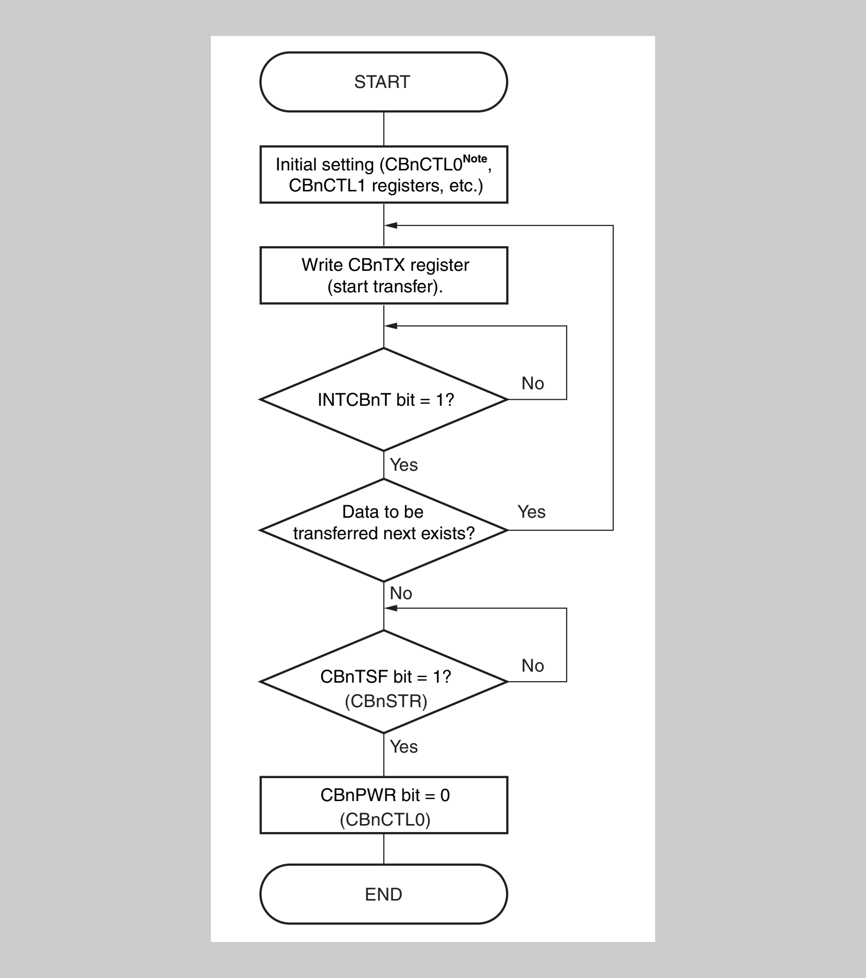
CBnTXE	CBnDAP	CBnDIR	SOBn pin output
0	×	×	Fixed to low level
1	0	×	SOBn latch value (low level)
	1	0	CBnTXn value (MSB)
		1	CBnTXn value (LSB)

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

17.6 Operation Flow

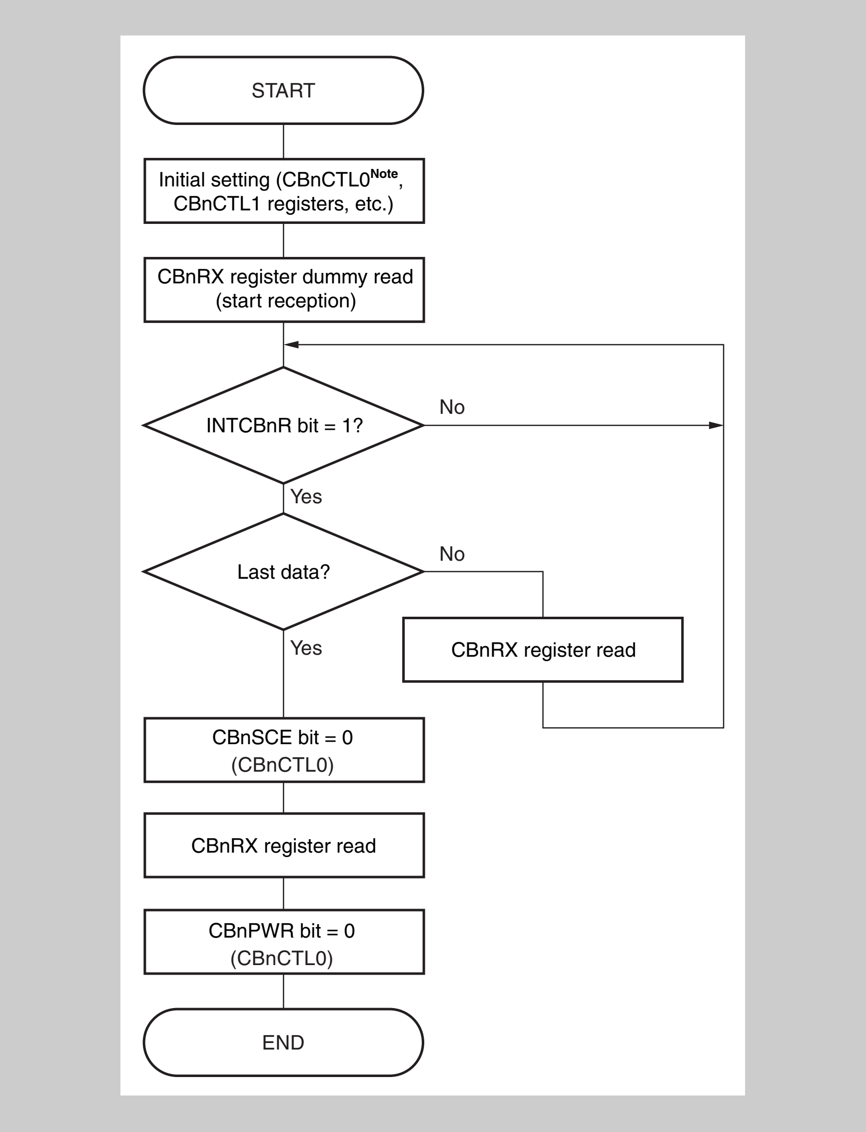
(1) Single transmission



Note Set the CBnSCE bit to 1 in the initial setting.

Caution In the slave mode, data cannot be correctly transmitted if the next transfer clock is input earlier than the CBnTX register is written.

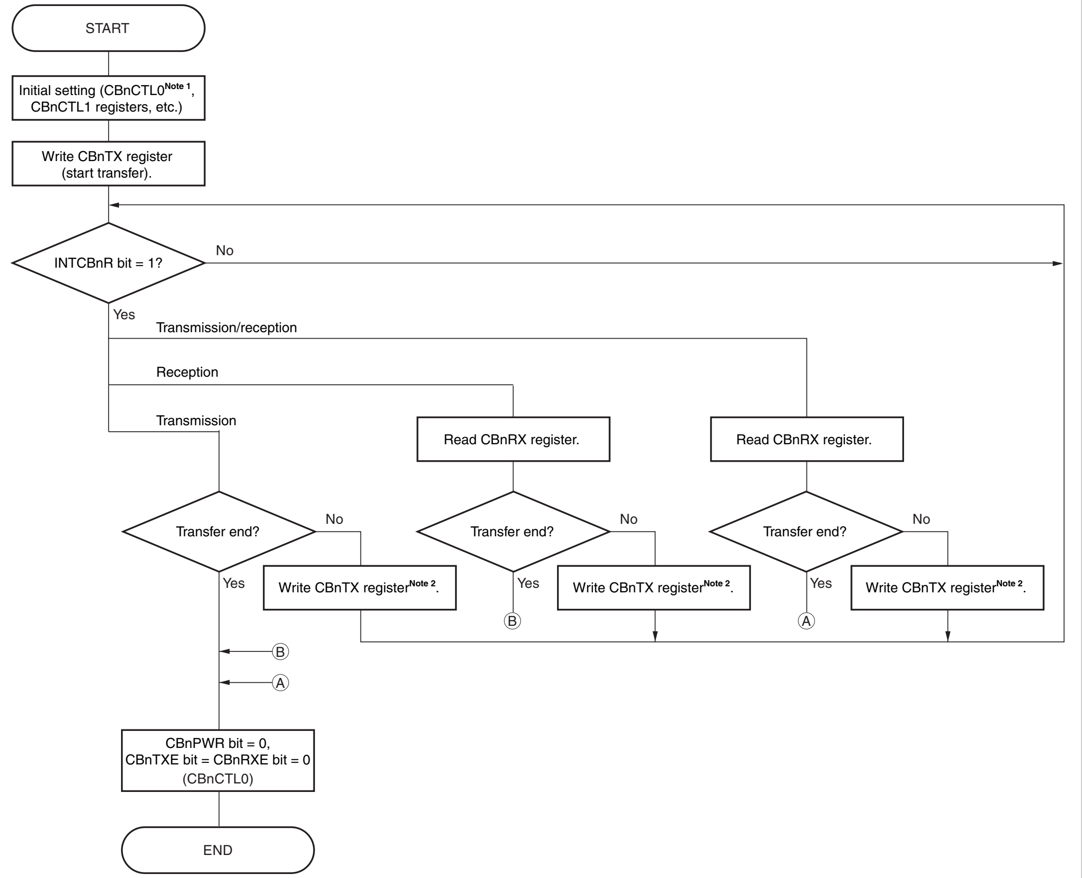
(2) Single reception



Note Set the CBnSCE bit to 1 in the initial setting.

Caution In the single mode, data cannot be correctly received if the next transfer clock is input earlier than the CBnRX register is read.

(3) Single transmission/reception

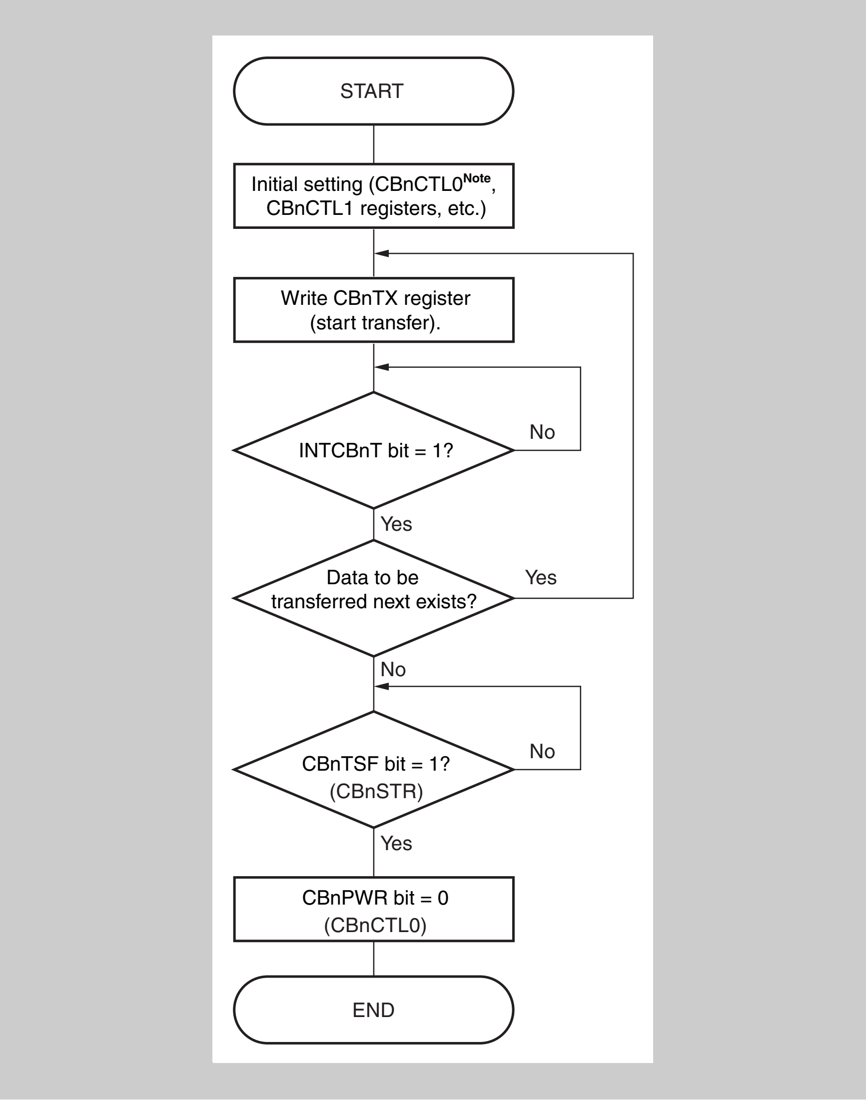


Note 1. Set the CBnSCE bit to 1 in the initial setting.

2. If the next transfer is reception only, dummy data is written to the CBnTX register.

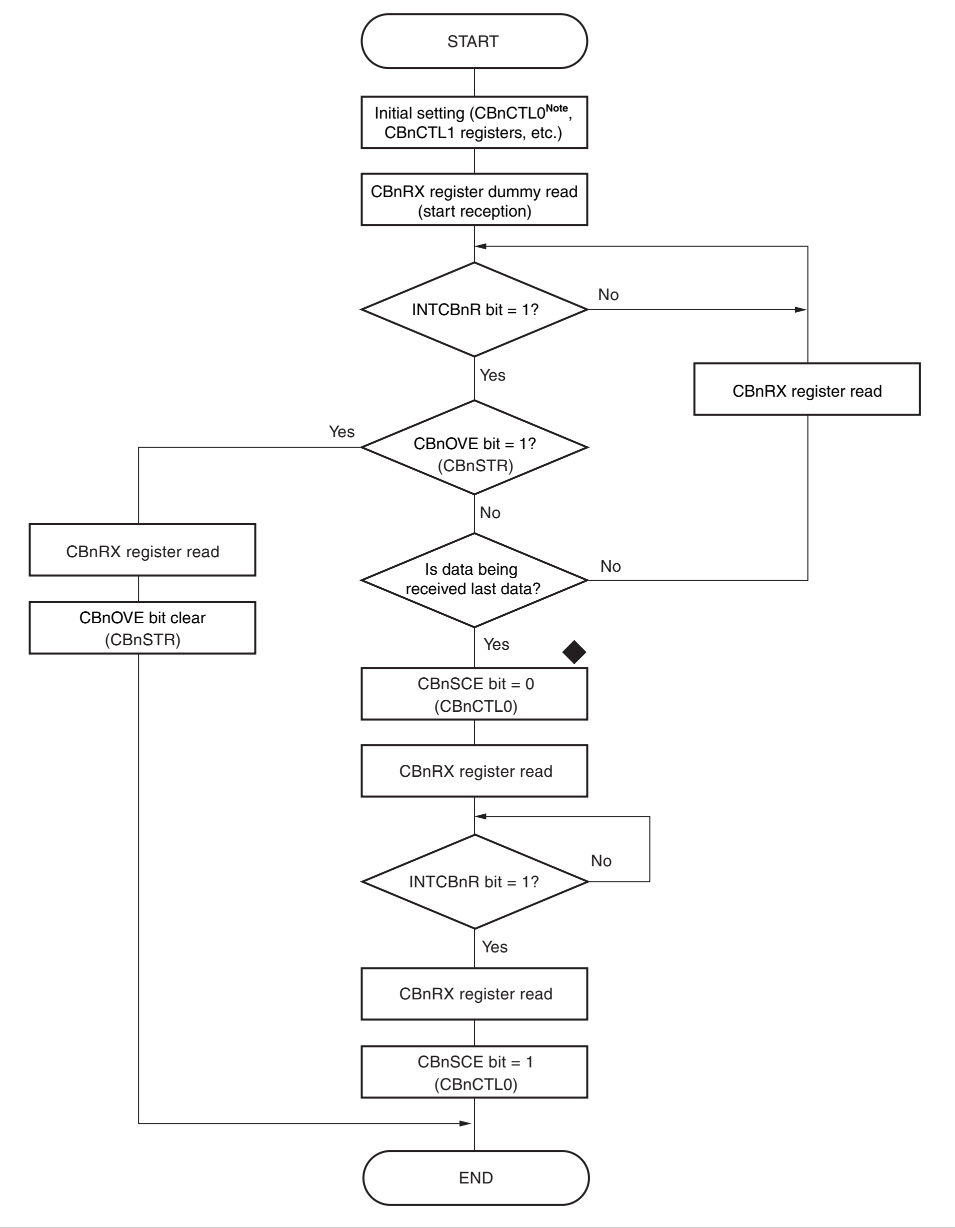
Caution Even in the single mode, the CBnSTR.CBnOVE flag is set to 1. If only transmission is used in the transmission/reception mode, therefore, programming without checking the CBnOVE flag is recommended.

(4) Continuous transmission



Note Set the CBnSCE bit to 1 in the initial setting.

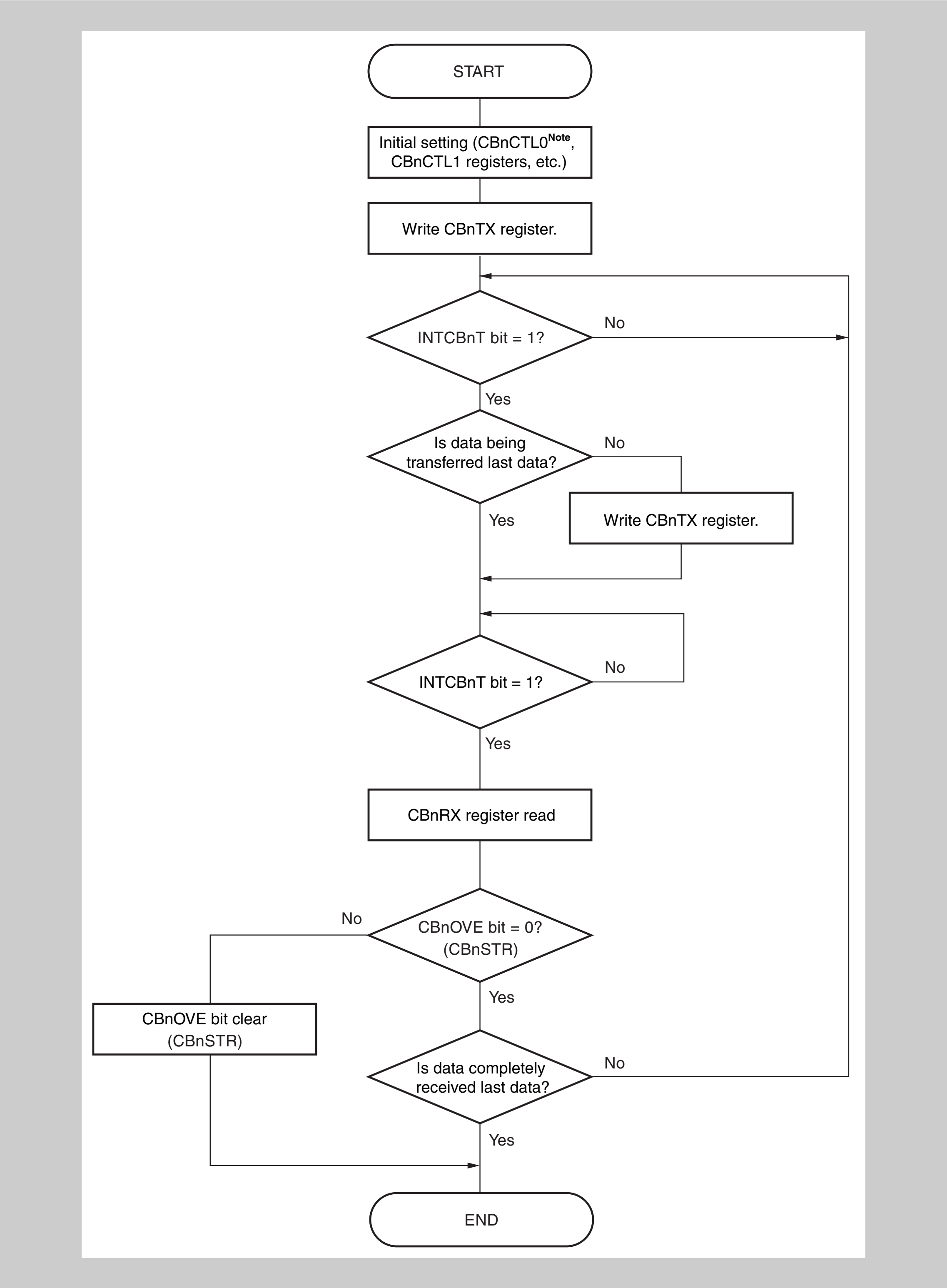
(5) Continuous reception



Note Set the CBnSCE bit to 1 in the initial setting.

Caution In the master mode, the clock is output without limit when dummy data is read from the CBnRX register. To stop the clock, execute the flow marked ◆ in the above flowchart.
In the slave mode, malfunction due to noise during communication can be prevented by executing the flow marked ◆ in the above flowchart.
Before resuming communication, set the CBnCTL0.CBnSCE bit to 1, and read dummy data from the CBnRX register.

(6) Continuous transmission/reception

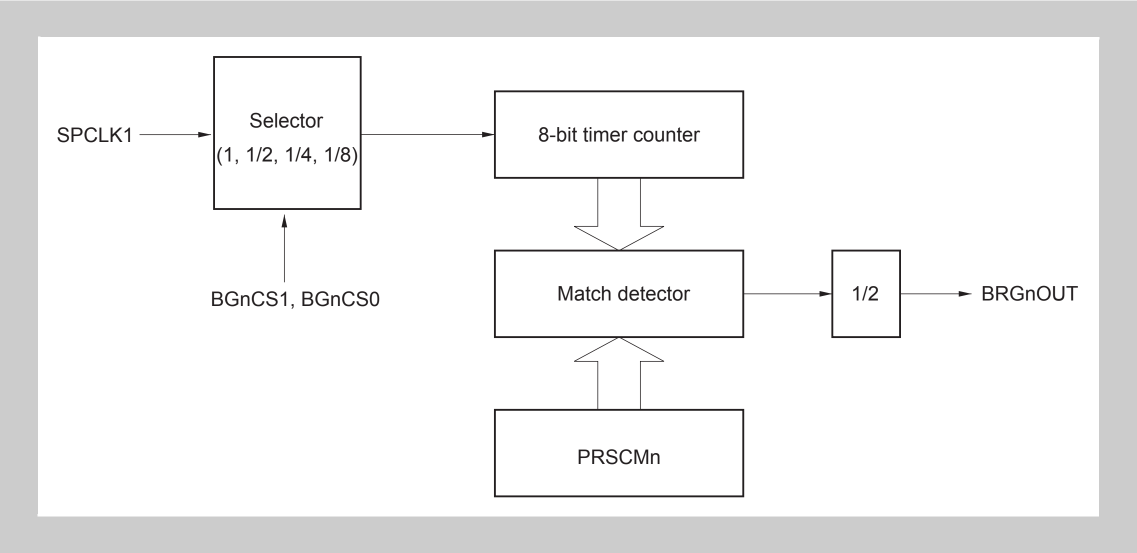


Note Set the CBnSCE bit to 1 in the initial setting.

17.7 Baud Rate Generator

17.7.1 Overview

Each CSIBSn interface is equipped with a dedicated baud rate generator.



17.7.2 Baud Rate Generator registers

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

Table 17-9 BRGn registers overview

Register name	Shortcut	Address
BRGn prescaler mode register	PRSMn	<BRG_base>
BRGn prescaler compare register	PRSCMn	<BRG_base> + 1_H

Table 17-10 BRGn register base address

Timer	Base address <BRG_base>
BRG0	FFFF $FDC0_H$
BRG1	FFFF $FDE0_H$
BRG2	FFFF $FDF0_H$

(1) PRSMn - Prescaler mode registers

The PRSMn registers control generation of the baud rate signal for CSIB.

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

Address <BRG_base>

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

7	6	5	4	3	2	1	0
0	0	0	BGCEn	0	0	BGCSn1	BGCSn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 17-11 PRSMn register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>4</td><td>BGCEn</td><td>Baud rate output
0: disabled
1: enabled</td></tr>
<tr><td>1 to 0</td><td>BGCSn[1:0]</td><td>Input clock selection
<table>
<tr><th>BGCSn1</th><th>BGCSn0</th><th>Input clock selection (f_{BGCSn})</th><th>Setting value k</th></tr>
<tr><td>0</td><td>0</td><td>f_{SPCLK1}</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>$f_{SPCLK1}/2$</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>$f_{SPCLK1}/4$</td><td>2</td></tr>
<tr><td>1</td><td>1</td><td>$f_{SPCLK1}/8$</td><td>3</td></tr>
</table>
</td></tr>
</table>

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

(2) PRSCMn - Prescaler compare registers

The PRSCMn registers are 8-bit compare registers.

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

Address <BRG_base> + 1_H

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

7	6	5	4	3	2	1	0
PRSCMn7	PRSCMn6	PRSCMn5	PRSCMn4	PRSCMn3	PRSCMn2	PRSCMn1	PRSCMn0
R/W							

Caution

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

17.7.3 Baud rate calculation

The transmission/reception clock is generated by dividing the main clock. The baud rate generated from the main clock is obtained by the following equation.

$$f_{BRGn} = \frac{f_{SPCLK1}}{2^k \times N \times 2}$$

Note

f_{BRGn}:	BRGn count clock
f_{SPCLK1}:	Main clock oscillation frequency
k:	PRSMn.BGCSn[1:0] register setting value ($0 \leq k \leq 3$)
N:	PRSCMn.PRSCMn[7:0] register value if PRSCMn = 00H: N = 256.

Chapter 18 I^{2}C Bus (IIC)

The V850E/Dx3 microcontrollers have following instances of the I^{2}C Bus interface IIC:

IIC	**All devices**
Instances	2
Names	IIC0 to IIC1

Throughout this chapter, the individual instances of ^{2}C Bus interface are identified by “n”, for example IICn, or IICCn for the IICn control register.

18.1 Features

The I^{2}C provides a synchronous serial interface with the following features:

- Supports Master and Slave mode
- 8-bit data transfer
- Transfer speed
  - up to 100 kbit/s (Standard Mode)
  - up to 400 kbit/s (Fast Mode)
- I^{2}C root clock sources from main oscillator, PLL and SSCG
- Two wire interface
  - SCLn: serial clock
  - SDAn: serial data
- Noise filter on SCLn and SDAn input
  - spikes with a width of less than one period of IICLK are suppressed
- IICn interrupts can be used for triggering the DMA Controller

18.2 I^2C Pin Configuration

The I^2C function requires to define the pins SCLn and SDAn as input and open drain output pins simultaneously. In the following the pin configuration registers are listed to be set up properly for I^2C:

- PFSR0.PFSR04/5 = 1/0: select input for I^2Cn (where applicable)
- PLCDCn.PLCDCnm = 0: no LCD output (where applicable)
- PFCn.PFCnm = 1/0: select ALT1-/ALT2-OUT (where applicable)
- PMCn.PMCnm = 1: alternative mode
- PICCn.PICCnm = 0: non-Schmitt Trigger input
- PDSCn.PDSCnm = 1: drive strength control Limit2
- PODCn.PODCnm = 1: open drain output
- PMn.PMnm = 1: input mode

It is recommended to set the output mode as the last step.

Table 17-3 shows how to set up the registers for activating I^2C0 and I^2C1 from different pin groups.

Table 18-1 I^2C interface pins set up

I^2Cn	PFSR0 register	Pins and pin group	Register settings
I^2C0	PFSR0.PFSR04 = 0	SDA0/SCL0 via P16/P17	PMC1.PMC1[7:6] = 11_B PICC1.PICC1[7:6] = 00_B PDSC1.PDSC1[7:6] = 11_B PODC1.PODC1[7:6] = 11_B PM1.PM1[7:6] = 11_B
	PFSR0.PFSR04 = 1	SCL0/SDA0 via P64/P65	PLCDC6.PLCDC6[5:4] = 00_B PFC6.PFC6[5:4] = 00_B PMC6.PMC65 = 1_B PICC6.PICC6[5:4] = 00_B PDSC6.PDSC6[5:4] = 11_B PODC6.PODC6[5:4] = 11_B PM6.PM6[5:4] = 11_B
I^2C1	PFSR0.PFSR05 = 0	SDA1/SCL1 via P20/P21	PLCDC2.PLCDC2[1:0] = 00_B PMC2.PMC2[1:0] = 11_B PICC2.PICC2[1:0] = 00_B PDSC2.PDSC2[1:0] = 11_B PODC2.PODC2[1:0] = 11_B PM2.PM2[1:0] = 11_B
	PFSR0.PFSR05 = 1	SDA1/SCL1 via P30/P31	PFC3.PFC30 = 1_B PMC3.PMC3[1:0] = 11_B PICC3.PICC3[1:0] = 00_B PDSC3.PDSC3[1:0] = 11_B PODC3.PODC3[1:0] = 11_B PM3.PM3[1:0] = 11_B

18.3 Configuration

The block diagram of the I^2C0n is shown below.

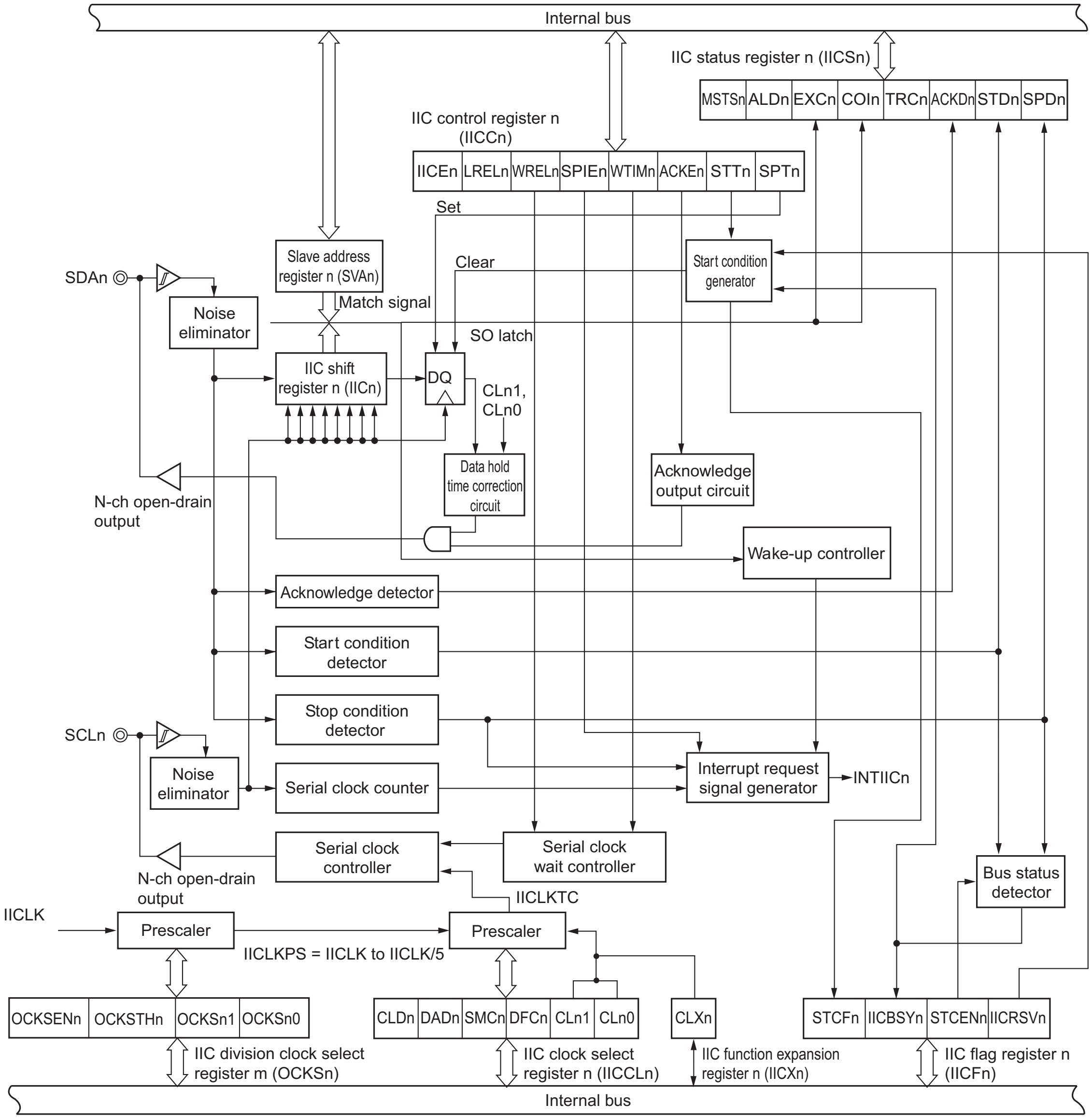


Figure 18-1 Block diagram of I^2C0n

A serial bus configuration example is shown below.

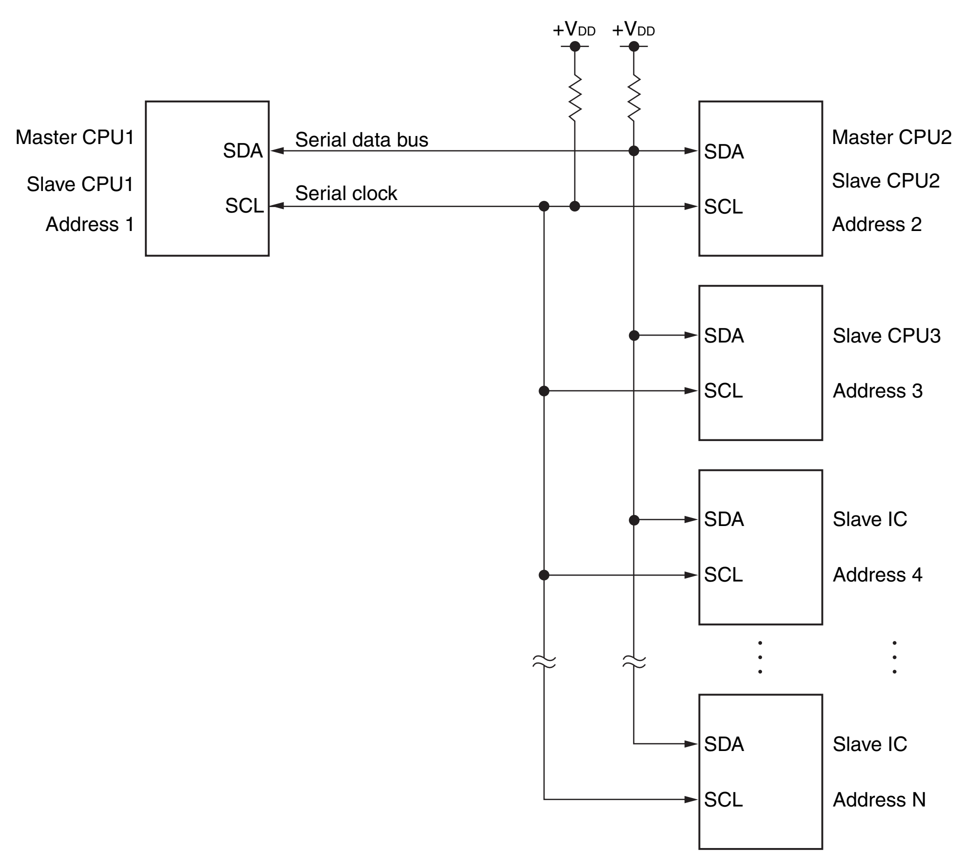


Figure 18-2 Serial bus configuration example using I^2C bus

I^2C0n includes the following hardware.

(1) IIC shift register n (IICn)

The IICn register converts 8-bit serial data into 8-bit parallel data and vice versa, and can be used for both transmission and reception.

Write and read operations to the IICn register are used to control the actual transmit and receive operations.

(2) Slave address register n (SVAn)

The SVAn register sets local addresses when in slave mode.

(3) SO latch

The SO latch is used to retain the output level of the SDAn pin.

(4) Wakeup controller

This circuit generates an interrupt request when the address received by this register matches the address value set to the SVAn register or when an extension code is received.

(5) Prescaler

This selects the sampling clock to be used.

(6) Serial clock counter

This counter counts the serial clocks that are output and the serial clocks that are input during transmit/receive operations and is used to verify that 8-bit data was transmitted or received.

(7) Interrupt request signal generator

This circuit controls the generation of interrupt request signals (INTIICn).

An I^2C interrupt is generated following either of two triggers:

- Falling edge of eighth or ninth clock of the serial clock (set by IICCn.WTIMn bit)
- Interrupt occurrence due to stop condition detection (set by IICCn.SPIEn bit)

(8) Serial clock controller

In master mode, this circuit generates the clock output via the SCLn pin from the sampling clock.

(9) Serial clock wait controller

This circuit controls the wait timing.

(10) $\overline{\text{ACK}}$ output circuit, stop condition detector, start condition detector, and $\overline{\text{ACK}}$ detector

These circuits are used to output and detect various control signals.

(11) Data hold time correction circuit

This circuit generates the hold time for data corresponding to the falling edge of the SCLn pin.

(12) Start condition generator

A start condition is issued when the IICCn.STTn bit is set.

However, in the communication reservation disabled status (IICFn.IICRSVn = 1), this request is ignored and the IICFn.STCFn bit is set if the bus is not released (IICFn.IICBSYn = 1).

(13) Bus status detector

Whether the bus is released or not is ascertained by detecting a start condition and stop condition.

However, the bus status cannot be detected immediately after operation, so set the bus status detector to the initial status by using the IICFn.STCENn bit.

18.4 IIC Registers

The I^2C serial interfaces IICn are controlled and operated by means of the following registers:

Table 18-2 IICn registers overview

Register name	Shortcut	Address
IICn shift register	IICn	<base>
IICn control register	IICCn	<base> + 2_H
IICn slave address register	SVAn	<base> + 3_H
IICn clock select register	IICCLn	<base> + 4_H
IICn function expansion register	IICXn	<base> + 5_H
IICn status register	IICSn	<base> + 6_H
IICn flag register	IICF0n	<base> + A_H
IICn division clock select registers	OCKSn	<base> + 20_H

Table 18-3 IICn register base address

IICn	Base address <base>
IIC0	FFFF $FD80_H$
IIC1	FFFF $FD90_H$

(1) IICCn - IICn control registers

The IICCn registers enable/stop I^{2}Cn operations, set the wait timing, and set other I^{2}Cn operations.

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

Address <base> + 2_H

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

7	6	5	4	3	2	1	0
IICEn	LRELn	WRELn	SPIEn	WTIMn	ACKEn	STTn	SPTn
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

IICEn	Specification of I^{2}Cn operation enable/disable	
0	Operation stopped. IICSn register reset[Note 1]. Internal operation stopped.	
1	Operation enabled.	
Condition for clearing (IICEn = 0)		**Condition for setting (IICEn = 1)**
• Cleared by instruction • After reset		• Set by instruction

LRELn	Exit from communications	
0	Normal operation	
1	This exits from the current communication operation and sets stand-by mode. This setting is automatically cleared after being executed. Its uses include cases in which a locally irrelevant extension code has been received. The SCLn and SDAn lines are set to high impedance. The STTn and SPTn bits and the MSTSn, EXCn, COIn, TRCn, ACKDn, and STDn bits of the IICSn register are cleared.	
The stand-by mode following exit from communications remains in effect until the following communication entry conditions are met. • After a stop condition is detected, restart is in master mode. • An address match occurs or an extension code is received after the start condition.		
Condition for clearing (LRELn = 0)[Note 2]		**Condition for setting (LRELn = 1)**
• Automatically cleared after execution • After reset		• Set by instruction

WRELn	Wait cancellation control	
0	Wait not cancelled	
1	Wait cancelled. This setting is automatically cleared after wait is cancelled.	
Condition for clearing (WRELn = 0)[Note 2]		**Condition for setting (WRELn = 1)**
• Automatically cleared after execution • After reset		• Set by instruction

<table>
<tr><th>SPIEn</th><th colspan="2">Enable/disable generation of interrupt request when stop condition is detected</th></tr>
<tr><td>0</td><td colspan="2">Disabled</td></tr>
<tr><td>1</td><td colspan="2">Enabled</td></tr>
<tr><th colspan="2">Condition for clearing (SPIEn = 0)[Note 2]</th><th>Condition for setting (SPIEn = 1)</th></tr>
<tr><td colspan="2">• Cleared by instruction
• After reset</td><td>• Set by instruction</td></tr>
</table>

Note
1. The IICS register, IICFn.STCFn and IICFn.IICBSYn bits, and IICCLn.CLDn and IICCLn.DADn bits are reset.
2. This flag's signal is invalid when the IICEn = 0.

<table>
<tr><th>WTIMn</th><th colspan="2">Control of wait and interrupt request generation</th></tr>
<tr><td>0</td><td colspan="2">Interrupt request is generated at the eighth clock's falling edge.
Master mode: After output of eight clocks, clock output is set to low level and wait is set.
Slave mode: After input of eight clocks, the clock is set to low level and wait is set for the master device.
In order to generate the ninth clock on SCLn the wait status must be cancelled by writing to IICn or setting IICCn.WRELn = 1. Consequently the ninth clock will be delayed until the wait status is cancelled.</td></tr>
<tr><td>1</td><td colspan="2">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.</td></tr>
<tr><td colspan="3">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.</td></tr>
<tr><th colspan="2">Condition for clearing (WTIMn = 0)[Note]</th><th>Condition for setting (WTIMn = 1)</th></tr>
<tr><td colspan="2">• Cleared by instruction
• After reset</td><td>• Set by instruction</td></tr>
</table>

<table>
<tr><th>ACKEn</th><th colspan="2">Acknowledgement control</th></tr>
<tr><td>0</td><td colspan="2">Acknowledgment disabled.</td></tr>
<tr><td>1</td><td colspan="2">Acknowledgment enabled. During the ninth clock period, the SDAn line is set to low level.
However, $\overline{\text{ACK}}$ is invalid in other than extension mode during address transfers.</td></tr>
<tr><th colspan="2">Condition for clearing (ACKEn = 0)[Note]</th><th>Condition for setting (ACKEn = 1)[Note]</th></tr>
<tr><td colspan="2">• Cleared by instruction
• After reset</td><td>• Set by instruction</td></tr>
</table>

Note This flag's signal is invalid when the IICEn = 0.

<table>
<tr><th>STTn</th><th>Start condition trigger</th></tr>
<tr><td>0</td><td>Start condition is not generated.</td></tr>
<tr><td>1</td><td>When bus is released (in STOP mode):
A start condition is generated (for starting as master). The SDAn line is changed from high level to low level and then the start condition is generated. Next, after the rated amount of time has elapsed, the SCLn line is changed to low level.
During communication with a third party:
If the communication reservation function is enabled (IICFn.IICRSVn = 0)
• This trigger functions as a start condition reserve flag. When set, it releases the bus and then automatically generates a start condition.
If the communication reservation function is disabled (IICRSVn = 1)
• The IICFn.STCFn bit is set. This trigger does not generate a start condition.
In the wait state (when master device):
A restart condition is generated after the wait is released.</td></tr>
<tr><td colspan="2">Cautions concerning set timing
For master reception: Cannot be set during transfer. Can be set only when the ACKEn bit has been set to 0 and the slave has been notified of final reception.
For master transmission: A start condition cannot be generated normally during the $\overline{\text{ACK}}$ period. Set during the wait period.
For slave: Even when the communication reservation function is disabled (IICRSVn bit = 1), the communication reservation status is entered.</td></tr>
<tr><th>Condition for clearing (STTn = 0)^{Note}</th><th>Condition for setting (STTn = 1)</th></tr>
<tr><td>• Cleared by loss in arbitration
• Cleared after start condition is generated by master device
• When the LRELn = 1 (communication save)
• When the IICEn= 0 (operation stop)
• After reset</td><td>• Set by instruction</td></tr>
</table>

Note **1.** Clearing the IICEn bit to 0 invalidates the signals of this flag.

2. The STTn bit is 0 if it is read immediately after data setting.

SPTn	Stop condition trigger
0	Stop condition is not generated.
1	Stop condition is generated (termination of master device's transfer). After the SDAn line goes to low level, either set the SCLn line to high level or wait until it goes to high level. Next, after the rated amount of time has elapsed, the SDAn line is changed from low level to high level and a stop condition is generated.

Cautions concerning set timing

For master reception:	Cannot be set during transfer. Can be set only when the ACKEn bit has been set to 0 and during the wait period after the slave has been notified of final reception.
For master transmission:	A stop condition cannot be generated normally during the $\overline{\text{ACK}}$ period. Set during the wait period.

- SPTn cannot be set at the same time as the STTn bit.
- The SPTn bit can be set only when in master mode[Note 1].
- When the WTIMn bit has been set to 0 and the SPTn bit is set during the wait period that follows output of eight clocks, note that a stop condition will be generated during the high-level period of the ninth clock. When the ninth clock must be output to apply the $\overline{\text{ACK}}$ on the bus by the receiving device, proceed as follows:
  - Change IICCn.WTIMn from 0 to 1 in order to receive an additional interrupt after the ninth clock.
  - Cancel the wait state by IICCn.WRELn = 1 or by writing to the IICn register.
  - Upon the interrupt after the ninth clock require to set the stop condition by IICCn.STPn = 1. By this the wait status will be cancelled and the stop condition will be generated on the bus.

Condition for clearing (SPTn = 0)[Note 2]	Condition for setting (SPTn = 1)
• Cleared by loss in arbitration • Automatically cleared after stop condition is detected • When the LRELn = 1 (communication save) • When the IICEn = 0 (operation stop) • After reset	• Set by instruction

Note

1. Set the SPTn bit only in master mode. However, when communication reservation is enabled (IICFn.IICRSVn = 0), the SPTn bit must be set and a stop condition generated before the first stop condition is detected following the switch to the operation enabled status. For details, see *"Cautions" on page 624*.
2. Clearing the IICEn bit to 0 invalidates the signals of this flag.
3. The SPTn bit is 0 if it is read immediately after data setting.

Caution When the TRCn = 1, the WRELn bit is set during the ninth clock and wait is canceled, after which the TRCn bit is cleared and the SDAn line is set to high impedance.

---

(2) IICSn - IICn status registers

The IICSn registers indicate the status of the I^{2}Cn bus.

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

Address <base> + 6$_H$

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

7	6	5	4	3	2	1	0
MSTSn	ALDn	EXCn	COIn	TRCn	ACKDn	STDn	SPDn
R	R	R	R	R	R	R	R

MSTSn	Master device status
0	Slave device status or communication stand-by status
1	Master device communication status

Condition for clearing (MSTSn = 0)	Condition for setting (MSTSn = 1)
• When a stop condition is detected • When the ALDn = 1 (arbitration loss) • Cleared by LRELn = 1 (communication save) • When the IICEn bit changes from 1 to 0 (operation stop) • After reset	• When a start condition is generated

ALDn	Arbitration loss detection
0	This status means either that there was no arbitration or that the arbitration result was a “win”.
1	This status indicates the arbitration result was a “loss”. The MSTSn bit is cleared.

Condition for clearing (ALDn = 0)	Condition for setting (ALDn = 1)
• Automatically cleared after the IICSn register is read[Note] • When the IICEn bit changes from 1 to 0 (operation stop) • After reset	• When the arbitration result is a “loss”.

Note Any bit manipulation instruction targetting this register also clears this bit.

EXCn	Detection of extension code reception
0	Extension code was not received.
1	Extension code was received.

Condition for clearing (EXCn = 0)	Condition for setting (EXCn = 1)
• When a start condition is detected • When a stop condition is detected • Cleared by LRELn = 1 (communication save) • When the IICEn bit changes from 1 to 0 (operation stop) • After reset	• When the higher four bits of the received address data are either “0000” or “1111” (set at the rising edge of the eighth clock).

COIn	Matching address detection
0	Addresses do not match.
1	Addresses match.

Condition for clearing (COIn = 0)	Condition for setting (COIn = 1)
• When a start condition is detected • When a stop condition is detected • Cleared by LRELn bit = 1 (communication save) • When the IICEn bit changes from 1 to (operation stop) • After reset	• When the received address matches the local address (SVAn register) (set at the rising edge of the eighth clock).

TRCn	Transmit/receive status detection
0	Receive status (other than transmit status). The SDAn line is set to high impedance.
1	Transmit status. The value in the SO latch is enabled for output to the SDAn line (valid starting at the falling edge of the first byte's ninth clock).

Condition for clearing (TRCn = 0)	Condition for setting (TRCn = 1)
• When a stop condition is detected • Cleared by LRELn = 1 (communication save) • When the IICEn bit changes from 1 to 0 (operation stop) • Cleared by WRELn = 1**Note** • When the ALDn bit changes from 0 to 1 (arbitration loss) • After reset Master • When "1" is output to the first byte's LSB (transfer direction specification bit) Slave • When a start condition is detected When not used for communication	Master • When a start condition is generated Slave • When "1" is input by the first byte's LSB (transfer direction specification bit)

ACKDn	ACK detection
0	ACK was not detected.
1	ACK was detected.

Condition for clearing (ACKDn = 0)	Condition for setting (ACKD = 1)
• When a stop condition is detected • At the rising edge of the next byte's first clock • Cleared by LRELn = 1 (communication save) • When the IICEn bit changes from 1 to 0 (operation stop) • After reset	• After the SDAn bit is set to low level at the rising edge of the SCLn pin's ninth clock

Note The TRCn bit is cleared and SDAn line becomes high impedance when the WRELn bit is set and the wait state is canceled at the ninth clock by TRCn = 1.

STDn	Start condition detection
0	Start condition was not detected.
1	Start condition was detected. This indicates that the address transfer period is in effect
Condition for clearing (STDn = 0)	**Condition for setting (STDn = 1)**
• When a stop condition is detected • At the rising edge of the next byte's first clock following address transfer • Cleared by LRELn = 1 (communication save) • When the IICEn bit changes from 1 to 0 (operation stop) • After reset	• When a start condition is detected

SPDn	Stop condition detection
0	Stop condition was not detected.
1	Stop condition was detected. The master device's communication is terminated and the bus is released.
Condition for clearing (SPDn = 0)	**Condition for setting (SPDn = 1)**
• At the rising edge of the address transfer byte's first clock following setting of this bit and detection of a start condition • When the IICEn bit changes from 1 to 0 (operation stop) • After reset	• When a stop condition is detected

(3) IICFn - IICn flag registers

The registers set the I^2Cn operation mode and indicate the I^2C bus status.

Access This register can be read/written in 8-bit or 1-bit units.
STCFn and IICBSYn bits are read-only.

Address <base> + A_H

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

7	6	5	4	3	2	1	0
STCFn	IICBSYn	0	0	0	0	STCENn	IICRSVn
R	R	R/W	R/W	R/W	R/W	R/W	R/W

IICRSVn enables/disables the communication reservation function.

The initial value of the IICBSYn bit is set by using the STCENn bit (see *“Cautions“ on page 624*).

The IICRSVn and STCENn bits can be written only when operation of I^2Cn is disabled (IICCn.IICEn = 0). After operation is enabled, IICFn can be read.

STCFn	STTn clear
0	Start condition issued
1	Start condition cannot be issued, STTn bit cleared
Condition for clearing (STCFn = 0)	**Condition for setting (STCFn = 1)**
• Cleared by IICCn.STTn = 1 • After reset	• When start condition is not issued and STTn flag is cleared during communication reservation is disabled (IICRSVn = 1).

IICBSYn	I^2Cn bus status
0	Bus released status
1	Bus communication status
Condition for clearing (IICBSYn = 0)	**Condition for setting (IICBSYn = 1)**
• When stop condition is detected • After reset	• When start condition is detected • By setting the IICCn.IICEn bit when the STCENn = 0

STCENn	Initial start enable trigger
0	Start conditions cannot be generated until a stop condition is detected following operation enable (IICEn bit = 1).
1	Start conditions can be generated even if a stop condition is not detected following operation enable (IICEn = 1).
Condition for clearing (STCENn = 0)	**Condition for setting (STCENn = 1)**
• When start condition is detected • After reset	• Setting by instruction

<table>
<tr><th>IICRSVn</th><th colspan="2">Communication reservation function disable bit</th></tr>
<tr><td>0</td><td colspan="2">Communication reservation enabled</td></tr>
<tr><td>1</td><td colspan="2">Communication reservation disabled</td></tr>
<tr><th colspan="2">Condition for clearing (IICRSVn = 0)</th><th>Condition for setting (IICRSVn = 1)</th></tr>
<tr><td colspan="2">• Clearing by instruction
• After reset</td><td>• Setting by instruction</td></tr>
</table>

Note Bits 6 and 7 are read-only bits.

Caution

1. Write the STCENn bit only when operation is stopped (IICEn = 0).
2. When the STCENn = 1, the bus released status (IICBSYn = 0) is recognized regardless of the actual bus status immediately after the I^2Cn bus operation is enabled. Therefore, to issue the first start condition (STTn = 1), it is necessary to confirm that the bus has been released, so as to not disturb other communications.
3. Write the IICRSVn bit only when operation is stopped (IICEn = 0).

(4) IICCLn - IICn clock select registers

The IICCLn registers set the transfer clock for the I^{2}Cn bus.

The SMCn, CLn1, and CLn0 bits are set by the combination of the IICXn.CLXn bit and the OCKSTHn, OCKSn[1:0] bits of the OCKSn register (see *"Transfer rate setting" on page 590*).

Access This register can be read/written in 8-bit or 1-bit units.
CLDn and DADn bits are read-only.

Address <base> + 4_H

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

7	6	5	4	3	2	1	0
0	0	CLDn	DADn	SMCn	DFCn	CLn1	CLn0
R/W	R/W	R	R	R/W	R/W	R/W	R/W

CLDn	Detection of SCLn pin level (valid only when IICCn.IICEn = 1)
0	The SCLn pin was detected at low level.
1	The SCLn pin was detected at high level.

Condition for clearing (CLDn = 0)	Condition for setting (CLDn = 1)
• When the SCLn pin is at low level • When the IICEn = 0 (operation stop) • After reset	• When the SCLn pin is at high level

DADn	Detection of SDAn pin level (valid only when IICEn = 1)
0	The SDAn pin was detected at low level.
1	The SDAn pin was detected at high level.

Condition for clearing (DADn = 0)	Condition for setting (DAD0n = 1)
• When the SDAn pin is at low level • When the IICEn = 0 (operation stop) • After reset	• When the SDAn pin is at high level

SMCn	Operation mode switching
0	Operation in standard mode.
1	Operation in fast-speed mode.

DFCn	Digital filter operation control
0	Digital filter off.
1	Digital filter on.
The digital filter can be used only in fast-speed mode. In fast-speed mode, the transfer clock does not vary regardless of the DFCn bit setting (on/off). The digital filter is used to eliminate noise in fast-speed mode.	

(5) IICXn - IICn function expansion registers

The IICXn registers provide additional transfer data rate configuration in fast-speed mode. Setting of the IICXn.CLXn is performed in combination with the IICCLn.SMCn, IICCLn.CLn[1:0], OCKSn.OCKSTHn and OCKSn.OCKSn[1:0] (refer to *“Transfer rate setting“ on page 590*)

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

Address <base> + 5_H

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

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

(6) OCKSn - IICn division clock select registers

The OCKSn registers control the I^2Cn division clock.

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

Address <base> + 20_H

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

7	6	5	4	3	2	1	0
0	0	0	1	OCKSTHn	0	OCKSn1	OCKSn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution Bit of OCKSn must be set to “1” after reset and must not be changed afterwards.

OCKSTHn	OCKSn1	OCKSn0	Output clock IICLKPS
0	0	0	IICLK/2 (Only allowed if 4MHz IICLK is used!)
0	0	1	IICLK/3
0	1	0	IICLK/4
0	1	1	IICLK/5
1	0	0	IICLK (Only allowed if 4MHz IICLK is used!)
Other than above			Setting prohibited

(7) Transfer rate setting

The nominal transfer rate of the I^2C interface is determined by the following means:

- the root clock source for the I^2C clock IICLK can be chosen as
  - main oscillator (4 MHz): ICC.IICSEL1 = 0
  - 32 MHz clock from the PLL: ICC.IICSEL1 = 1

The output clock IICLK supplies the IIC interface.

- The IICLK is divided by 1 to 5, configured by OCKSn.OCKSTHn and OCKSn.OCKSTn[1:0] (refer to *"OCKSn - IICn division clock select registers" on page 589*). 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 table:

Note The clock chosen as the input clock, that means IICLKPS, must lie in the range of 1 MHz to 10 MHz.

IICXn.CLXn	IICCLn.SMCn	IICCLn.CLn1	IICCLn.CLn0	Input clock	Transfer clock	Mode
0	0	0	0	$f_{IICLKPS}$	$f_{IICLKPS}/44$	standard
		0	1	$f_{IICLKPS}$	$f_{IICLKPS}/86$	standard
		1	0	n.a.	n.a.	n.a.
		1	1	$f_{IICLKPS}$	$f_{IICLKPS}/66$	standard
	1	0	0	$f_{IICLKPS}$	$f_{IICLKPS}/24$	fast-speed
		0	1	$f_{IICLKPS}$	$f_{IICLKPS}/24$	fast-speed
		1	0	n.a.	n.a.	n.a.
		1	1	$f_{IICLKPS}$	$f_{IICLKPS}/18$	fast-speed
1	0	x	x	n.a.	n.a.	n.a.
	1	0	0	$f_{IICLKPS}$	$f_{IICLKPS}/12$	fast-speed
		0	1	$f_{IICLKPS}$	$f_{IICLKPS}/12$	fast-speed
		1	0	n.a.	n.a.	n.a.
		1	1	$f_{IICLKPS}$	$f_{IICLKPS}/18$	fast-speed

Following table lists set-ups for some useful I^2C transfer clocks.

Prescaler		I^2C module set-up				Transfer clock [KHz]
OCKSn	divisor	IICCLn. SMCn	IICXn. CLXn	IICCLn. CLn[1:0]	divisor	
1 0011_B = 13_H	5	1	1	11_B	18	355,56
		0	0	11_B	66	96,97

Note The calculations in the above table assumes that IICLK is 32 MHz (IIC.IICSEL1 = 1)

Clock Stretching

Heavy capacitive load and the dimension of the external pull-up resistor on the I^2C bus pins may yield extended rise times of the rising edge of SCLn and SDAn. Since the controller senses the level of the I^2C bus signals it recognizes such situation and takes countermeasures by stretching the clock SCLn in order to ensure proper high level time t_{SCLH} of SCLn.

After the microcontoller releases the (open-drain) SCLn pin it waits until the SCLn level exceeds the valid high level threshold V_{thH}. Then it does not pull SCLn to low level before the nominal high level time t_{SCLH_nom} has elapsed.

This mechanism is the same used, when a slow I^2C slave device is pulling down SCLn to low level to initiate a wait state.

Figure 18-3 shows an example.

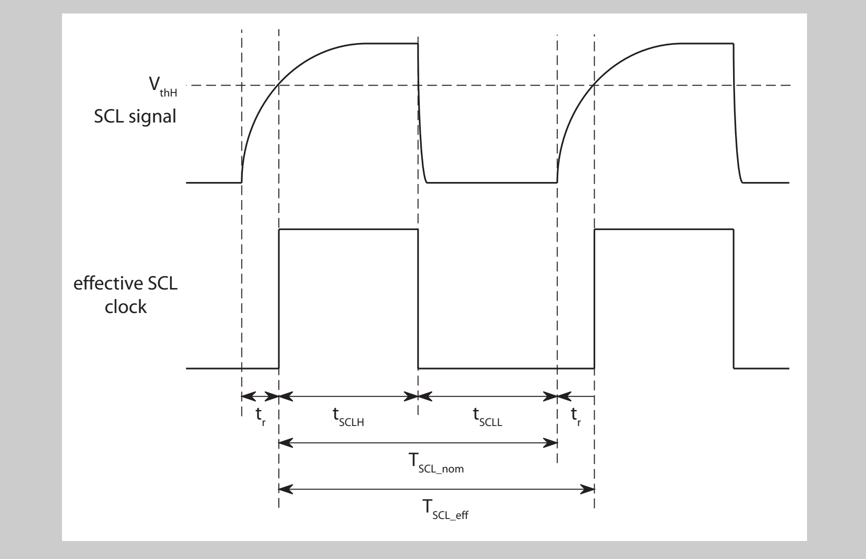


Figure 18-3 **Clock Stretching of SCLn**

The effective clock frequency appearing at the SCLn pin calculates to

$f_{SCL\_eff} = 1 / (T_{SCL\_nom} + t_r)$

With a nominal frequency of f_{SCL_nom} = 355 KHz (T_{SCL_nom} = 2.817 µs and a rise time of t_r = 135 ns the effective frequency is f_{eff} = 339 KHz.

(8) IICn - IICn shift registers

The IICn registers are used for serial transmission/reception (shift operations) synchronized with the serial clock.

A wait state is released by writing the IICn register during the wait period, and data transfer is started.

Access This register can be read/written in 8-bit units.
Data should not be written to the IICn register during a data transfer.

Address <base>

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

7	6	5	4	3	2	1	0
Input/output data							
R/W							

(9) SVAn - IICn slave address registers

The SVAn registers hold the I^{2}C bus's slave addresses.

Access This register can be read/written in 8-bit units.
Bit 0 should be fixed to 0.

Address <base> + 3_H

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

7	6	5	4	3	2	1	0
Slave address							
R/W							

18.5 I^2C Bus Pin Functions

The serial clock pin (SCLn) and serial data bus pin (SDAn) are configured as follows.

SCLn	This pin is used for serial clock input and output. This pin is an N-ch open-drain output for both master and slave devices. Input is Schmitt input.
SDAn	This pin is used for serial data input and output. This pin is an N-ch open-drain output for both master and slave devices. Input is Schmitt input.

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.

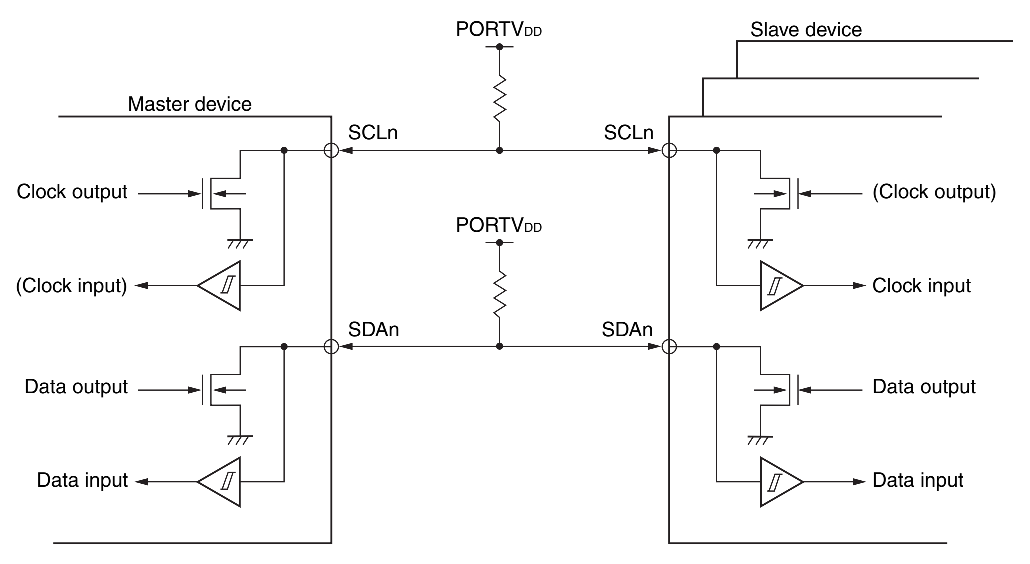


Figure 18-4 Pin configuration diagram

18.6 I^2C Bus Definitions and Control Methods

The following section describes the I^2C bus's serial data communication format and the signals used by the I^2C bus. The transfer timing for the “start condition", "data", and "stop condition" output via the I²C bus's serial data bus is shown below.

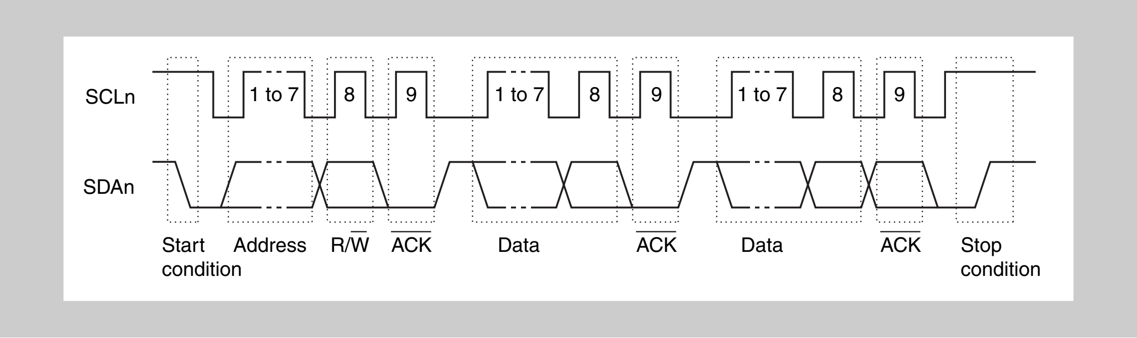


Figure 18-5 **I²C bus serial data transfer timing**

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 (SCLn) is continuously output by the master device. However, in the slave device, the SCLn pin's low-level period can be extended and a wait can be inserted.

18.6.1 Start condition

A start condition is met when the SCLn pin is high level and the SDAn pin changes from high level to low level. The start condition for the SCLn and SDAn pins is a signal that the master device outputs to the slave device when starting a serial transfer. The slave device can defect the start condition.

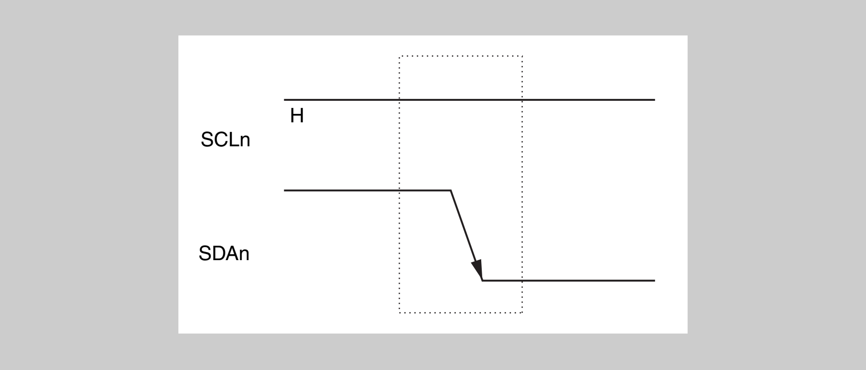


Figure 18-6 **Start condition**

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). By setting IICCN.STTn=1 the master device will also cancel its own wait status.

18.6.2 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.

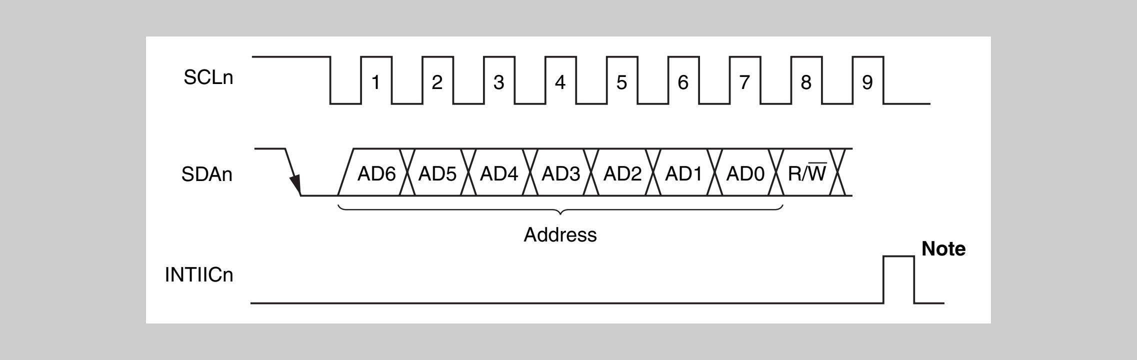


Figure 18-7 Address

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 596*, 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.

18.6.3 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.

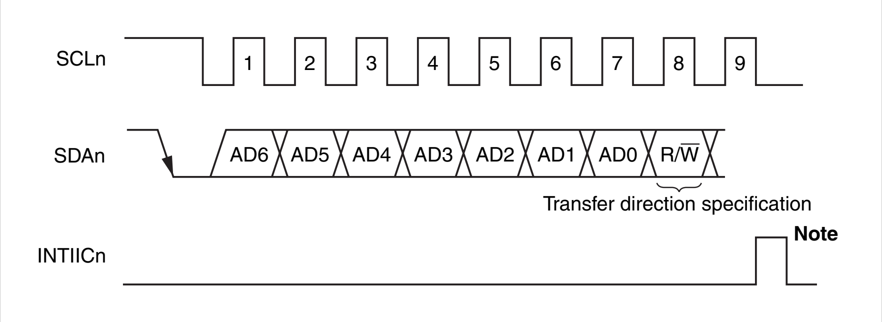


Figure 18-8 **Transfer direction specification**

Note The INTIICn signal is generated if a local address or extension code is received during slave device operation.

18.6.4 Acknowledge signal ($\overline{\text{ACK}}$)

The acknowledge signal ($\overline{\text{ACK}}$) is used by the transmitting and receiving devices to confirm serial data reception.

The receiving device returns one $\overline{\text{ACK}}$ signal for each 8 bits of data it receives. The transmitting device normally receives an $\overline{\text{ACK}}$ signal after transmitting 8 bits of data. However, when the master device is the receiving device, it does not output an $\overline{\text{ACK}}$ signal after receiving the final data to be transmitted. The transmitting device detects whether or not an $\overline{\text{ACK}}$ signal is returned after it transmits 8 bits of data. When an $\overline{\text{ACK}}$ signal is returned, the reception is judged as normal and processing continues. If the slave device does not return an $\overline{\text{ACK}}$ signal, the master device outputs either a stop condition or a restart condition and then stops the current transmission. Failure to return an $\overline{\text{ACK}}$ signal may be caused by the following two factors.

(a) Reception was not performed normally.

(b) The final data was received.

When the receiving device sets the SDAn line to low level during the ninth clock, the $\overline{\text{ACK}}$ signal becomes active (normal receive response).

When the IICCn.ACKEn bit is set to 1, automatic $\overline{\text{ACK}}$ signal generation is enabled.

Transmission of the eighth bit following the 7 address data bits causes the IICSn.TRCn bit to be set. When this TRCn bit's value is 0, it indicates receive mode. Therefore, the ACKEn bit should be set to 1.

When the slave device is receiving (when TRCn bit = 0), if the slave device does not need to receive any more data after receiving several bytes, clearing the ACKEn bit to 0 will prevent the master device from starting transmission of the subsequent data.

Similarly, when the master device is receiving (when TRCn bit = 0) and the subsequent data is not needed and when either a restart condition or a stop condition should therefore be output, clearing the ACKEn bit to 0 will prevent the $\overline{\text{ACK}}$ signal from being returned. This prevents the MSB from being output via the SDAn line (i.e., stops transmission) during transmission from the slave device.

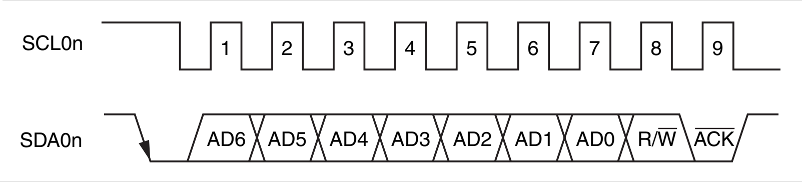


Figure 18-9 $\overline{\text{ACK}}$ **signal**

When the local address is received, an $\overline{\text{ACK}}$ signal is automatically output in synchronization with the falling edge of the SCLn pin's eighth clock regardless of the value of the ACKEn bit. No $\overline{\text{ACK}}$ signal is output if the received address is not a local address.

The $\overline{\text{ACK}}$ signal output method during data reception is based on the wait timing setting, as described below.

When 8-clock wait is selected (IICCn.WTIMn bit = 0):

> The $\overline{\text{ACK}}$ signal is output at the falling edge of the SCLn pin's eighth clock if the ACKEn bit is set to 1 before wait cancellation.

When 9-clock wait is selected (IICCn.WTIMn bit = 1):

> The $\overline{\text{ACK}}$ signal is automatically output at the falling edge of the SCLn pin's eighth clock if the ACKEn bit has already been set to 1.

18.6.5 Stop condition

When the SCLn pin is high level, changing the SDAn pin from low level to high level generates a stop condition.

A stop condition is a signal that the master device outputs to the slave device when serial transfer has been completed. When used as the slave device, the start condition can be detected.

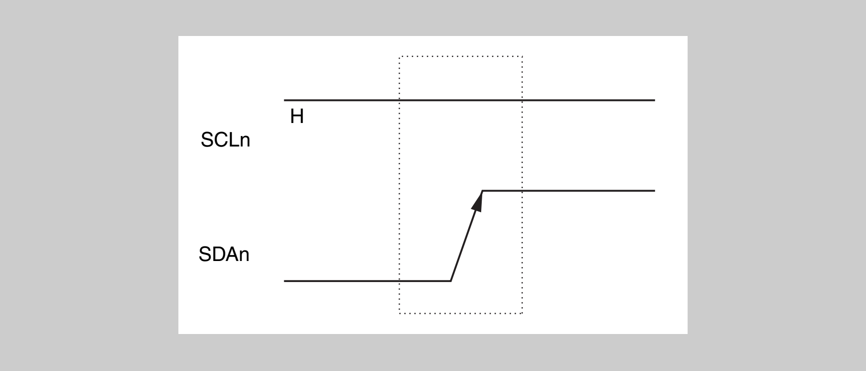


Figure 18-10 Stop condition

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. By setting IICCN.STPn=1 the master device will also cancel its own wait status.

18.6.6 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 SCLn pin to low level notifies the communication partner of the wait status. When the wait status has been cancelled for both the master and slave devices, the next data transfer can begin.

(1) When master device has a nine-clock wait and slave device has an eight-clock wait (master: transmission, slave: reception, and IICCn.ACKEn bit = 1)

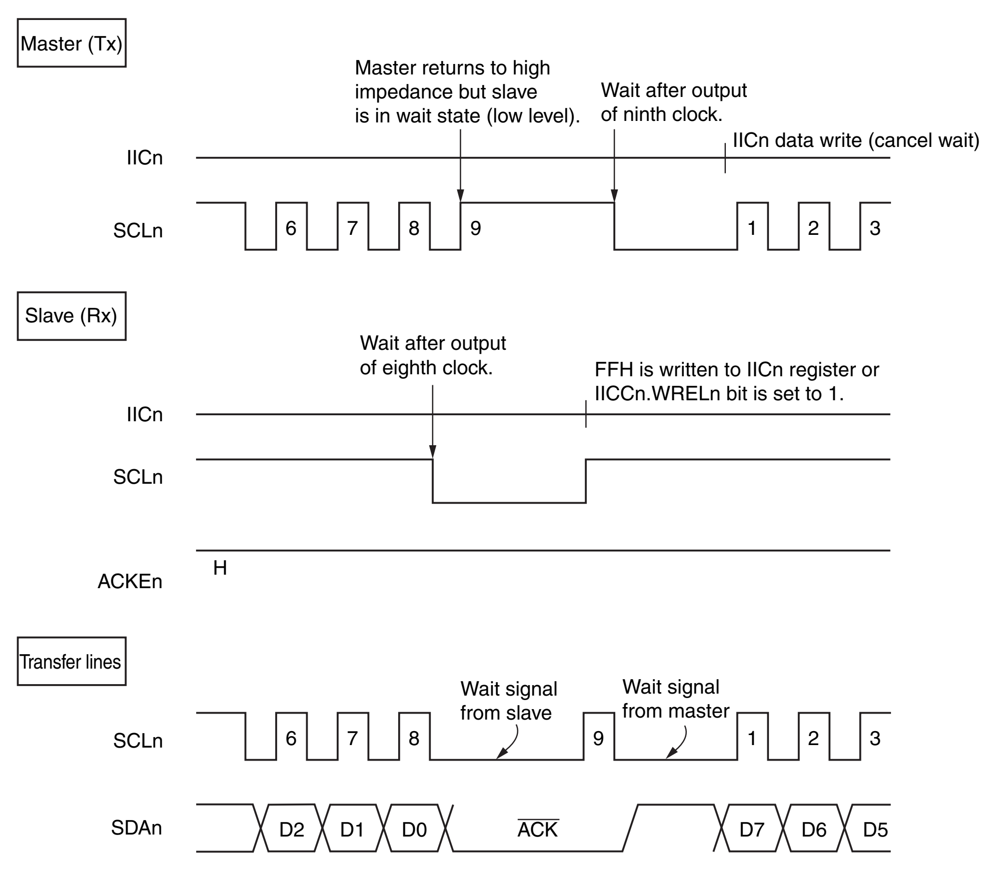


Figure 18-11 Wait signal (1/2)

(2) When master and slave devices both have a nine-clock wait (master: transmission, slave: reception, and ACKEn bit = 1)

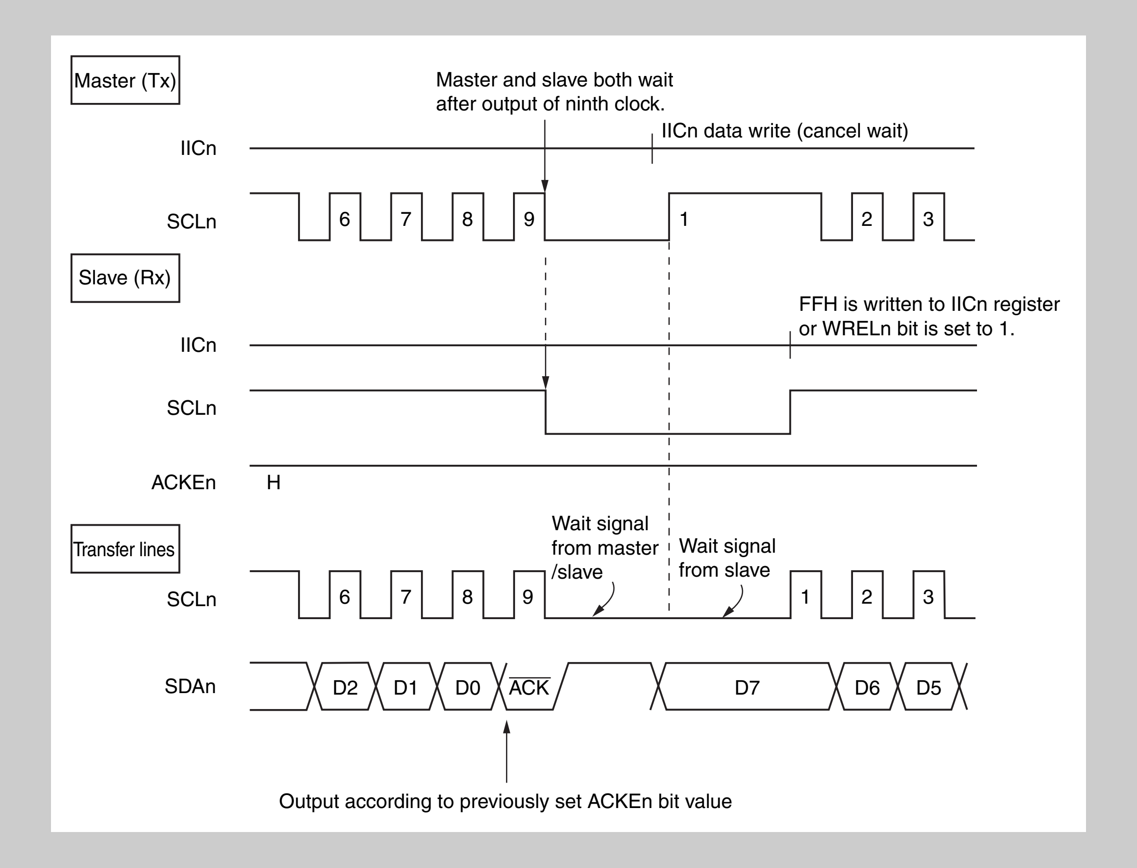


Figure 18-12 Wait signal (2/2)

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 its own wait status via either of the following methods.

- By setting the IICCn.STTn bit to 1
- By setting the IICCn.SPTn bit to 1

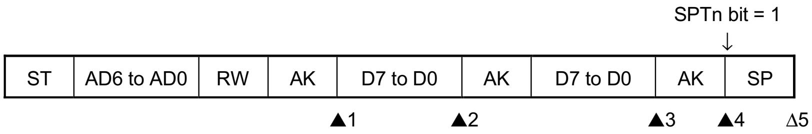
18.7 I^{2}C Interrupt Request Signals (INTIICn)

The following shows the value of the IICSn register at the INTIICn interrupt request signal generation timing and at the INTIICn signal timing.

18.7.1 Master device operation

(1) Start ~ Address ~ Data ~ Data ~ Stop (normal transmission/reception)

<1> When WTIMn bit = 0



▲1: IICSn register = 10XXX110B

▲2: IICSn register = 10XXX000B

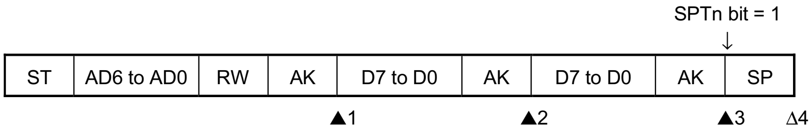
▲3: IICSn register = 10XXX000B (WTIMn bit = 1)

▲4: IICSn register = 10XXXX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0 to 2

<2> When WTIMn bit = 1



▲1: IICSn register = 10XXX110B

▲2: IICSn register = 10XXX100B

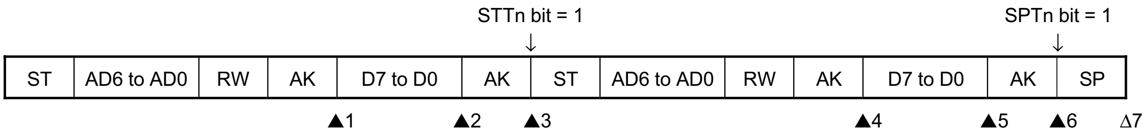
▲3: IICSn register = 10XXXX00B

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0 to 2

(2) Start ~ Address ~ Data ~ Start ~ Address ~ Data ~ Stop (restart)

<1> When WTIMn bit = 0



▲1: IICSn register = 10XXX110B

▲2: IICSn register = 10XXX000B (WTIMn bit = 1)

▲3: IICSn register = 10XXXX00B (WTIMn bit = 0)

▲4: IICSn register = 10XXX110B (WTIMn bit = 0)

▲5: IICSn register = 10XXX000B (WTIMn bit = 1)

▲6: IICSn register = 10XXXX00B

Δ 7: IICSn register = 00000001B

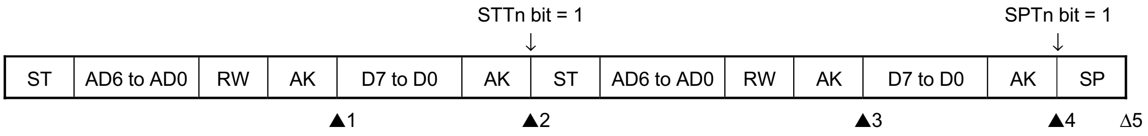
Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

<2> When WTIMn bit = 1



▲1: IICSn register = 10XXX110B

▲2: IICSn register = 10XXXX00B

▲3: IICSn register = 10XXX110B

▲4: IICSn register = 10XXXX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated

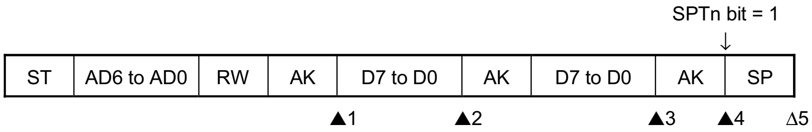
Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

(3) Start ~ Code ~ Data ~ Data ~ Stop (extension code transmission)

<1> When WTIMn bit = 0



▲1: IICSn register = 1010X110B

▲2: IICSn register = 1010X000B

▲3: IICSn register = 1010X000B (WTIMn bit = 1)

▲4: IICSn register = 1010XX00B

Δ 5: IICSn register = 00000001B

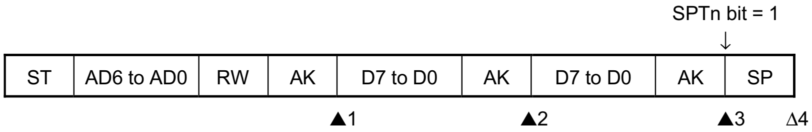
Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don't care

2. n = 0 to 2

<2> When WTIMn bit = 1



▲1: IICSn register = 1010X110B

▲2: IICSn register = 1010X100B

▲3: IICSn register = 1010XX00B

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

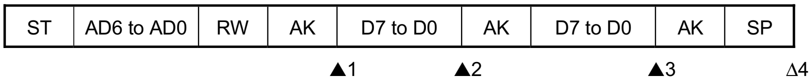
X: don't care

2. n = 0 to 2

18.7.2 Slave device operation

(1) Start ~ Address ~ Data ~ Data ~ Stop

<1> When WTIMn bit = 0



▲1: IICSn register = 0001X110B

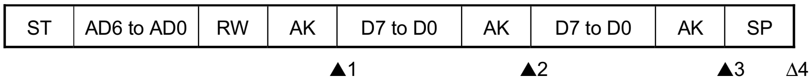
▲2: IICSn register = 0001X000B

▲3: IICSn register = 0001X000B

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0 to 2

<2> When WTIMn bit = 1



▲1: IICSn register = 0001X110B

▲2: IICSn register = 0001X100B

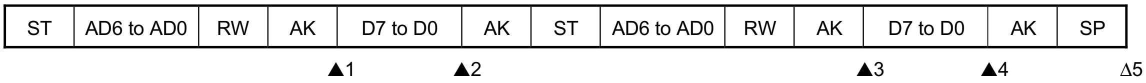
▲3: IICSn register = 0001XX00B

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0 to 2

(2) Start ~ Address ~ Data ~ Start ~ Address ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, address match)



▲1: IICSn register = 0001X110B

▲2: IICSn register = 0001X000B

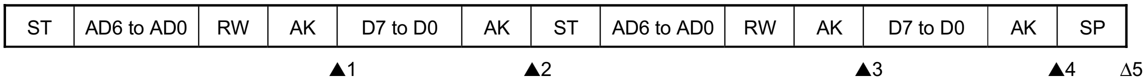
▲3: IICSn register = 0001X110B

▲4: IICSn register = 0001X000B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0 to 2

<2> When WTIMn bit = 1 (after restart, address match)



▲1: IICSn register = 0001X110B

▲2: IICSn register = 0001XX00B

▲3: IICSn register = 0001X110B

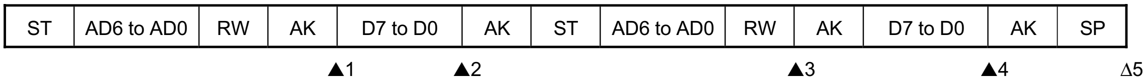
▲4: IICSn register = 0001XX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0 to 2

(3) Start ~ Address ~ Data ~ Start ~ Code ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, extension code reception)



▲1: IICSn register = 0001X110B

▲2: IICSn register = 0001X000B

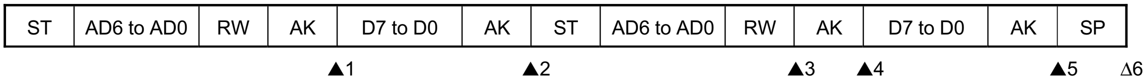
▲3: IICSn register = 0010X010B

▲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)



▲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	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	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 to 2

<2> When WTIMn bit = 1 (after restart, address mismatch (= not extension code))

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP
				▲1		▲2				▲3		Δ4

▲1: IICSn register = 0001X110B

▲2: IICSn register = 0001XX00B

▲3: IICSn register = 00000X10B

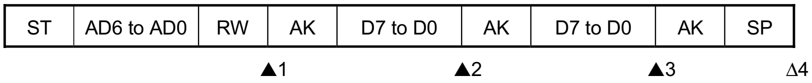
Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0 to 2

18.7.3 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

∆ 4: IICSn register = 00000001B

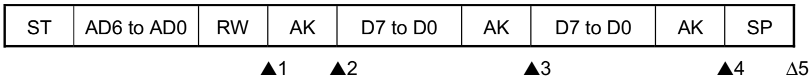
Remarks 1. ▲: Always generated

∆: Generated only when SPIEn bit = 1

X: don't care

2. n = 0 to 2

<2> When WTIMn bit = 1



▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X110B

▲3: IICSn register = 0010X100B

▲4: IICSn register = 0010XX00B

∆ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated

∆: Generated only when SPIEn bit = 1

X: don't care

2. n = 0 to 2

(2) Start ~ Code ~ Data ~ Start ~ Address ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, address match)

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP

▲1 ▲2 ▲3 ▲4 Δ5

▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X000B

▲3: IICSn register = 0001X110B

▲4: IICSn register = 0001X000B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

<2> When WTIMn bit = 1 (after restart, address match)

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP

▲1 ▲2 ▲3 ▲4 ▲5 Δ6

▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X110B

▲3: IICSn register = 0010XX00B

▲4: IICSn register = 0001X110B

▲5: IICSn register = 0001XX00B

Δ 6: IICSn register = 00000001B

Remarks 1. ▲: Always generated

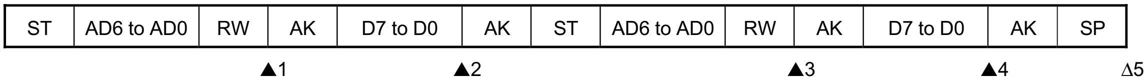
Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

(3) Start ~ Code ~ Data ~ Start ~ Code ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, extension code reception)



▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X000B

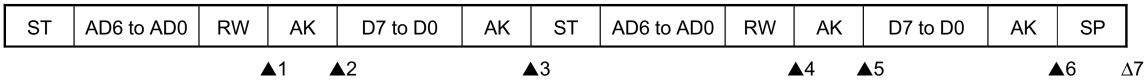
▲3: IICSn register = 0010X010B

▲4: IICSn register = 0010X000B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0 to 2

<2> When WTIMn bit = 1 (after restart, extension code reception)



▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X110B

▲3: IICSn register = 0010XX00B

▲4: IICSn register = 0010X010B

▲5: IICSn register = 0010X110B

▲6: IICSn register = 0010XX00B

Δ 7: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0 to 2

(4) Start ~ Code ~ Data ~ Start ~ Address ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, address mismatch (= not extension code))

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP

▲1 ▲2 ▲3 ∆4

▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X000B

▲3: IICSn register = 00000X10B

∆ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated

∆: Generated only when SPIEn bit = 1

X: don't care

2. n = 0 to 2

<2> When WTIMn bit = 1 (after restart, address mismatch (= not extension code))

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP

▲1 ▲2 ▲3 ▲4 ∆5

▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X110B

▲3: IICSn register = 0010XX00B

▲4: IICSn register = 00000X10B

∆ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated

∆: Generated only when SPIEn bit = 1

X: don't care

2. n = 0 to 2

18.7.4 Operation without communication

(1) Start ~ Code ~ Data ~ Data ~ Stop

ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	SP

Δ1

Δ 1: IICSn register = 00000001B

Remarks **1.** Δ: Generated only when SPIEn bit = 1

2. n = 0 to 2

18.7.5 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	RW	AK	D7 to D0	AK	D7 to D0	AK	SP

▲1 ▲2 ▲3 Δ4

▲1: IICSn register = 0101X110B (Example: When ALDn bit is read during interrupt servicing)

▲2: IICSn register = 0001X000B

▲3: IICSn register = 0001X000B

Δ 4: IICSn register = 00000001B

Remarks **1.** ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

<2> When WTIMn bit = 1

ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	SP

▲1 ▲2 ▲3 Δ4

▲1: IICSn register = 0101X110B (Example: When ALDn bit is read during interrupt servicing)

▲2: IICSn register = 0001X100B

▲3: IICSn register = 0001XX00B

Δ 4: IICSn register = 00000001B

Remarks **1.** ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

(2) When arbitration loss occurs during transmission of extension code

<1> When WTIMn bit = 0

ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	SP	
			▲1		▲2		▲3		Δ4

▲1: IICSn register = 0110X010B (Example: When ALDn bit is read during interrupt servicing)

▲2: IICSn register = 0010X000B

▲3: IICSn register = 0010X000B

Δ 4: IICSn register = 00000001B

Remarks **1.** ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

<2> When WTIMn bit = 1

ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	SP	
			▲1	▲2		▲3		▲4	Δ5

▲1: IICSn register = 0110X010B (Example: When ALDn bit is read during interrupt servicing)

▲2: IICSn register = 0010X110B

▲3: IICSn register = 0010X100B

▲4: IICSn register = 0010XX00B

Δ 5: IICSn register = 00000001B

Remarks **1.** ▲: Always generated

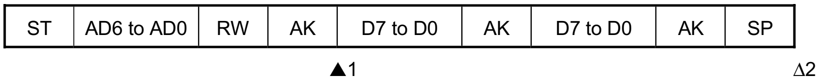
Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

18.7.6 Operation when arbitration loss occurs

(1) When arbitration loss occurs during transmission of slave address data



▲1: IICSn register = 01000110B (Example: When ALDn bit is read during interrupt servicing)

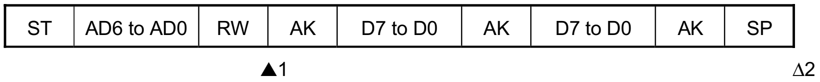
Δ 2: IICSn register = 00000001B

Remarks **1.** ▲: Always generated

Δ: Generated only when SPIEn bit = 1

2. n = 0 to 2

(2) When arbitration loss occurs during transmission of extension code



▲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

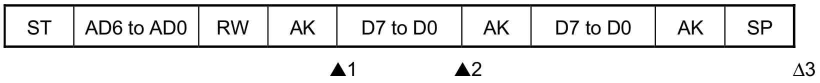
Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

(3) When arbitration loss occurs during data transfer

<1> When WTIMn bit = 0



▲1: IICSn register = 10001110B

▲2: IICSn register = 01000000B (Example: When ALDn bit is read during interrupt servicing)

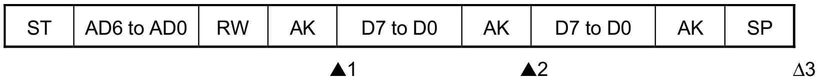
Δ 3: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

2. n = 0 to 2

<2> When WTIMn bit = 1



▲1: IICSn register = 10001110B

▲2: IICSn register = 01000100B (Example: When ALDn bit is read during interrupt servicing)

Δ 3: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

2. n = 0 to 2

(4) When arbitration loss occurs due to restart condition during data transfer

<1> Not extension code (Example: Address mismatch)

ST	AD6 to AD0	RW	AK	D7 to Dn	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP	
				▲1					▲2			Δ3

▲1: IICSn register = 1000X110B

▲2: IICSn register = 01000110B (Example: When ALDn bit is read during interrupt servicing)

Δ 3: IICSn register = 00000001B

Remarks **1.** ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. Dn = D6 to D0
n = 0 to 2

<2> Extension code

ST	AD6 to AD0	RW	AK	D7 to Dn	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP	
				▲1					▲2			Δ3

▲1: IICSn register = 1000X110B

▲2: IICSn register = 0110X010B (Example: When ALDn bit is read during interrupt servicing)

IICCn.LRELn bit is set to 1 by software

Δ 3: IICSn register = 00000001B

Remarks **1.** ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. Dn = D6 to D0
n = 0 to 2

(5) When arbitration loss occurs due to stop condition during data transfer

ST	AD6 to AD0	RW	AK	D7 to Dn	SP	
				▲1		Δ2

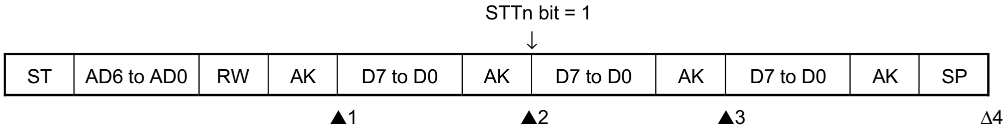
▲1: IICSn register = 1000X110B

Δ 2: IICSn register = 01000001B

Remarks **1.** ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. Dn = D6 to D0
n = 0 to 2

(6) When arbitration loss occurs due to low level of SDAn pin when attempting to generate a restart condition

When WTIMn bit = 1



▲1: IICSn register = 1000X110B

▲2: IICSn register = 1000XX00B

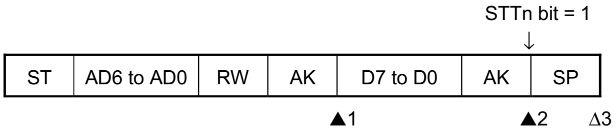
▲3: IICSn register = 01000100B (Example: When ALDn bit is read during interrupt servicing)

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0 to 2

(7) When arbitration loss occurs due to a stop condition when attempting to generate a restart condition

When WTIMn bit = 1



▲1: IICSn register = 1000X110B

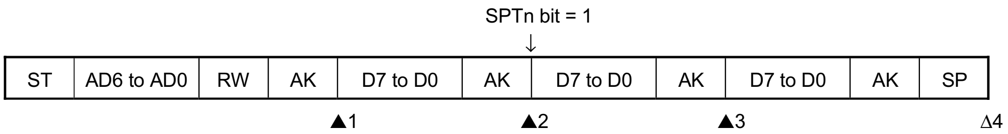
▲2: IICSn register = 1000XX00B

Δ 3: IICSn register = 01000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0 to 2

(8) When arbitration loss occurs due to low level of SDAn pin when attempting to generate a stop condition

When WTIMn bit = 1



▲1: IICSn register = 1000X110B

▲2: IICSn register = 1000XX00B

▲3: IICSn register = 01000000B (Example: When ALDn bit is read during interrupt servicing)

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

18.8 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 18-4 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)$^{\text{Note}}$
- By stop condition setting (IICCn.SPTn bit = 1)$^{\text{Note}}$

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.

18.9 Address Match Detection Method

In I^{2}C bus mode, the master device can select a particular slave device by transmitting the corresponding slave address.

Address match detection is performed automatically by hardware. The INTIICn signal occurs when a local address has been set to the SVAn register and when the address set to the SVAn register matches the slave address sent by the master device, or when an extension code has been received.

18.10 Error Detection

In I^{2}C bus mode, the status of the serial data bus pin (SDAn) during data transmission is captured by the IICn register of the transmitting device, so the data of the IICn register prior to transmission can be compared with the transmitted IICn data to enable detection of transmission errors. A transmission error is judged as having occurred when the compared data values do not match.

18.11 Extension Code

- When the higher 4 bits of the receive address are either 0000 or 1111, the extension code flag (IICSn.EXCn bit) is set for extension code reception and an interrupt request signal (INTIICn) is issued at the falling edge of the eighth clock.

  The local address stored in the SVAn register is not affected.

- If 11110xx0 is set to the SVAn register by a 10-bit address transfer and 11110xx0 is transferred from the master device, the results are as follows. Note that the INTIICn signal occurs at the falling edge of the eighth clock
  – Higher four bits of data match: EXCn bit = 1
  – Seven bits of data match: IICSn.COIn bit = 1

- Since the processing after the interrupt request signal occurs differs according to the data that follows the extension code, such processing is performed by software.

  For example, when operation as a slave is not desired after the extension code is received, set the IICCn.LRELn bit to 1 and the CPU will enter the next communication wait state.

Table 18-5 Extension code bit definitions

Slave Address	R/W Bit	Description
0000 000	0	General call address
0000 000	1	Start byte
0000 001	X	CBUS address
0000 010	X	Address that is reserved for different bus format
1111 0xx	X	10-bit slave address specification

18.12 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 SCLn and SDAn lines are both set to high impedance, which releases the bus.

Arbitration loss is detected based on the timing of the next interrupt request signal (the eighth or ninth clock, when a stop condition is detected, etc.) and the setting of the ALDn bit to 1, which is made by software.

For details of interrupt request timing, see*"I2C Interrupt Request Signals (INTIICn)" on page 601*.

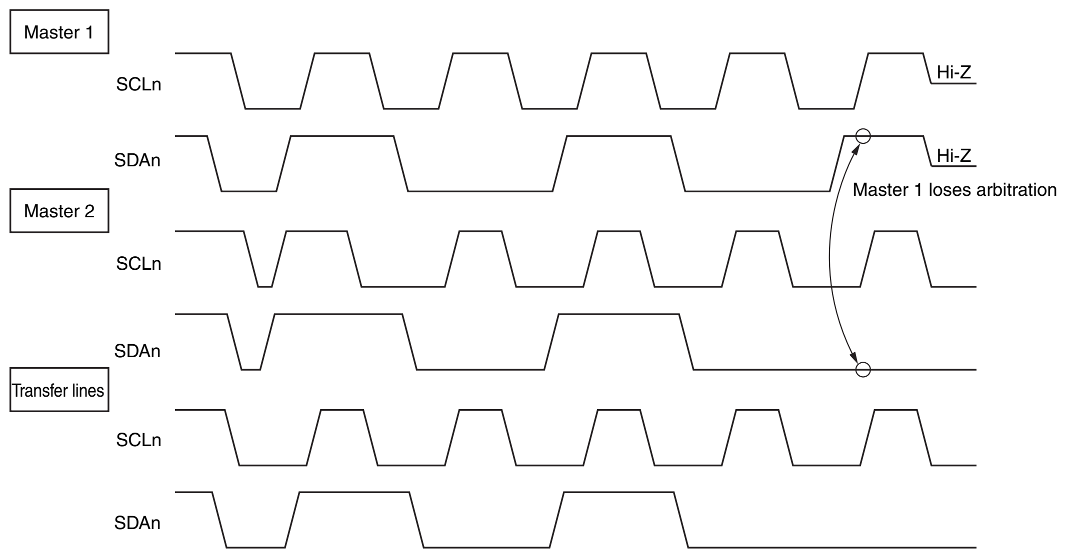


Figure 18-13 Arbitration timing example

Table 18-6 Status during arbitration and interrupt request signal generation timing

Status During Arbitration	Interrupt Request Generation Timing
Transmitting address transmission	At falling edge of eighth or ninth clock following byte transfer[Note 1]
Read/write data after address transmission	
Transmitting extension code	
Read/write data after extension code transmission	
Transmitting data	
$\overline{\text{ACK}}$ signal transfer period after data reception	
When restart condition is detected during data transfer	
When stop condition is detected during data transfer	When stop condition is output (when IICCn.SPIEn bit = 1)[Note 2]
When SDAn pin is low level while attempting to output restart condition	At falling edge of eighth or ninth clock following byte transfer[Note 1]
When stop condition is detected while attempting to output restart condition	When stop condition is output (when IICCn.SPIEn bit = 1)[Note 2]
When DSA0n pin is low level while attempting to output stop condition	At falling edge of eighth or ninth clock following byte transfer[Note 1]
When SCLn pin is low level while attempting to output restart condition	

Note

1. When the IICCn.WTIMn bit = 1, an interrupt request signal occurs at the falling edge of the ninth clock. When the WTIMn bit = 0 and the extension code's slave address is received, an interrupt request signal occurs at the falling edge of the eighth clock.
2. When there is a possibility that arbitration will occur, set the SPIEn bit to 1 for master device operation.

18.13 Wakeup Function

The I^2C bus slave function is a function that generates an interrupt request signal (INTIICn) when a local address and extension code have been received.

This function makes processing more efficient by preventing unnecessary interrupt request signals from occurring when addresses do not match.

When a start condition is detected, wakeup stand-by mode is set. This wakeup stand-by mode is in effect while addresses are transmitted due to the possibility that an arbitration loss may change the master device (which has output a start condition) to a slave device.

However, when a stop condition is detected, the IICCn.SPIEn bit is set regardless of the wakeup function, and this determines whether interrupt request signals are enabled or disabled.

18.14 Cautions

(1) When IICFn.STCENn bit = 0

Immediately after the I^{2}C0n operation is enabled, the bus communication status (IICFn.IICBSYn bit = 1) is recognized regardless of the actual bus status. To execute master communication in the status where a stop condition has not been detected, generate a stop condition and then release the bus before starting the master communication.

Use the following sequence for generating a stop condition.

<1> Set the IICCLn register.

<2> Set the IICCn.IICEn bit.

<3> Set the IICCn.SPTn bit.

(2) When IICFn.STCENn bit = 1

Immediately after I^{2}C0n operation is enabled, the bus released status (IICBSYn bit = 0) is recognized regardless of the actual bus status. To issue the first start condition (IICCn.STTn bit = 1), it is necessary to confirm that the bus has been released, so as to not disturb other communications.

18.15 Communication Operations

18.15.1 Master operation 1

The following shows the flowchart for master communication when the communication reservation function is enabled (IICFn.IICRSVn bit = 0) and the master operation is started after a stop condition is detected (IICFn.STCENn bit = 0).

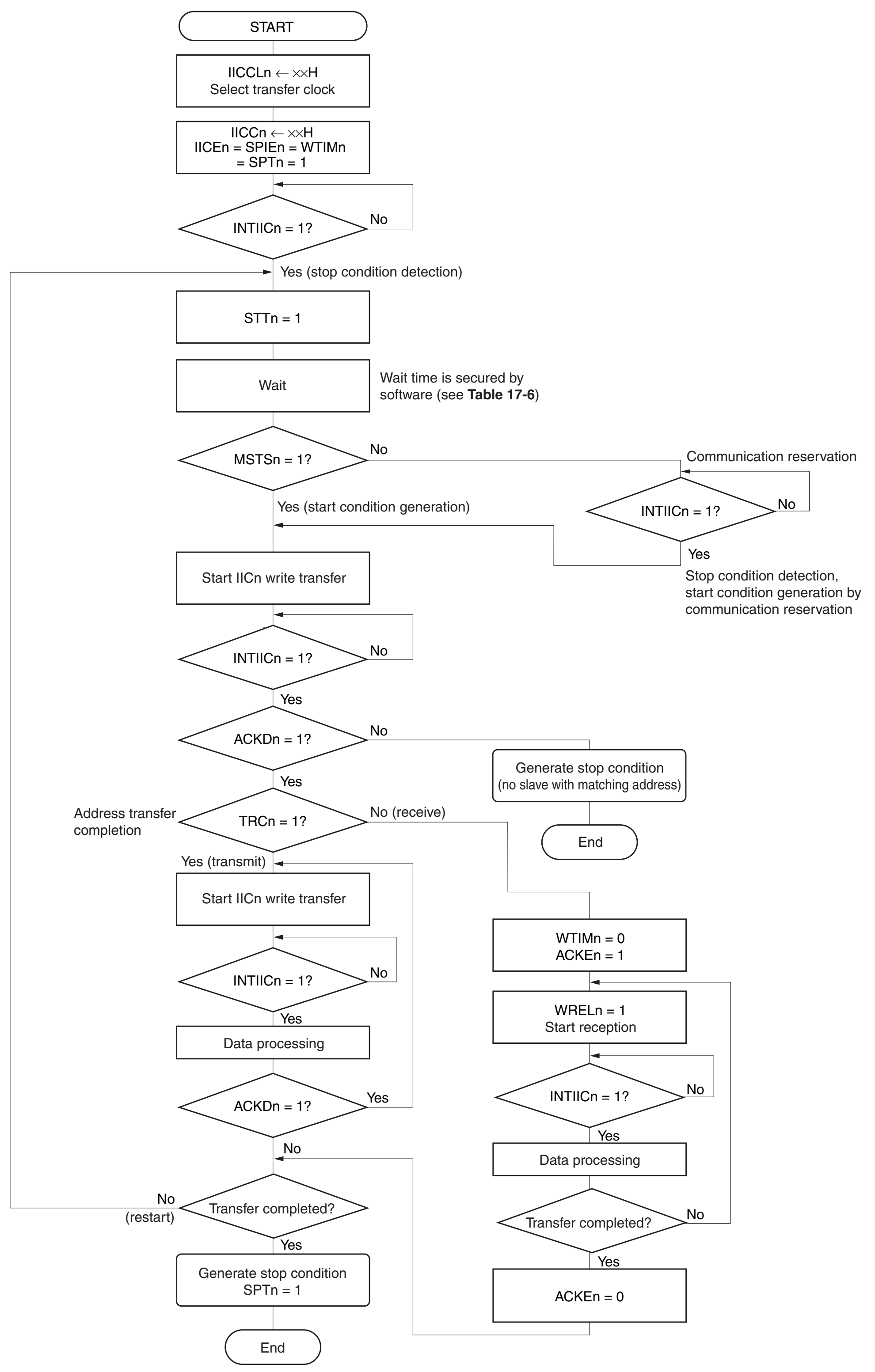


Figure 18-14 Master operation flowchart (1)

18.15.2 Master operation 2

The following shows the flowchart for master communication when the communication reservation function is disabled (IICRSVn bit = 1) and the master operation is started without detecting a stop condition (STCENn bit = 1).

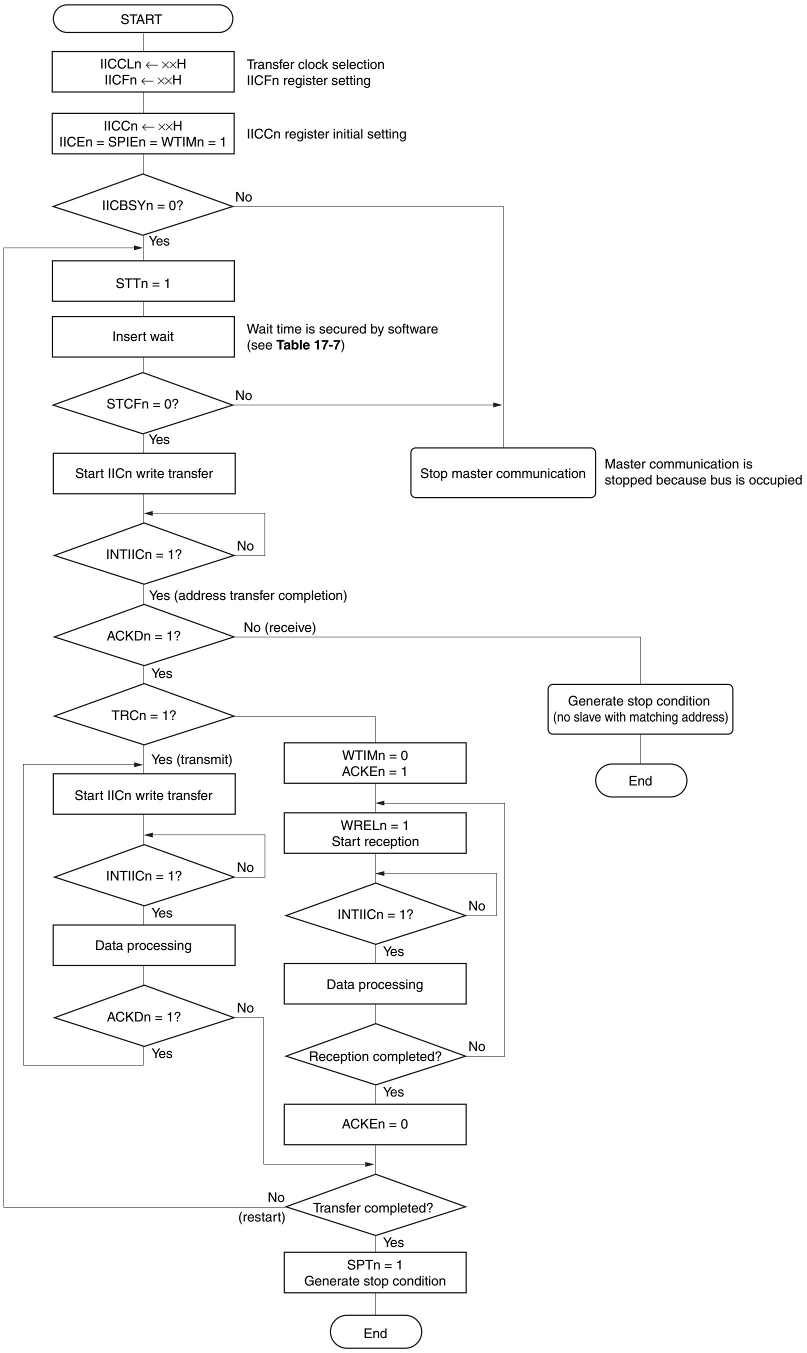


Figure 18-15 Master operation flowchart (2)

18.15.3 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.

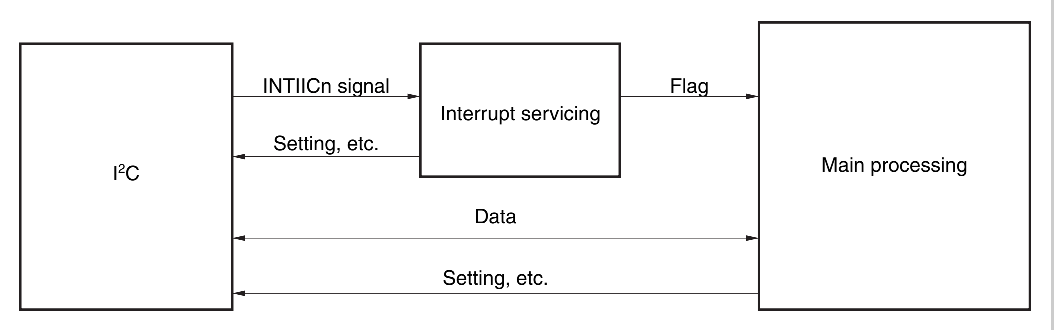


Figure 18-16 Software outline during slave operation

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^2C0n and wait for the communication enabled status. When communication is enabled, perform transfer using the communication mode flag and ready flag (the processing of the stop condition and start condition is performed by interrupts, conditions are confirmed by flags).

For transmission, repeat the transmission operation until the master device stops returning $\overline{\text{ACK}}$ signal. When the master device stops returning $\overline{\text{ACK}}$ signal, transfer is complete.

For reception, receive the required number of data and do not return $\overline{\text{ACK}}$ signal for the next data immediately after transfer is complete. After that, the master device generates the stop condition or restart condition. This causes exit from communications.

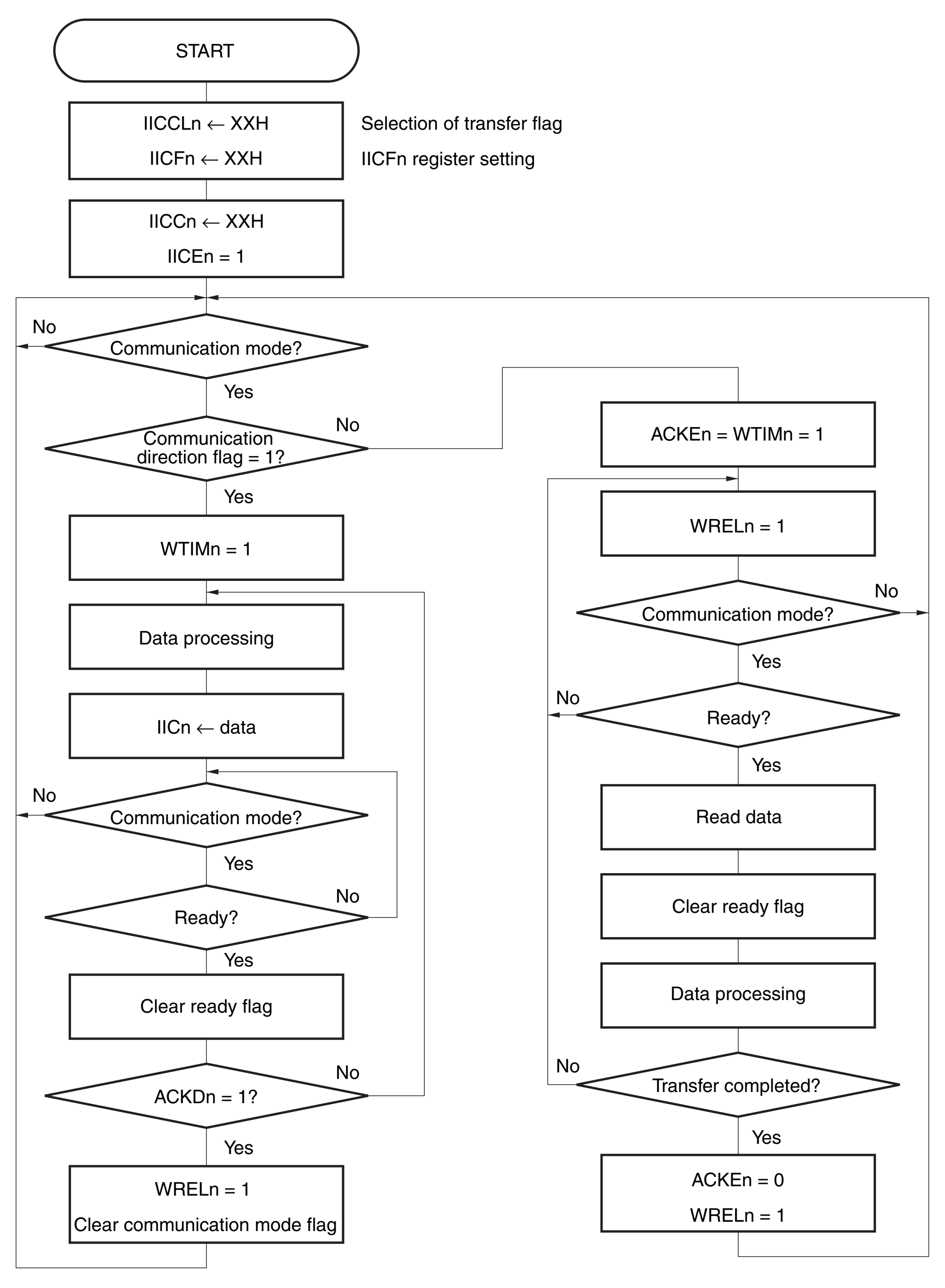


Figure 18-17 Slave operation flowchart (1)

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 18-18*.

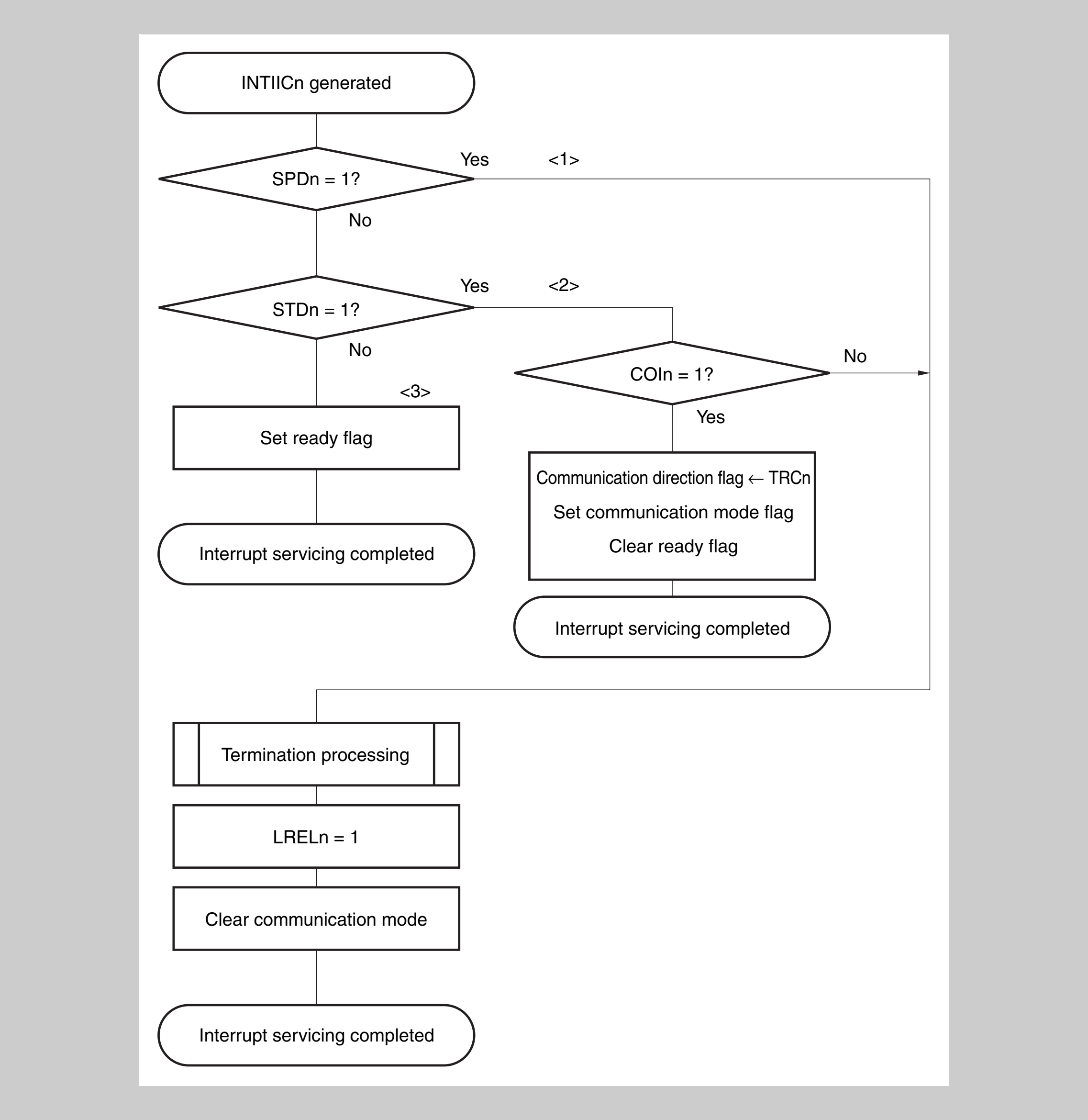


Figure 18-18 Slave operation flowchart (2)

18.16 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 (SCLn). The transmit data is transferred to the SO latch and is output (MSB first) via the SDAn pin.

Data input via the SDAn pin is captured by the IICn register at the rising edge of the SCLn pin.

The data communication timing is shown below.

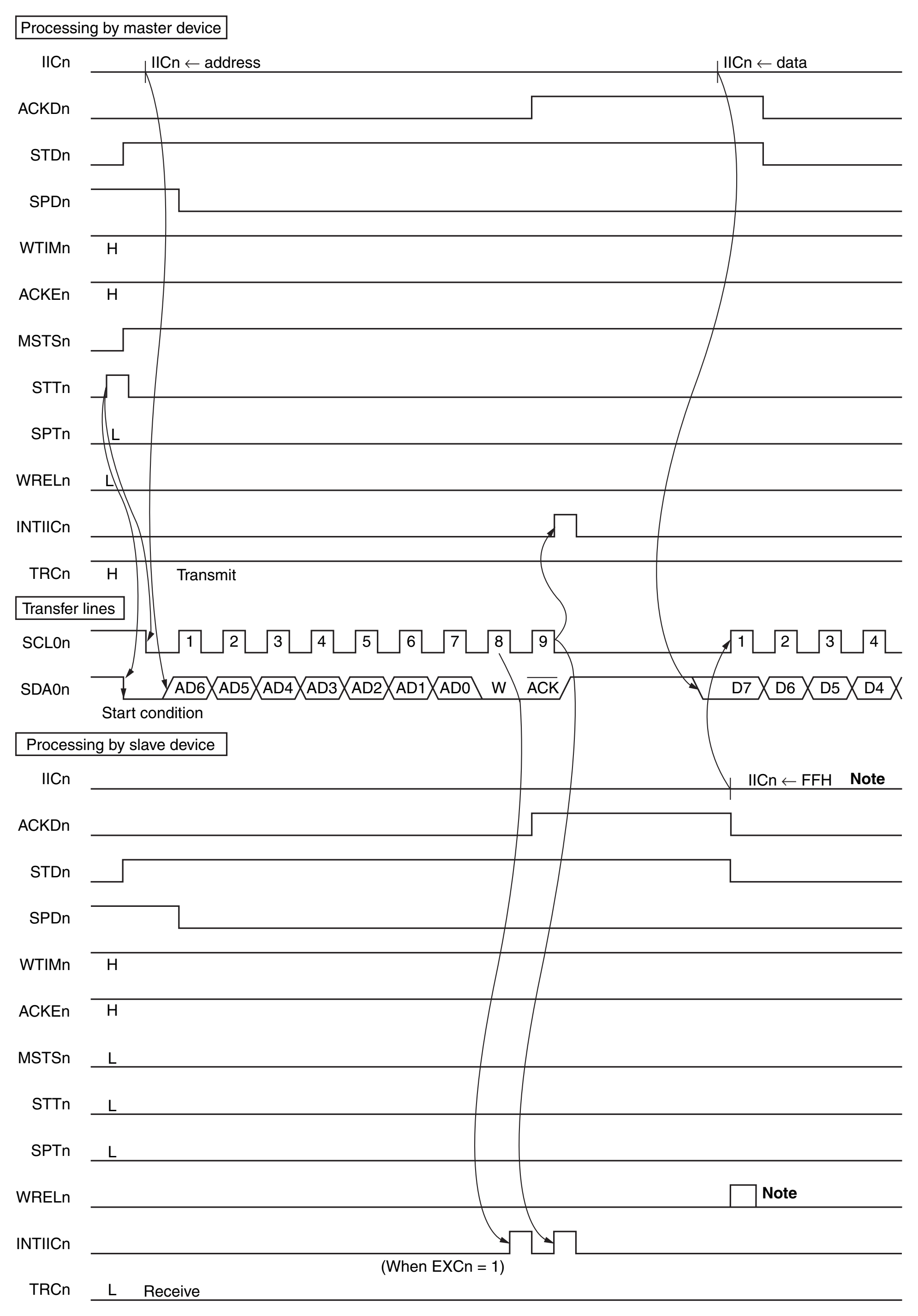


Figure 18-19 **Example of master to slave communication (when 9-clock wait is selected for both master and slave) (1/3) start condition ~ address**

Note To cancel slave wait, write FFH to IICn or set WRELn.

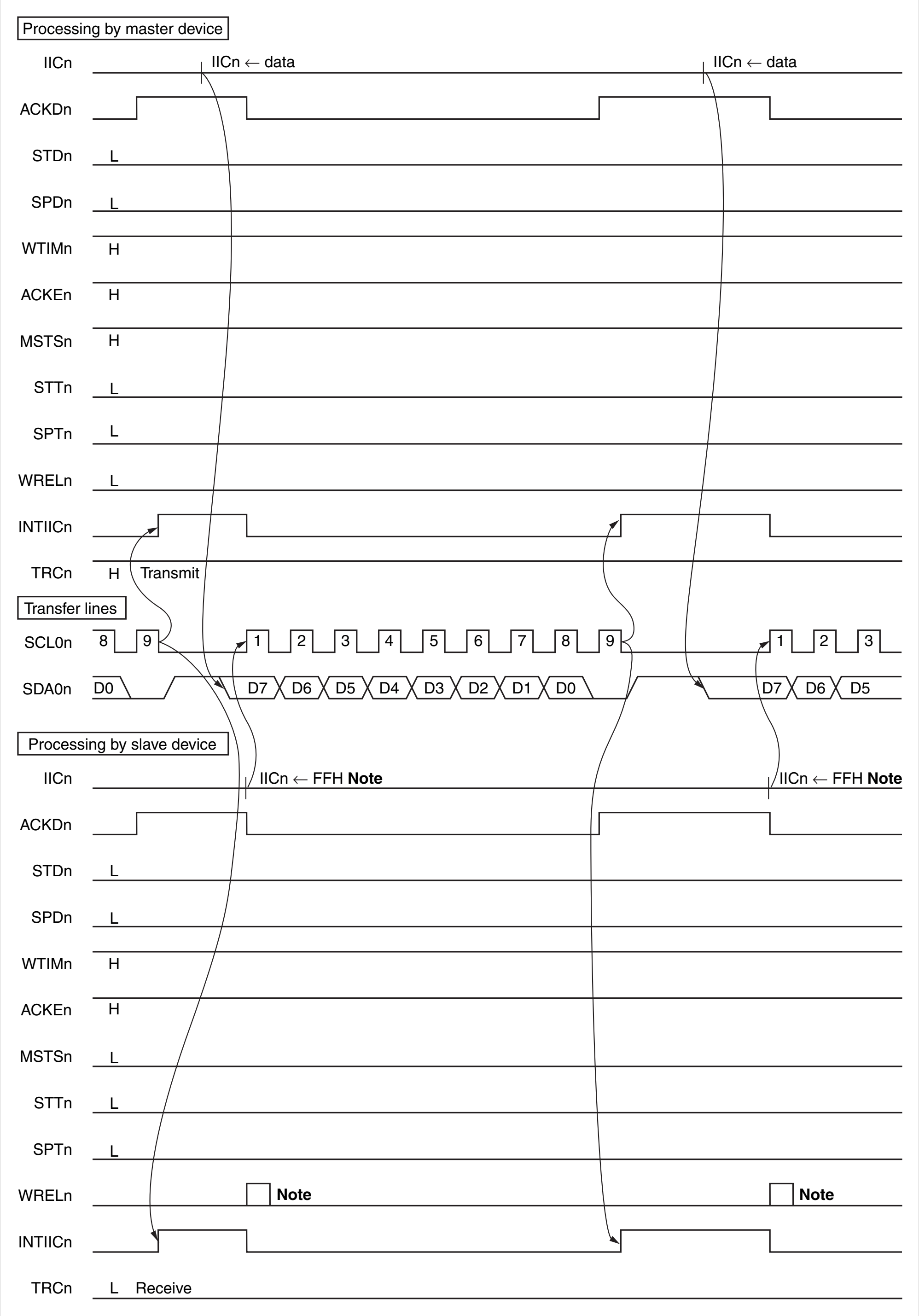


Figure 18-20 Example of master to slave communication (when 9-clock wait is selected for both master and slave) (2/3) (b) data

Note To cancel slave wait, write FFH to IICn or set WRELn.

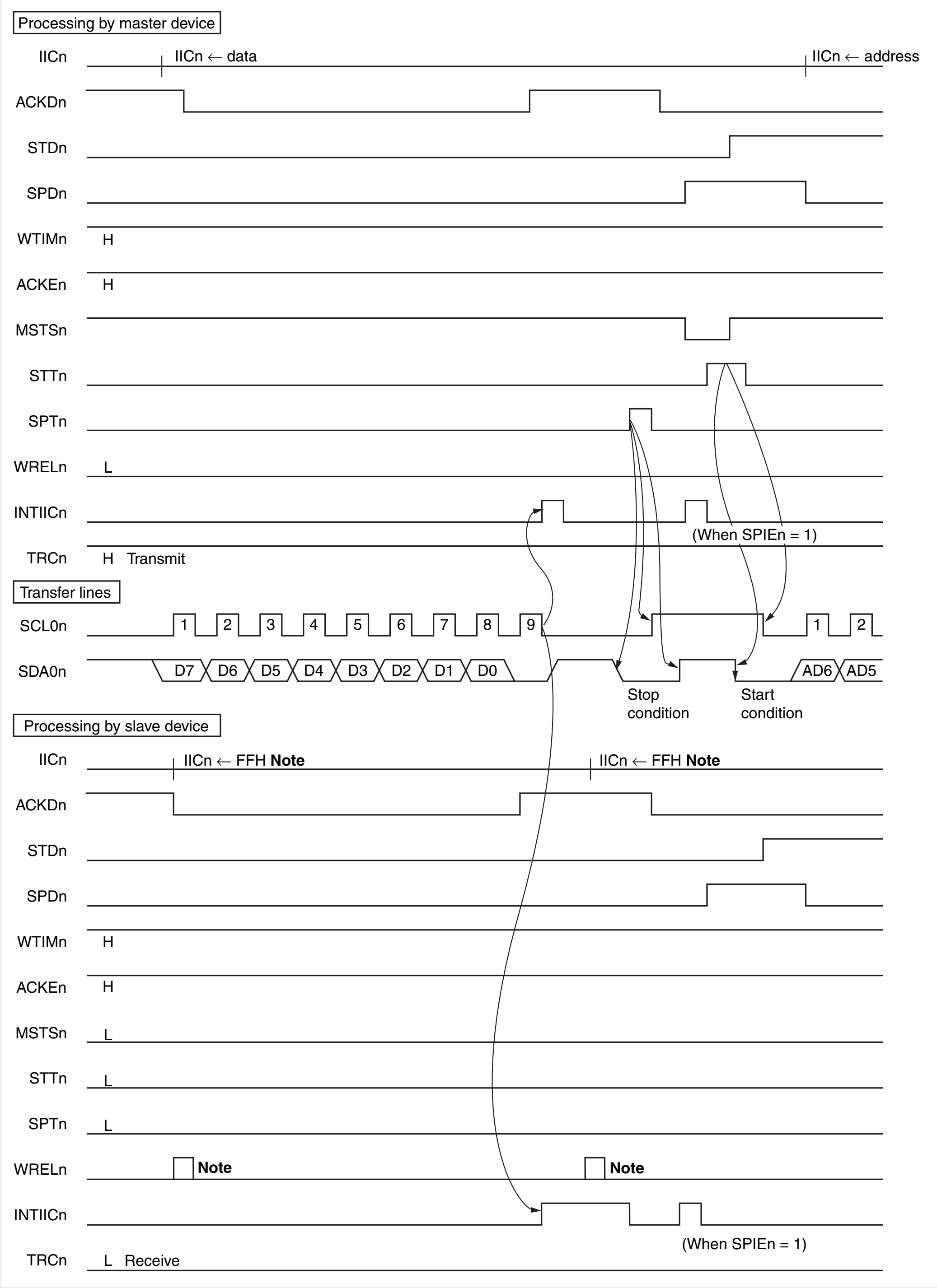


Figure 18-21 **Example of master to slave communication (when 9-clock wait is selected for both master and slave) (3/3) (c) stop condition**

Note To cancel slave wait, write FFH to IICn or set WRELn.

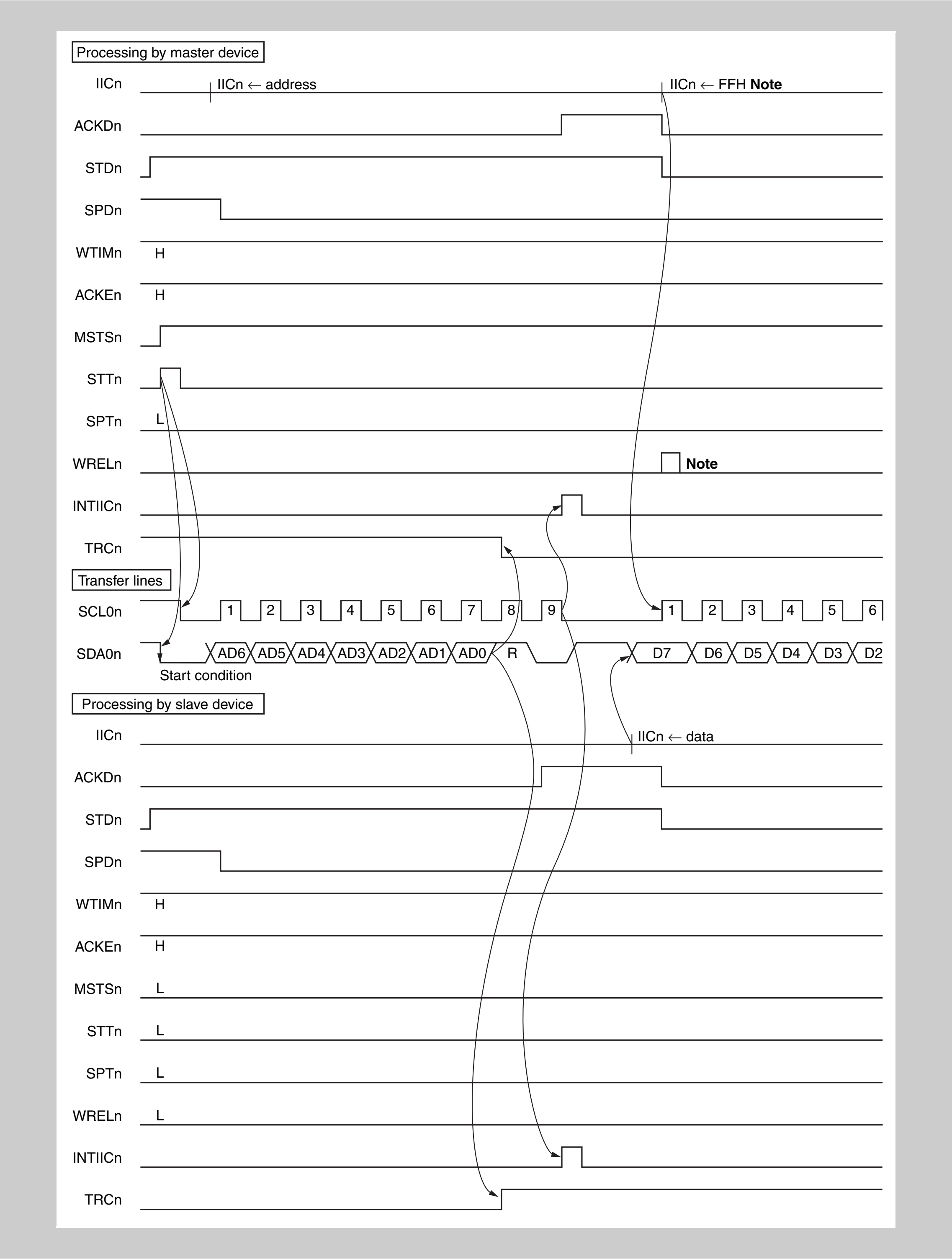


Figure 18-22 **Example of slave to master communication (when 9-clock wait is selected for both master and slave) (1/3) (a) start condition ~ address**

Note To cancel master wait, write FFH to IICn or set WRELn.

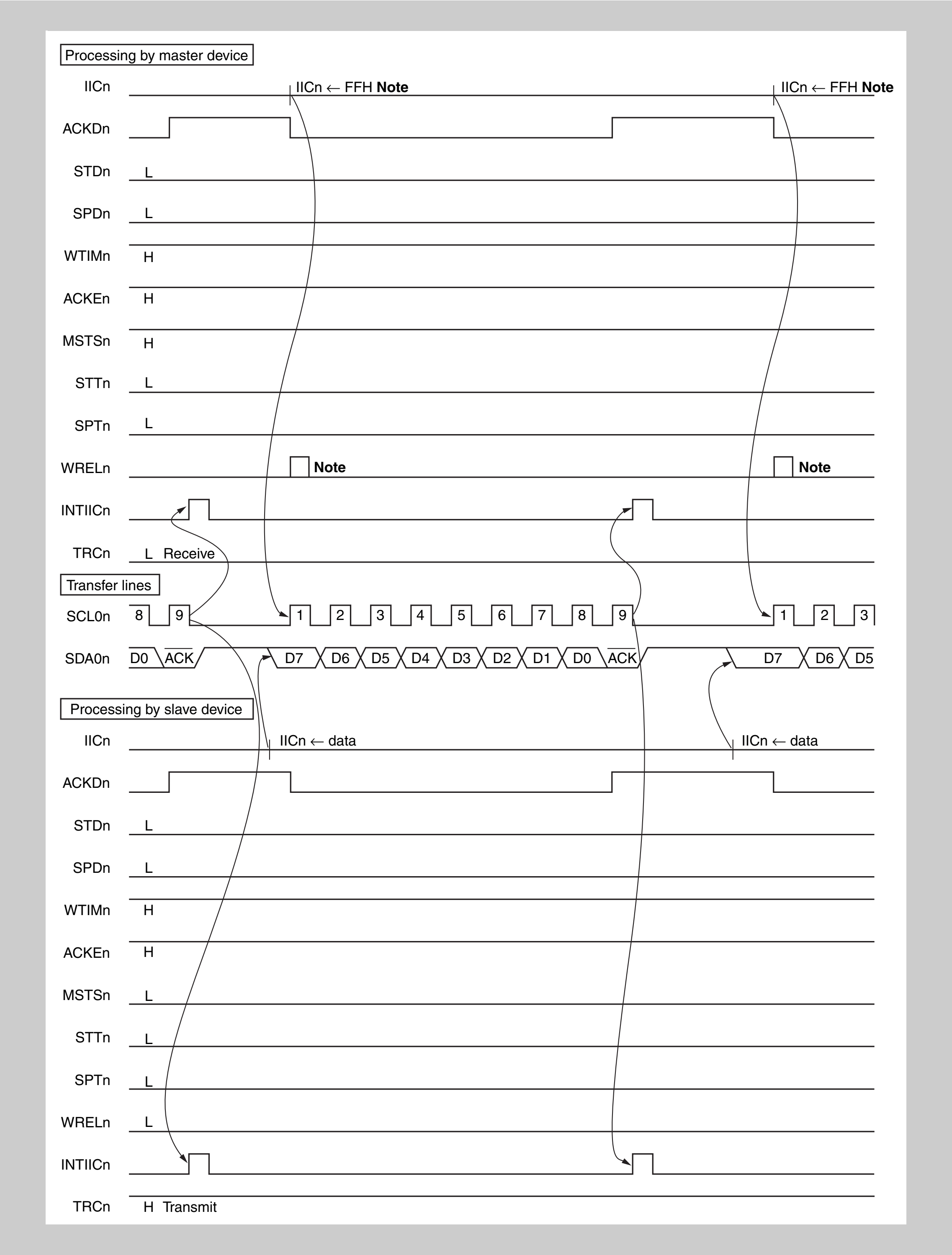


Figure 18-23 **Example of slave to master communication (when 9-clock wait is selected for both master and slave) (2/3) (b) data**

Note To cancel master wait, write FFH to IICn or set WRELn.

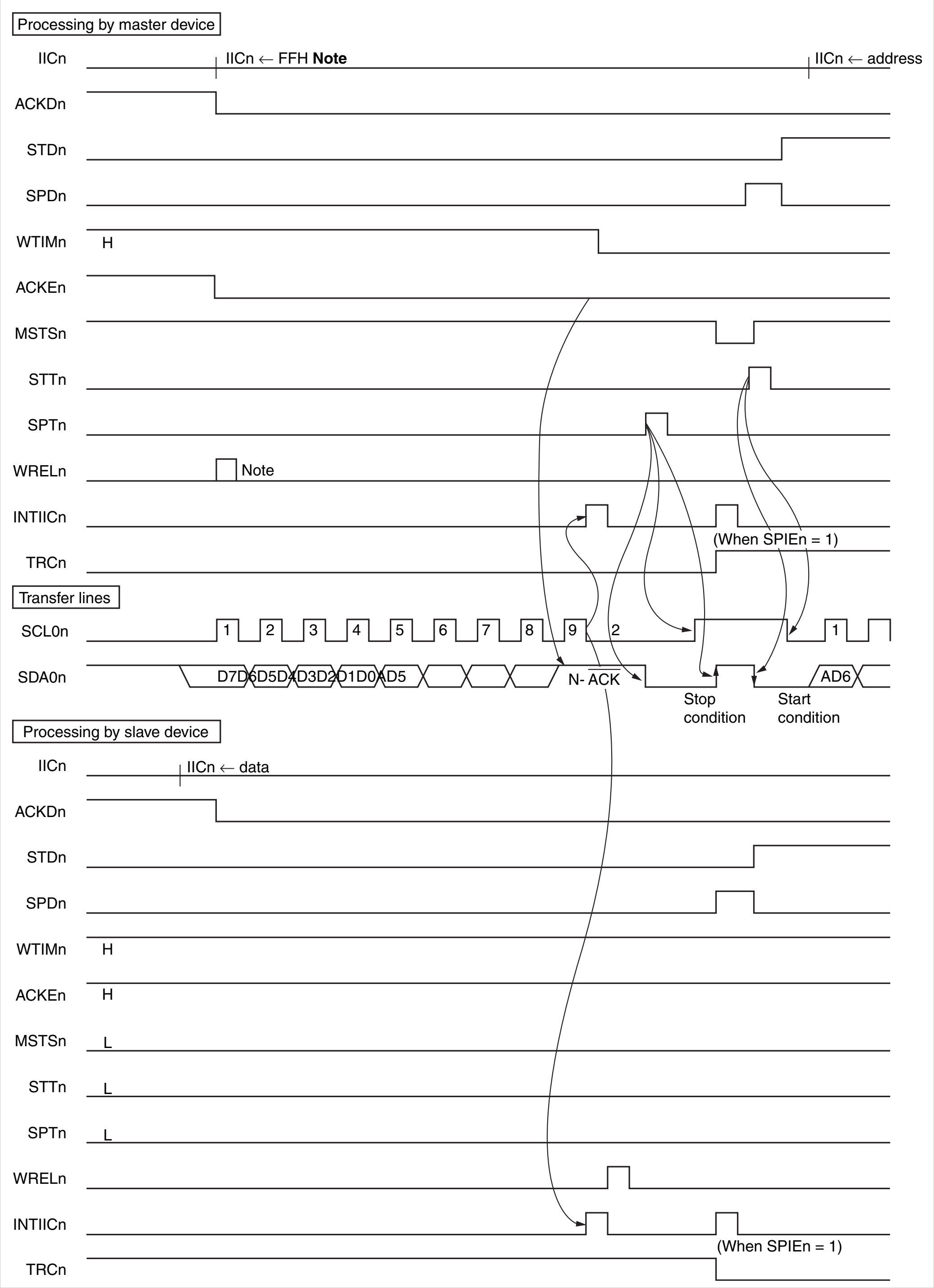


Figure 18-24 Example of slave to master communication (when 9-clock wait is selected for both master and slave) (3/3) (c) stop condition

Note To cancel master wait, write FFH to IICn or set WRELn.

Chapter 19 CAN Controller (CAN)

These microcontrollers feature an on-chip n-channel CAN (Controller Area Network) controller that complies with the CAN protocol as standardized in ISO 11898.

The V850E/Dx3 microcontrollers have following number of channels of the CAN controller:

CAN	All devices
Instances	2
Names	CAN0 to CAN1

Note

1. Throughout this chapter, the individual CAN channels are identified by “n”, for example CANn, or CnGMCTRL for the CANn global control register.
2. Throughout this chapter, the CAN message buffer registers are identified by “m” (m = 0 to 31), for example C0MDATA4m for CAN0 message data byte 4 of message buffer register m.

19.1 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)
- 32 message buffers per channel
- Receive/transmit history list function
- Automatic block transmission function
- Multi-buffer receive block function
- Mask setting of four patterns is possible for each channel
- Wake-Up capability on CAN receive data pins CRXDn
- Data bit time, communication baud rate and sample point can be controlled by CAN module bit-rate prescaler register (CnBRP) and bit rate register (CnBTR)
  - As an example the following sample-point configurations can be configured:
  - 66.7%, 70.0%, 75.0%, 80.0%, 81.3%, 85.0%, 87.5%
  - Baudrates in the range of 10 kbps up to 1000 kbps can be configured
- Enhanced features:
  - Each message buffer can be configured to operate as a transmit or a receive message buffer
  - Transmission priority is controlled by the identifier or by mailbox number (selectable)
  - A transmission request can be aborted by clearing the dedicated Transmit-Request flag of the concerned message buffer.
  - Automatic block transmission operation mode (ABT)
  - Time stamp function for CAN channels 0 and 1 in collaboration with timer Timer G0 and Timer G1 capture channels

19.1.1 Overview of functions

Table 19-1 presents an overview of the CAN Controller functions.

Table 19-1 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.
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
Forced release from bus-off state	• Default mode can be set while bus is off, so that bus can be forcibly released from the bus-off state.
Power save mode	• CAN Sleep mode (can be woken up by CAN bus) • CAN Stop mode (cannot be woken up by CAN bus)

19.1.2 Configuration

The CAN Controller is composed of the following four blocks.

- NPB interface
  This functional block provides an NPB (NEC Peripheral I/O Bus) interface and means of transmitting and receiving signals between the CAN module and the host CPU.
- MAC (Memory Access Controller)
  This functional block controls access to the CAN protocol layer and to the CAN RAM within the CAN module.
- CAN protocol layer
  This functional block is involved in the operation of the CAN protocol and its related settings.
- CAN RAM
  This is the CAN memory functional block, which is used to store message IDs, message data, etc.

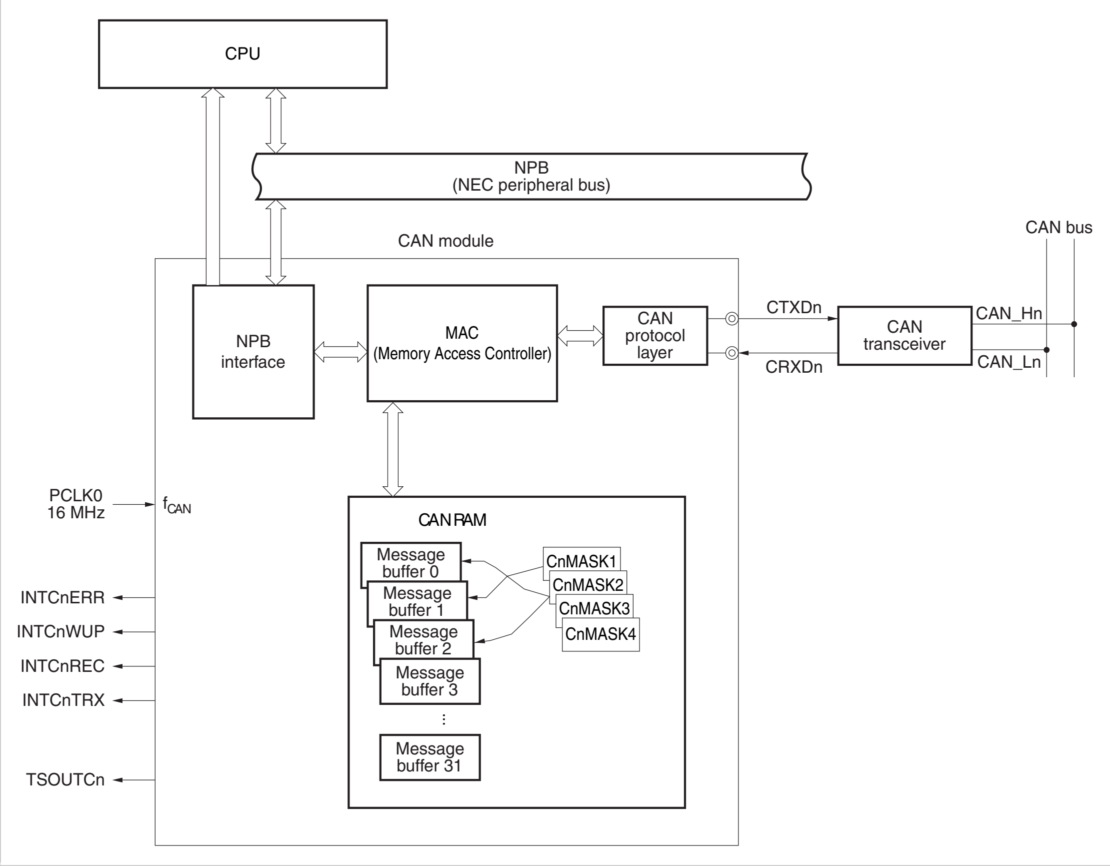


Figure 19-1 Block diagram of CAN module

19.2 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 19-2 Composition of layers

Note CAN Controller specification

19.2.1 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.

19.2.2 Frame types

The following four types of frames are used in the CAN protocol.

Table 19-2 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.

19.2.3 Data frame and remote frame

(1) Data frame

A data frame is composed of seven fields.

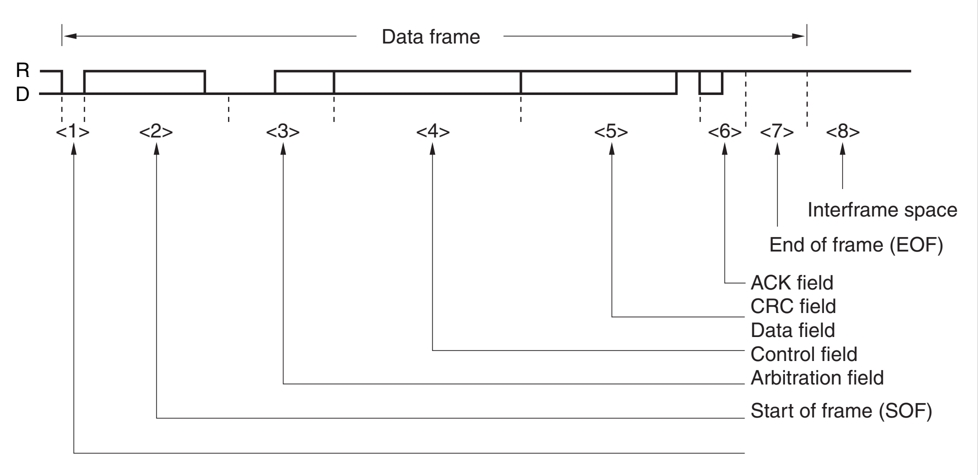


Figure 19-3 Data frame

Note D: Dominant = 0
R: Recessive = 1

(2) Remote frame

A remote frame is composed of six fields.

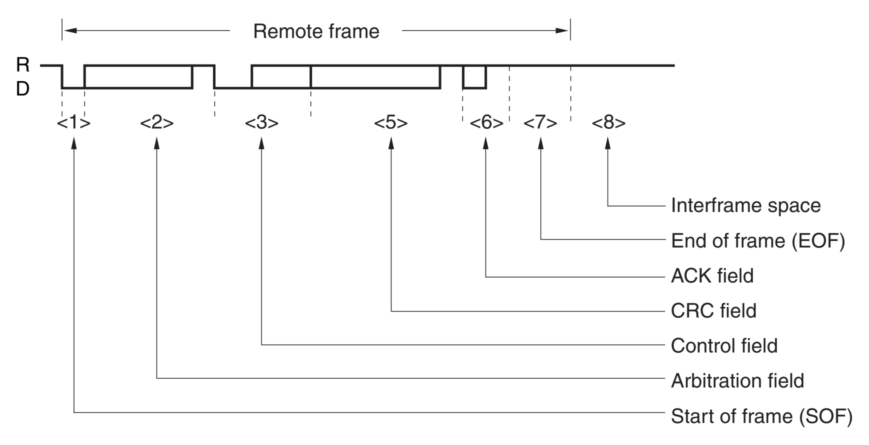


Figure 19-4 Remote frame

Note 1. The data field is not transferred even if the control field’s data length code is not “0000B”.

2. D: Dominant = 0
R: Recessive = 1

(3) Description of fields

<1> Start of frame (SOF)

The start of frame field is located at the start of a data frame or remote frame.

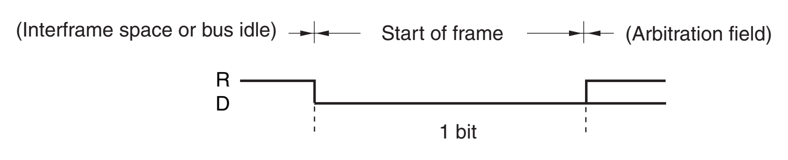


Figure 19-5 Start of frame (SOF)

Note D: Dominant = 0
R: Recessive = 1

- If dominant level is detected in the bus idle state, the start of frame is recognized.
- If recessive level is detected at the sample point of the start of frame, the preceding dominant level is judged as noise and the bus idle state is entered again.

<2> Arbitration field

The arbitration field is used to set the priority, data frame/remote frame, and frame format.

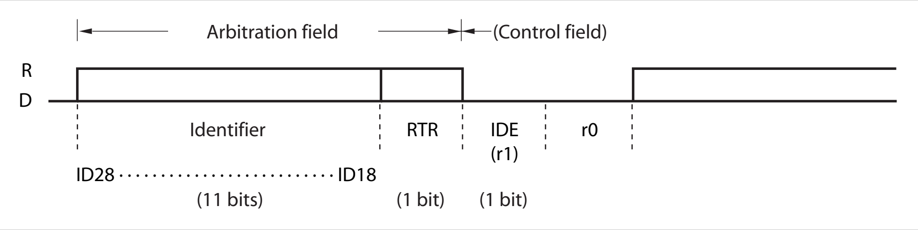


Figure 19-6 **Arbitration field (in standard format mode)**

Caution
1. ID28 to ID18 are identifiers.
2. An identifier is transmitted MSB first.

Note D: Dominant = 0
R: Recessive = 1

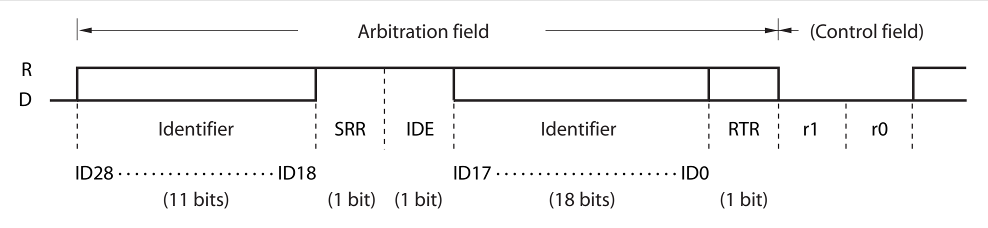


Figure 19-7 **Arbitration field (in extended format mode)**

Caution
1. ID28 to ID18 are identifiers.
2. An identifier is transmitted MSB first.

Note D: Dominant = 0
R: Recessive = 1

Table 19-3 RTR frame settings

Frame Type	RTR Bit
Data frame	0 (D)
Remote frame	1 (R)

Table 19-4 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

<3> Control field

The control field sets “N” as the number of data bytes in the data field (N = 0 to 8).

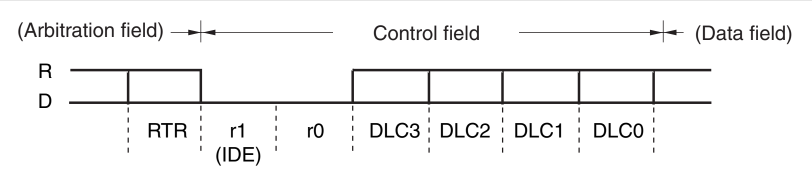


Figure 19-8 Control field

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 19-5 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 0000B.

<4> 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.

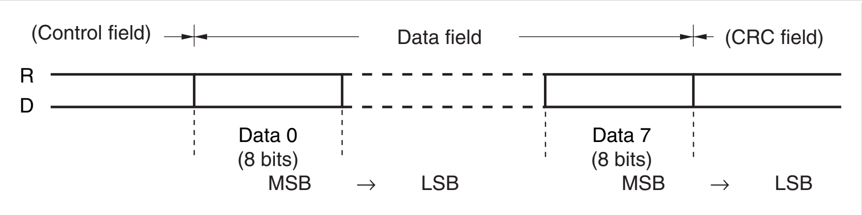


Figure 19-9 **Data field**

Note D: Dominant = 0
R: Recessive = 1

<5> CRC field

The CRC field is a 16-bit field that is used to check for errors in transmit data.

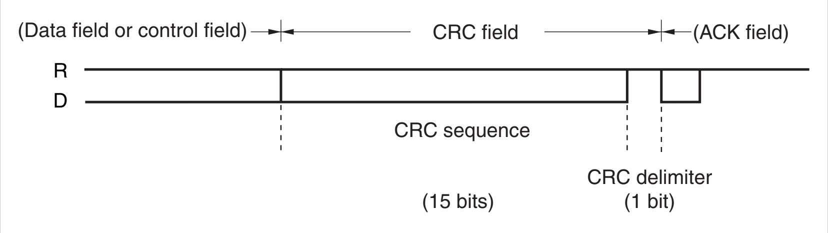


Figure 19-10 **CRC field**

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.

<6> ACK field

The ACK field is used to acknowledge normal reception.

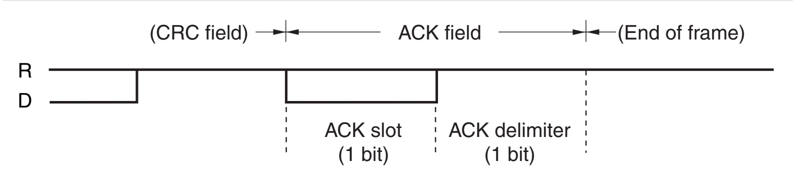


Figure 19-11 ACK field

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.

<7> End of frame (EOF)

The end of frame field indicates the end of data frame/remote frame.

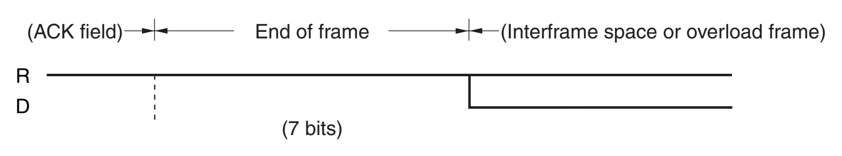


Figure 19-12 End of frame (EOF)

Note D: Dominant = 0
R: Recessive = 1

<8> 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.

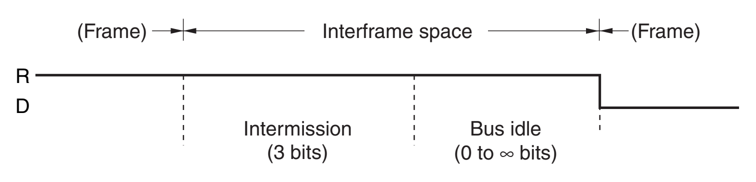


Figure 19-13 Interframe space (error active node)

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.

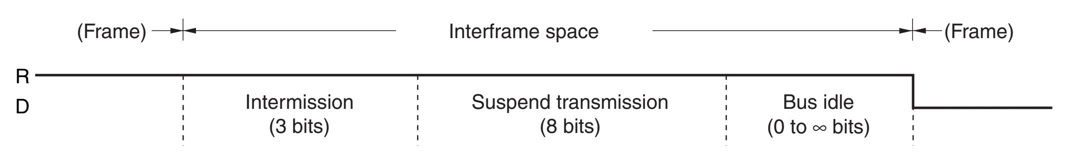


Figure 19-14 Interframe space (error passive node)

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 19-6 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.

19.2.4 Error frame

An error frame is output by a node that has detected an error.

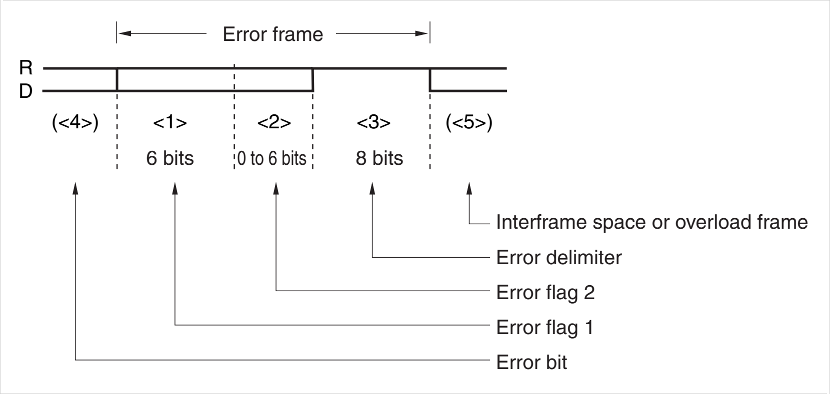


Figure 19-15 Error frame

Note D: Dominant = 0
R: Recessive = 1

Table 19-7 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.

19.2.5 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

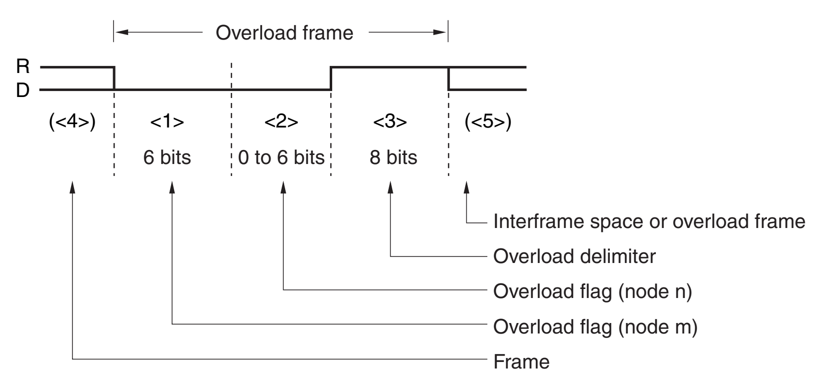


Figure 19-16 Overload frame

Note 1. D: Dominant = 0
R: Recessive = 1

2. Node n ≠ node m

Table 19-8 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.

19.3 Functions

19.3.1 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 19-9 Determining bus priority

Level match	Continuous transmission
Level mismatch	Continuous transmission

(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.

Caution 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.

19.3.2 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 19-10 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.

19.3.3 Multi masters

As the bus priority (a node acquiring transmit functions) is determined by the identifier, any node can be the bus master.

19.3.4 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.

19.3.5 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).

19.3.6 Error control function

(1) Error types

Table 19-11 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 19-12 Output timing of error frame

Type	Output Timing
Bit error, stuff error, form error, ACK error	Error frame output is started at the timing of the bit following the detected error.
CEC error	Error frame output is started at the timing of the bit following the ACK delimiter.

(3) Processing in case of error

The transmission node re-transmits the data frame or remote frame after the error frame. (However, it does not re-transmit the frame in the single-shot mode.)

(4) Error state

(a) Types of error states

The following three types of error states are defined by the CAN specification:

- Error active
- Error passive
- Bus-off

These types of error states are classified by the values of the CnERC.TEC7 to CnERC.TEC0 bits (transmission error counter bits) and the CnERC.REC6 to CnERC.REC0 bits (reception error counter bits) as shown in *Table 19-13*.

The present error state is indicated by the CnINFO register.

When each error counter value becomes equal to or greater than the error warning level (96), the CnINFO.TECS0 or CnINFO.RECS0 bit 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 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 CnINFO.BOFF bit is set to 1.
- If only one node is active on the bus at startup (i.e., 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 19-13 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. <1> TSOUT toggles. <2> REC is incremented/decremented. <3> VALID bit is set. If the initialization mode is set and then 11 recessive-level bits are generated 128 times in a row in an operation mode other than the initialization mode, the error counter is reset to 0 and the error active state can be restored.

Note 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 during the first bit of the error delimiter.

Table 19-14 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 (REPS bit = 0)
Receiving node detects dominant level following error flag of error frame.	No change	+8 (REPS bit = 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 (receiving, REPS bit = 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, REPS bit = 0) • ±0 (REC6 to REC0 = 0, REPS bit = 0) • Any value of 119 to 127 is set (REPS bit = 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 transmission pins (CTXDn) cut off from the CAN bus always output the recessive level.

The CAN module recovers from the bus-off state in the following bus-off recovery sequence.

<1> Request to enter the CAN initialization mode

<2> Request to enter a CAN operation mode

- (a) Recovery operation through normal recovery sequence
- (b) Forced recovery operation that skips recovery sequence

(a) Recovery from bus-off state through normal recovery sequence

The CAN module first issues a request to enter the initialization mode (refer too timing <1> in *Figure 19-17 on page 660*). This request will be immediately acknowledged, and the CnCTRL.OPMODE bit is cleared to 000B. 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 CnGMCTRL.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 19-17 on page 660*). 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 or more. 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 19-17 on page 660*), 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. Whether the CAN module has entered the operation mode can be confirmed by reading OPMODE.

During the bus-off period and bus-off recovery sequence, the CnINFO.BOFF bit stays set (to 1). In the bus-off recovery sequence, the reception error counter (CnERC.REC0 to CnERC.REC6) 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 the REC0 to REC6 bits.

Caution In the bus-off recovery sequence, the REC0 to REC6 bits 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 be released from the bus-off state, the module must enter the initialization mode once. If the module is in the CAN sleep mode or CAN stop mode, however, it cannot enter the initialization mode. In this case, release the module from the CAN sleep or stop mode, and then make a request to place the module in the initialization mode.

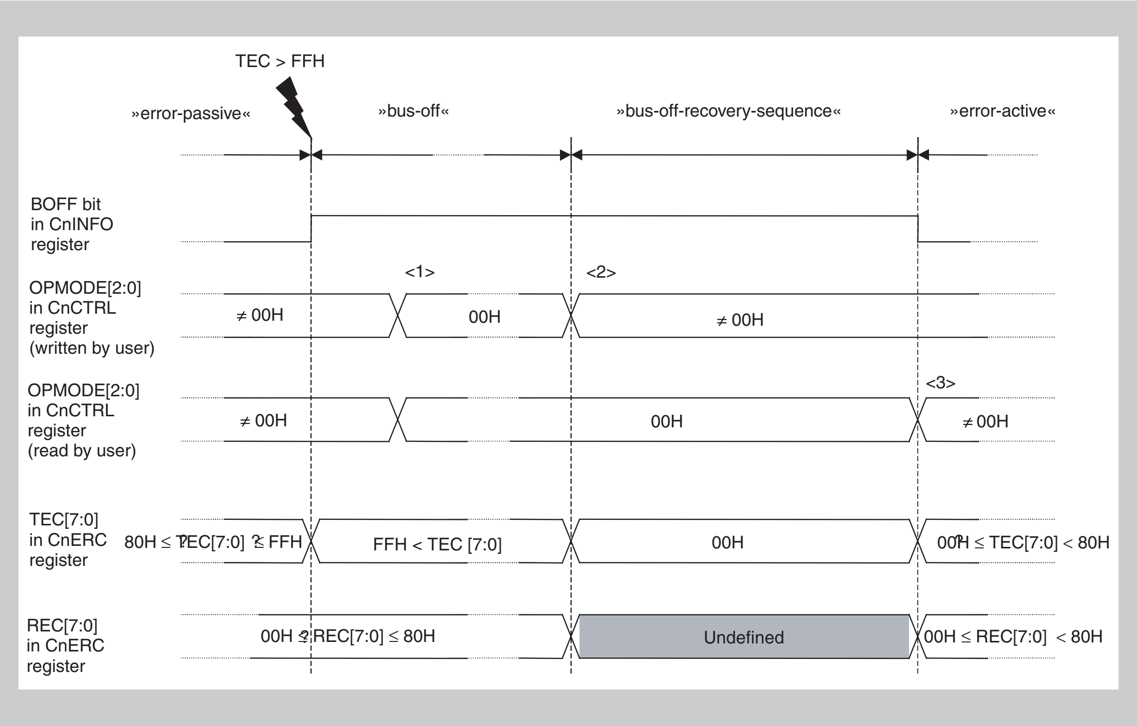


Figure 19-17 Recovery from bus-off state through normal recovery sequence

(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 659*.

Next, the module requests to enter an operation mode. At the same time, the CnCTRL.CCERC bit 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 19-51 on page 762*.

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 CnERC and CnINFO registers for debugging or evaluating a program, they can be initialized to the default value by setting the CnCTRL.CCERC bit 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.

19.3.7 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 base clock (f_{TQ}) that is the CAN module system clock (f_{CANMOD}) divided by 1 to 256 (*"CnBRP - CANn module bit rate prescaler register" on page 693*).

(2) Data bit time (8 to 25 time quanta)

One data bit time is defined as follows.

1 time quantum = $1/f_{TQ}$

The CAN Controller sets time segment 1, time segment 2, and reSynchronization Jump Width (SJW) as the data bit time, as shown in *Figure 19-18*. 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.

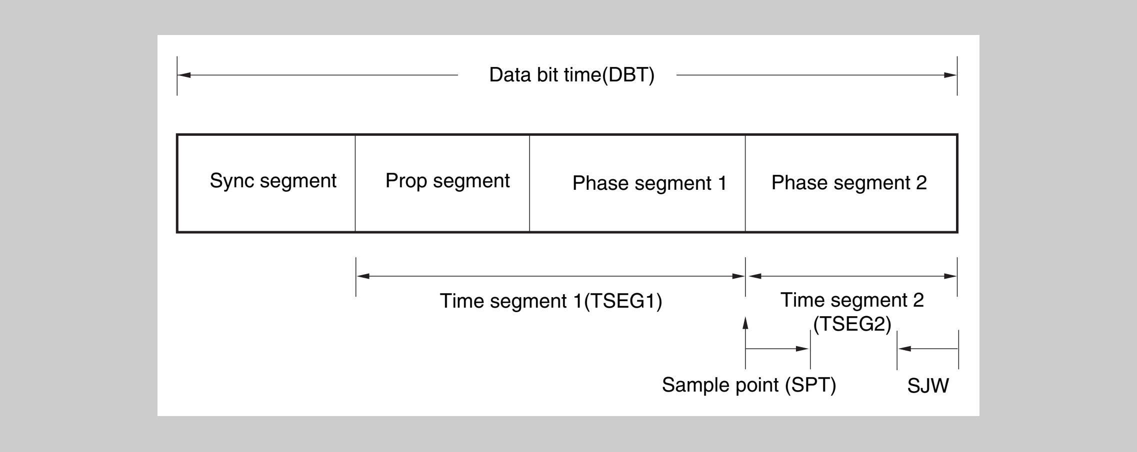


Figure 19-18 Segment setting

Table 19-15 Segment setting

Segment name	Settable range	Notes on setting to conform to CAN specification
Time segment 1 (TSEG1)	2TQ to 15TQ	-
Time segment 2 (TSEG2)	1TQ to 8TQ	IPT^{Note} of the CAN Controller is 0TQ. To conform to the CAN protocol specification, therefore, a length 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 settable upper limit of SJW is the length of time segment 1 minus 1TQ.

Note
1. IPT: Information Processing Time
2. Reference: The CAN protocol specification defines the segments constituting the data bit time as shown in *Figure 19-19*.

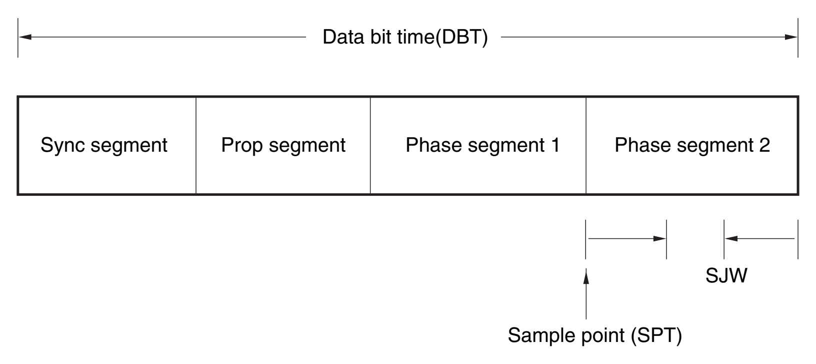


Figure 19-19 Configuration of data bit time defined by CAN specification

Table 19-16 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 (Propagation segment)	Programmable to 1 to 8	This segment absorbs the delay of the output buffer, CAN bus, and input buffer. The length of this segment is set so that ACK is returned before the start of phase segment 1. Time of prop segment ≥ (Delay of output buffer) + 2 × (Delay of CAN bus) + (Delay of input buffer) This segment compensates for an error in the data bit time. The longer this segment, the wider the permissible range but the slower the communication speed.
Phase segment 1 (Phase buffer segment 1)	Programmable to 1 to 8	
Phase segment 2 (Phase buffer segment 2)	Phase segment 1 or IPT[Note], whichever greater	
SJW (reSynchronization Jump Width)	Programmable from 1 to phase segment 1 to 4, whichever is smaller	This width sets the upper limit of expansion or contraction of the phase segment during resynchronization.

Note 1. 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.

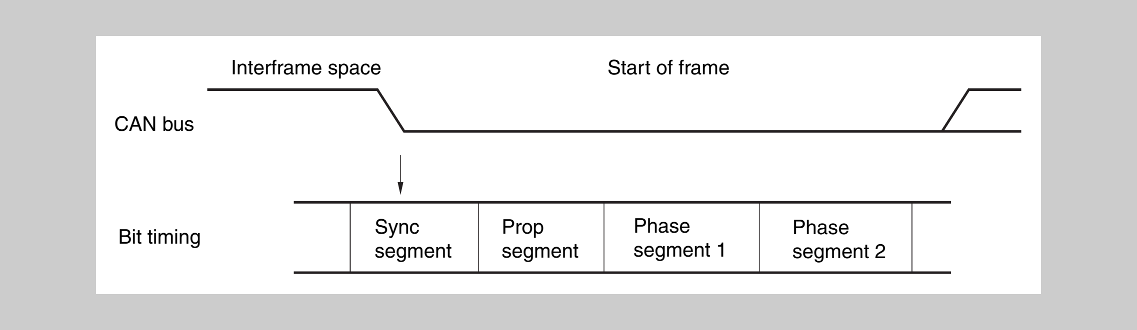


Figure 19-20 Adjusting synchronization of data bit

(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 longer by specified SJW.

  If phase error is negative: Phase segment 2 is shorter 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.

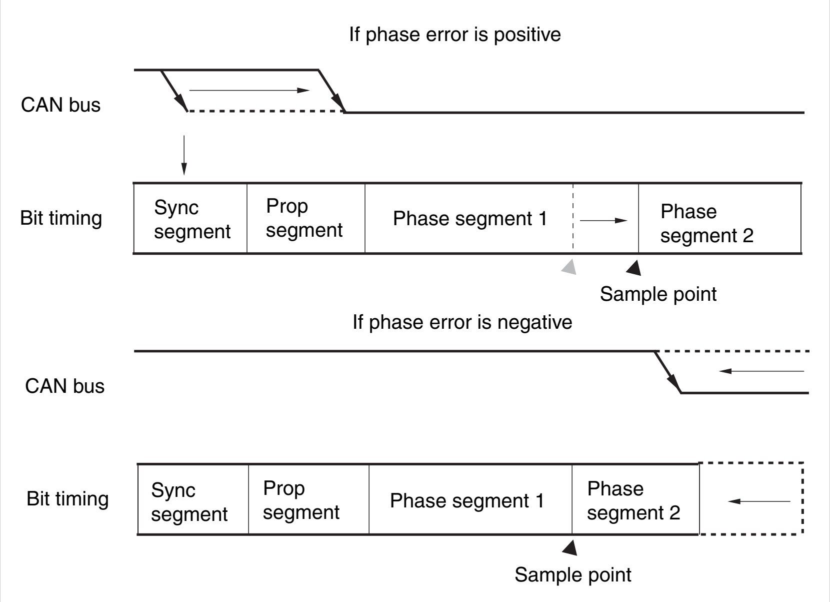


Figure 19-21 Resynchronization

19.4 Connection with Target System

The CAN module has to be connected to the CAN bus using an external transceiver.

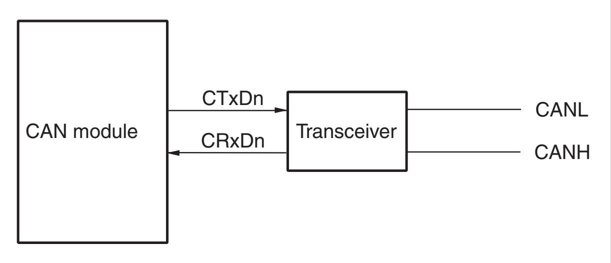


Figure 19-22 Connection to CAN bus

19.5 Internal Registers of CAN Controller

19.5.1 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 122* or to *“Programmable peripheral I/O area (PPA)“ on page 257*).

The addresses given in the following tables are offsets to the programmable peripheral area base address PBA.

The recommended setting of PBA is 8FFB$_H$. This setting would define the programmable peripheral area base address

PBA = 03FE C000$_H$

Table 19-17 lists all base addresses used throughout this chapter.

Table 19-17 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$_H$

In the following <CnRBaseAddr> respectively <CnMBaseAddr> are used for the base address names for CAN channel n.

19.5.2 CAN controller configuration

Table 19-18 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 mask3 register (CnMASK3L, CnMASK3H)
	CANn module mask 4 registers (CnMASK4L, CnMASK4H)
	CANn module control register (CnCTRL)
	CANn module last error information register (CnLEC)
	CANn module information register (CnINFO)
	CANn module error counter register (CnERC)
	CANn module interrupt enable register (CnIE)
	CANn module interrupt status register (CnINTS)
	CANn module bit rate prescaler register (CnBRP)
	CANn module bit rate register (CnBTR)
	CANn module last in-pointer register (CnLIPT)
	CANn module receive history list register (CnRGPT)
	CANn module last out-pointer register (CnLOPT)
	CANn module transmit history list register (CnTGPT)
	CANn module time stamp register (CnTS)
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)

19.5.3 CAN registers overview

(1) CAN0 module registers

The following table lists the address offsets to the CAN0 register base address:

C0RBaseAddr = PBA

Table 19-19 CAN0 global and module registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
000H	CAN0 global control register	C0GMCTRL	R/W			√	0000H
002H	CAN0 global clock selection register	C0GMCS			√		0FH
006H	CAN0 global automatic block transmission register	C0GMABT				√	0000H
008H	CAN0 global automatic block transmission delay register	C0GMABTD			√		00H
040H	CAN0 module mask 1 register	C0MASK1L				√	Undefined
042H		C0MASK1H				√	Undefined
044H	CAN0 module mask 2 register	C0MASK2L				√	Undefined
046H		C0MASK2H				√	Undefined
048H	CAN0 module mask 3 register	C0MASK3L				√	Undefined
04AH		C0MASK3H				√	Undefined
04CH	CAN0 module mask 4 register	C0MASK4L				√	Undefined
04EH		C0MASK4H				√	Undefined
050H	CAN0 module control register	C0CTRL				√	0000H
052H	CAN0 module last error code register	C0LEC			√		00H
053H	CAN0 module information register	C0INFO	R		√		00H
054H	CAN0 module error counter register	C0ERC				√	0000H
056H	CAN0 module interrupt enable register	C0IE	R/W			√	0000H
058H	CAN0 module interrupt status register	C0INTS				√	0000H
05AH	CAN0 module bit-rate prescaler register	C0BRP			√		FFH
05CH	CAN0 module bit-rate register	C0BTR				√	370FH
05EH	CAN0 module last in-pointer register	C0LIPT	R		√		Undefined
060H	CAN0 module receive history list register	C0RGPT	R/W			√	xx02H
062H	CAN0 module last out-pointer register	C0LOPT	R		√		Undefined
064H	CAN0 module transmit history list register	C0TGPT	R/W			√	xx02H
066H	CAN0 module time stamp register	C0TS				√	0000H

The addresses in the following table denote the address offsets to the CAN0 message buffer base address:

C0MBaseAddr = PBA + 100_H

Example CAN0, message buffer register m = 14 = E_H, byte 6 C0MDATA614 has the address E_H x 20_H + 6_H + C0MBaseAddr

Note The message buffer register number m in the register symbols has 2 digits, for example,
COMDATA01m = COMDATA0100 for m = 0.

Table 19-20 CAN0 message buffer registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
$mx20_H + 0_H$	CAN0 message data byte 01 register m	C0MDATA01m	R/W			√	Undefined
$mx20_H + 0_H$	CAN0 message data byte 0 register m	C0MDATA0m			√		Undefined
$mx20_H + 1_H$	CAN0 message data byte 1 register m	C0MDATA1m			√		Undefined
$mx20_H + 2_H$	CAN0 message data byte 23 register m	C0MDATA23m				√	Undefined
$mx20_H + 2_H$	CAN0 message data byte 2 register m	C0MDATA2m			√		Undefined
$mx20_H + 3_H$	CAN0 message data byte 3 register m	C0MDATA3m			√		Undefined
$mx20_H + 4_H$	CAN0 message data byte 45 register m	C0MDATA45m				√	Undefined
$mx20_H + 4_H$	CAN0 message data byte 4 register m	C0MDATA4m			√		Undefined
$mx20_H + 5_H$	CAN0 message data byte 5 register m	C0MDATA5m			√		Undefined
$mx20_H + 6_H$	CAN0 message data byte 67 register m	C0MDATA67m				√	Undefined
$mx20_H + 6_H$	CAN0 message data byte 6 register m	C0MDATA6m			√		Undefined
$mx20_H + 7_H$	CAN0 message data byte 7 register m	C0MDATA7m			√		Undefined
$mx20_H + 8_H$	CAN0 message data length register m	C0MDLCm			√		$0000\ xxxx_B$
$mx20_H + 9_H$	CAN0 message configuration register m	C0MCONFm			√		Undefined
$mx20_H + A_H$	CAN0 message identifier register m	C0MIDLm				√	Undefined
$mx20_H + C_H$		C0MIDHm				√	Undefined
$mx20_H + E_H$	CAN0 message control register m	C0MCTRLm				√	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 19-21 CAN1 global and module registersM

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
000H	CAN1 global control register	C1GMCTRL	R/W			√	0000H
002H	CAN1 global clock selection register	C1GMCS			√		0FH
006H	CAN1 global automatic block transmission register	C1GMABT				√	0000H
008H	CAN1 global automatic block transmission delay register	C1GMABTD			√		00H
040H	CAN1 module mask 1 register	C1MASK1L				√	Undefined
042H		C1MASK1H				√	Undefined
044H	CAN1 module mask 2 register	C1MASK2L				√	Undefined
046H		C1MASK2H				√	Undefined
048H	CAN1 module mask 3 register	C1MASK3L				√	Undefined
04AH		C1MASK3H				√	Undefined
04CH	CAN1 module mask 4 register	C1MASK4L				√	Undefined
04EH		C1MASK4H				√	Undefined
050H	CAN1 module control register	C1CTRL				√	0000H
052H	CAN1 module last error code register	C1LEC			√		00H
053H	CAN1 module information register	C1INFO	R		√		00H
054H	CAN1 module error counter register	C1ERC				√	0000H
056H	CAN1 module interrupt enable register	C1IE	R/W			√	0000H
058H	CAN1 module interrupt status register	C1INTS				√	0000H
05AH	CAN1 module bit-rate prescaler register	C1BRP			√		FFH
05CH	CAN1 module bit-rate register	C1BTR				√	370FH
05EH	CAN1 module last in-pointer register	C1LIPT	R		√		Undefined
060H	CAN1 module receive history list register	C1RGPT	R/W			√	xx02H
062H	CAN1 module last out-pointer register	C1LOPT	R		√		Undefined
064H	CAN1 module transmit history list register	C1TGPT	R/W			√	xx02H
066H	CAN1 module time stamp register	C1TS				√	0000H

The addresses in the following table denote the address offsets to the CAN0 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 19-22 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

19.5.4 Register bit configuration

Table 19-23 CAN global register bits 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
00H	CnGMCTRL (W)	0	0	0	0	0	0	0	Clear GOM
01H		0	0	0	0	0	0	Set EFSD	Set GOM
00H	CnGMCTRL (R)	0	0	0	0	0	0	EFSD	GOM
01H		MBON	0	0	0	0	0	0	0
02H	CnGMCS	0	0	0	0	CCP3	CCP2	CCP1	CCP0
06H	CnGMABT (W)	0	0	0	0	0	0	0	Clear ABTTRG
07H		0	0	0	0	0	0	Set ABTCLR	Set ABTTRG
06H	CnGMABT (R)	0	0	0	0	0	0	ABTCLR	ABTTRG
07H		0	0	0	0	0	0	0	0
08H	CnGMABTD	0	0	0	0	ABTD3	ABTD2	ABTD1	ABTD0

a) Base address: <CnRBaseAddr>

Table 19-24 CAN module register bit configuration (1/2)

Address offset[a]	Symbol	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
40H	CnMASK1L	CMID7 to CMID0							
41H		CMID15 to CMID8							
42H	CnMASK1H	CMID23 to CMID16							
43H		0	0	0	CMID28 to CMID24				
44H	CnMASK2L	CMID7 to CMID0							
45H		CMID15 to CMID8							
46H	CnMASK2H	CMID23 to CMID16							
47H		0	0	0	CMID28 to CMID24				
48H	CnMASK3L	CMID7 to CMID0							
49H		CMID15 to CMID8							
4AH	CnMASK3H	CMID23 to CMID16							
4BH		0	0	0	CMID28 to CMID24				
4CH	CnMASK4L	CMID7 to CMID0							
4DH		CMID15 to CMID8							
4EH	CnMASK4H	CMID23 to CMID16							
4FH		0	0	0	CMID28 to CMID24				
50H	CnCTRL (W)	0	Clear AL	Clear VALID	Clear PSMODE1	Clear PSMODE0	Clear OPMODE2	Clear OPMODE1	Clear OPMODE0
51H		Set CCERC	Set AL	0	Set PSMODE1	Set PSMODE0	Set OPMODE2	Set OPMODE1	Set OPMODE0
50H	CnCTRL (R)	CCERC	AL	VALID	PS MODE1	PS MODE0	OP MODE2	OP MODE1	OP MODE0
51H		0	0	0	0	0	0	RSTAT	TSTAT
52H	CnLEC (W)	0	0	0	0	0	0	0	0

Table 19-24 CAN module register bit configuration (2/2)

Address offset[a]	Symbol	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
52H	CnLEC (R)	0	0	0	0	0	LEC2	LEC1	LEC0
53H	CnINFO	0	0	0	BOFF	TECS1	TECS0	RECS1	RECS0
54H	CnERC	TEC7 to TEC0							
55H		REC7 to REC0							
56H	CnIE (W)	0	0	Clear CIE5	Clear CIE4	Clear CIE3	Clear CIE2	Clear CIE1	Clear CIE0
57H		0	0	Set CIE5	Set CIE4	Set CIE3	Set CIE2	Set CIE1	Set CIE0
56H	CnIE (R)	0	0	CIE5	CIE4	CIE3	CIE2	CIE1	CIE0
57H		0	0	0	0	0	0	0	0
58H	CnINTS (W)	0	0	Clear CINTS5	Clear CINTS4	Clear CINTS3	Clear CINTS2	Clear CINTS1	Clear CINTS0
59H		0	0	0	0	0	0	0	0
58H	CnINTS (R)	0	0	CINTS5	CINTS4	CINTS3	CINTS2	CINTS1	CINTS0
59H		0	0	0	0	0	0	0	0
5AH	CnBRP	TQPRS7 to TQPRS0							
5CH	CnBTR	0	0	0	0	TSEG13 to TSEG10			
5DH		0	0	SJW1, SJW0		0	TSEG22 to TSEG20		
5EH	CnLIPT	LIPT7 to LIPT0							
60H	CnRGPT (W)	0	0	0	0	0	0	0	Clear ROVF
61H		0	0	0	0	0	0	0	0
60H	CnRGPT (R)	0	0	0	0	0	0	RHPM	ROVF
61H		RGPT7 to RGPT0							
F62H	CnLOPT	LOPT7 to LOPT0							
64H	CnTGPT (W)	0	0	0	0	0	0	0	Clear TOVF
65H		0	0	0	0	0	0	0	0
64H	CnTGPT (R)	0	0	0	0	0	0	THPM	TOVF
65H		TGPT7 to TGPT0							
66H	CnTS (W)	0	0	0	0	0	Clear TSLOCK	Clear TSSEL	Clear TSEN
67H		0	0	0	0	0	Set TSLOCK	Set TSSEL	Set TSEN
66H	CnTS (R)	0	0	0	0	0	TSLOCK	TSSEL	TSEN
67H		0	0	0	0	0	0	0	0
68H to FFH	-	Access prohibited (reserved for future use)							

[a] Base address: <CnRBaseAddr>

Table 19-25 Message buffer register bit configuration

Address offset[a]	Symbol	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
0H	CnMDATA01m	Message data (byte 0)							
1H		Message data (byte 1)							
0H	CnMDATA0m	Message data (byte 0)							
1H	CnMDATA1m	Message data (byte 1)							
2H	CnMDATA23m	Message data (byte 2)							
3H		Message data (byte 3)							
2H	CnMDATA2m	Message data (byte 2)							
3H	CnMDATA3m	Message data (byte 3)							
4H	CnMDATA45m	Message data (byte 4)							
5H		Message data (byte 5)							
4H	CnMDATA4m	Message data (byte 4)							
5H	CnMDATA5m	Message data (byte 5)							
6H	CnMDATA67m	Message data (byte 6)							
7H		Message data (byte 7)							
6H	CnMDATA6m	Message data (byte 6)							
7H	CnMDATA7m	Message data (byte 7)							
8H	CnMDLCm	0				MDLC3	MDLC2	MDLC1	MDLC0
9H	CnMCONFm	OWS	RTR	MT2	MT1	MT0	0	0	MA0
AH	CnMIDLm	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
BH		ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CH	CnMIDHm	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
DH		IDE	0	0	ID28	ID27	ID26	ID25	ID24
EH	CnMCTRLm (W)	0	0	0	Clear MOW	Clear IE	Clear DN	Clear TRQ	Clear RDY
FH		0	0	0	0	Set IE	0	Set TRQ	Set RDY
EH	CnMCTRLm (R)	0	0	0	MOW	IE	DN	TRQ	RDY
FH		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 668*.

19.6 Control Registers

(1) CnGMCTRL - CANn global control register

The CnGMCTRL register is used to control the operation of the CAN module.

After reset: 0000H R/W Address: <CnRBaseAddr> + 000_H

(a) Read

	15	14	13	12	11	10	9	8
CnGMCTRL	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 MBON bit is cleared (to 0) when the CAN module enters CAN sleep mode/ CAN stop mode or GOM bit is cleared (to 0).
MBON bit is set (to 1) when the CAN sleep mode/the CAN stop mode is released or 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 immediately 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 bit is cleared only in the initialization mode.

(b) Write

	15	14	13	12	11	10	9	8
CnGMCTRL	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.

(2) CnGMCS - CANn global clock selection register

The CnGMCS register is used to select the CAN module system clock.

After reset: 0FH R/W Address: <CnRBaseAddr> + 002_H

	7	6	5	4	3	2	1	0
CnGMCS	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)

(3) CnGMABT - CANn global automatic block transmission control register

The CnGMABT register is used to control the automatic block transmission (ABT) operation.

After reset: 0000H R/W Address: <CnRBaseAddr> + 006_H

(a) Read

	15	14	13	12	11	10	9	8
CnGMABT	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 Do not set the ABTTRG bit 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.

(b) Write

	15	14	13	12	11	10	9	8
CnGMABT	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 (00H).

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.

After reset: 00H R/W Address: <CnRBaseAddr> + 008_H

	7	6	5	4	3	2	1	0
CnGMABTD	0	0	0	0	ABTD3	ABTD2	ABTD1	ABTD0

ABTD3	ABTD2	ABTD1	ABTD0	Data frame interval during automatic block transmission (Unit: Data bit time (DBT))
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

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 by masking part of the identifier (ID) of a message and invalidating the ID of the masked part.

(a) CANn module mask 1 register (CnMASK1L, CnMASK1H)

After reset: Undefined R/W Address: CnMASK1L <CnRBaseAddr> + 040$_H$
CnMASK1H <CnRBaseAddr> + 042$_H$

	15	14	13	12	11	10	9	8
CnMASK1L	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
	7	6	5	4	3	2	1	0
	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
	15	14	13	12	11	10	9	8
CnMASK1H	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)

After reset: Undefined R/W Address: CnMASK2L <CnRBaseAddr> + 044$_H$
CnMASK2H <CnRBaseAddr> + 046$_H$

	15	14	13	12	11	10	9	8
CnMASK2L	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
	7	6	5	4	3	2	1	0
	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
	15	14	13	12	11	10	9	8
CnMASK2H	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)

After reset: Undefined R/W Address: CnMASK3L <CnRBaseAddr> + 048_H
CnMASK3H <CnRBaseAddr> + $04A_H$

	15	14	13	12	11	10	9	8
CnMASK3L	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
	7	6	5	4	3	2	1	0
	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
	15	14	13	12	11	10	9	8
CnMASK3H	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)

After reset: Undefined R/W Address: CnMASK4L <CnRBaseAddr> + $04C_H$
CnMASK4H <CnRBaseAddr> + $04E_H$

	15	14	13	12	11	10	9	8
CnMASK4L	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
	7	6	5	4	3	2	1	0
	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
	15	14	13	12	11	10	9	8
CnMASK4H	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.

After reset: 0000H R/W Address: CnCTRL <CnRBaseAddr> + 050_H

(a) Read

	15	14	13	12	11	10	9	8
CnCTRL	0	0	0	0	0	0	RSTAT	TSTAT
	7	6	5	4	3	2	1	0
	CCERC	AL	VALID	PSMODE 1	PSMODE 0	OPMODE 2	OPMODE 1	OPMODE 0

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
   - The first bit of an error flag is detected during a transmit frame
2. The TSTAT bit is cleared to 0 under the following conditions (timing)
   - During transition to bus-off status
   - 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 status. 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.

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

1. The AL bit is valid only in the single-shot mode.
2. The AL bit is read-only in the CAN sleep mode or CAN stop 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 reception mode, the VALID bit is not set to 1 before the transmitting node enters the error passive status.
4. The VALID bit is read-only in the CAN sleep mode or CAN stop mode.
5. 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 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.

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

Note The OPMODE0 to OPMODE2 bits are read-only in the CAN sleep mode or CAN stop mode.

(b) Write

	15	14	13	12	11	10	9	8
CnCTRL	Set CCERC	Set AL	0	Set PSMODE 1	Set PSMODE 0	Set OPMODE 2	Set OPMODE 1	Set OPMODE 0
	7	6	5	4	3	2	1	0
	0	Clear AL	Clear VALID	Clear PSMODE 1	Clear PSMODE 0	Clear OPMODE 2	Clear OPMODE 1	Clear OPMODE 0

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	PSMODE 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.

After reset: 00H R/W Address: CnLEC <CnRBaseAddr> + 052_H

	7	6	5	4	3	2	1	0
CnLEC	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 00H 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.

After reset: 00H R Address: CnINFO <CnRBaseAddr> + 053_H

	7	6	5	4	3	2	1	0
CnINFO	0	0	0	BOFF	TECS1	TECS0	RECS1	RECS0

BOFF	Bus-off status bit
0	Not bus-off status (transmit error counter ≤ 255). (The value of the transmit counter is less than 256.)
1	Bus-off status (transmit error counter > 255). (The value of the transmit counter is 256 or more.)

TECS1	TECS0	Transmission error counter status bit
0	0	The value of the transmission error counter is less than that of the warning level (< 96).
0	1	The value of the transmission error counter is in the range of the warning level (96 to 127).
1	0	Undefined
1	1	The value of the transmission error counter is in the range of the error passive or bus-off status (≥ 128).

RECS1	RECS0	Reception error counter status bit
0	0	The value of the reception error counter is less than that of the warning level (< 96).
0	1	The value of the reception error counter is in the range of the warning level (96 to 127).
1	0	Undefined
1	1	The value of the reception error counter is in the error passive range (≥ 128).

(9) CnERC - CANn module error counter register

The CnERC register indicates the count value of the transmission/reception error counter.

After reset: 0000H R Address: CnERC <CnRBaseAddr> + 054H

	15	14	13	12	11	10	9	8
CnERC	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 value of the reception error counter is not error passive (< 128)
1	The value of 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 The REC6 to REC0 bits of the reception error counter are invalid in the reception error passive status (CnINFO.RECS1, CnINFO.RECS0 bit = 11B).

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 status (CnINFO.BOFF bit = 1).

(10) CnIE - CANn module interrupt enable register

The CnIE register is used to enable or disable the interrupts of the CAN module.

After reset: 0000H R/W Address: CnIE <CnRBaseAddr> + 056_H

(a) Read

	15	14	13	12	11	10	9	8
CnIE	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) Write

	15	14	13	12	11	10	9	8
CnIE	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.

After reset: 0000H R/W Address: CnINTS <CnRBaseAddr> + 058$_H$

(a) Read

	15	14	13	12	11	10	9	8
CnINTS	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^{Note}
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

Note 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) Write

	15	14	13	12	11	10	9	8
CnINTS	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.

(12) CnBRP - CANn module bit rate prescaler register

The CnBRP register is used to select the CAN protocol layer base clock (f_{TQ}). The communication baud rate is set to the CnBTR register.

After reset: FFH R/W Address: CnBRP <CnRBaseAddr> + 05A$_H$

	7	6	5	4	3	2	1	0
CnBRP	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)

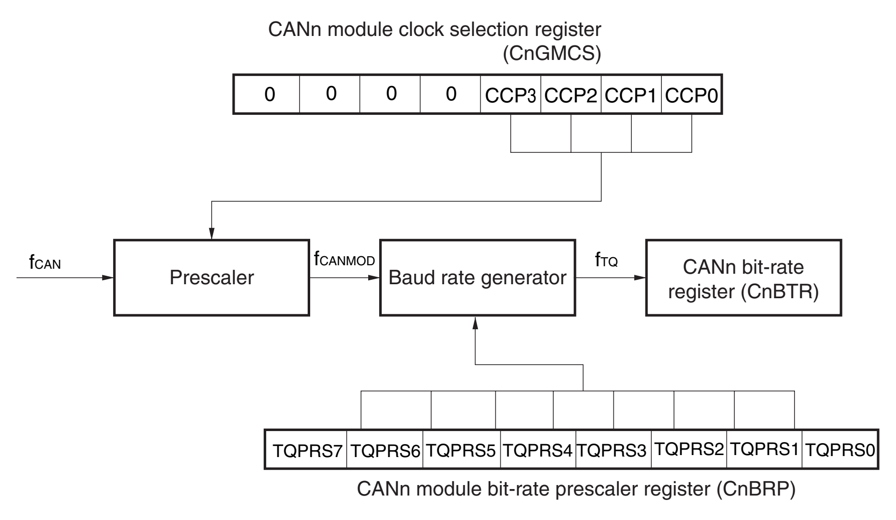


Figure 19-23 **CAN module clock**

Note 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.

After reset: 370FH R/W Address: CnBTR <CnRBaseAddr> + 05C$_H$

	15	14	13	12	11	10	9	8
CnBTR	0	0	SJW1	SJW0	0	TSEG22	TSEG21	TSEG20
	7	6	5	4	3	2	1	0
	0	0	0	0	TSEG13	TSEG12	TSEG11	TSEG10

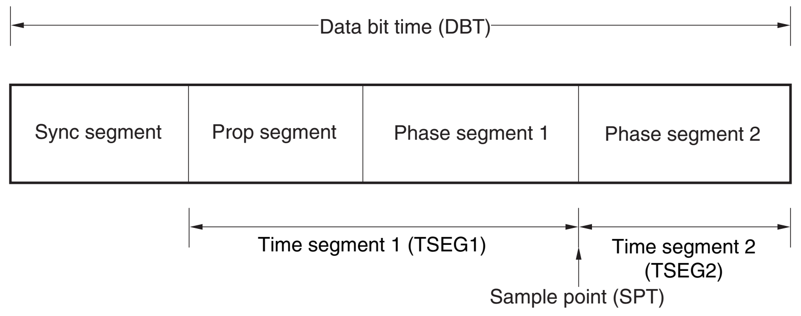


Figure 19-24 Data bit time

SJW1	SJW0	Length of synchronization jump width
0	0	1TQ
0	1	2TQ
1	0	3TQ
1	1	4TQ (default value)

TSEG22	TSEG21	TSEG20	Length of time segment 2
0	0	0	1TQ
0	0	1	2TQ
0	1	0	3TQ
0	1	1	4TQ
1	0	0	5TQ
1	0	1	6TQ
1	1	0	7TQ
1	1	1	8TQ (default value)

TSEG13	TSEG12	TSEG11	TSEG10	Length of time segment 1
0	0	0	0	Setting prohibited
0	0	0	1	2TQ$^{\text{Note}}$
0	0	1	0	3TQ$^{\text{Note}}$
0	0	1	1	4TQ
0	1	0	0	5TQ
0	1	0	1	6TQ
0	1	1	0	7TQ
0	1	1	1	8TQ
1	0	0	0	9TQ
1	0	0	1	10TQ
1	0	1	0	11TQ
1	0	1	1	12TQ
1	1	0	0	13TQ
1	1	0	1	14TQ
1	1	1	0	15TQ
1	1	1	1	16TQ (default value)

Note

1. This setting must not be made when the CnBRP register = 00H.
2. TQ = 1/fTQ (fTQ: 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.

After reset: Undefined R Address: CnLIPT <CnRBaseAddr> + $05E_H$

	7	6	5	4	3	2	1	0
CnLIPT	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 CnRGPT.RHPM 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 CnLIPT register is undefined.

(15) CnRGPT - CANn module receive history list register

The CnRGPT register is used to read the receive history list.

After reset: xx02H R/W Address: CnRGPT <CnRBaseAddr> + 060_H

(a) Read

	15	14	13	12	11	10	9	8
CnRGPT	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[Note 1]	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.

ROVF	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	All the message buffer numbers that are recorded are preserved except the message buffer number recorded last[Note 2]. The number of the message buffer in which a new data frame or remote frame has been received and stored is recorded to the receive history list, by overwriting the message buffer number that was recorded last (the receive history list does not have a vacant element).

Note

1. The read value of the RGPT0 to RGPT7 bits is invalid when the RHPM bit = 1.
2. If no new data frame or remote frame is received and stored in a message buffer after the ROVF bit has been set, the message buffer number last recorded to the receive history list is preserved.

(b) Write

	15	14	13	12	11	10	9	8
CnRGPT	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.

After reset: Undefined R Address: CnLOPT <CnRBaseAddr> + 062_H

	7	6	5	4	3	2	1	0
CnLOPT	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.

After reset: xx02H R/W Address: CnTGPT <CnRBaseAddr> + 064_H

(a) Read

	15	14	13	12	11	10	9	8
CnTGPT	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[Note 1]	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.

TOVF	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	All the message buffer numbers that are recorded are preserved except the message buffer number recorded last[Note 2]. The number of the message buffer to which a new data frame or remote frame has been transmitted is recorded to the transmit history list, by overwriting the message buffer number that was recorded last (the transmit history list does not have a vacant element).

Note

1. The read value of the TGPT0 to TGPT7 bits is invalid when the THPM bit = 1.
2. If no new data frame or remote frame is transmitted after the TOVF bit has been set, the message buffer number last recorded to the transmit history list is preserved.
3. Transmission from message buffers 0 to 7 is not recorded to the transmit history list in the normal operation mode with ABT.

(b) Write

	15	14	13	12	11	10	9	8
CnTGPT	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.

After reset: 0000H R/W Address: CnTS <CnRBaseAddr> + 066_H

(a) Read

	15	14	13	12	11	10	9	8
CnTS	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
	0	0	0	0	0	TSLOCK	TSSEL	TSEN

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 output signal is locked when a data frame has been correctly received to message buffer 0[Note].

Note The TSEN bit is automatically cleared to 0.

TSSEL	Time stamp capture event selection bit
0	The time capture event is SOF.
1	The time stamp capture event is the last bit of EOF.

TSEN	TSOUT operation setting bit
0	TSOUT toggle operation is disabled.
1	TSOUT toggle operation is enabled.

(b) Write

	15	14	13	12	11	10	9	8
CnTS	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

Note The time stamp function must not be used when the CAN module is in the normal operation mode with ABT.

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 - CANn message data byte register (x = 0 to 7)

The CnMDATAxm register is used to store the data of a transmit/receive message.

After reset: Undefined R/W Address: refer to *“CAN registers overview“ on page 668*

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	MDATA10

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

	15	14	13	12	11	10	9	8
CnMDATA45m	MDATA4515	MDATA4514	MDATA4513	MDATA4512	MDATA4511	MDATA4510	MDATA459	MDATA458
	7	6	5	4	3	2	1	0
	MDATA457	MDATA456	MDATA455	MDATA454	MDATA453	MDATA452	MDATA451	MDATA450

	7	6	5	4	3	2	1	0
CnMDATA4m	MDATA47	MDATA46	MDATA45	MDATA44	MDATA43	MDATA42	MDATA41	MDATA40

	7	6	5	4	3	2	1	0
CnMDATA5m	MDATA57	MDATA56	MDATA55	MDATA54	MDATA53	MDATA52	MDATA51	MDATA50

	15	14	13	12	11	10	9	8
CnMDATA67m	MDATA6715	MDATA6714	MDATA6713	MDATA6712	MDATA6711	MDATA6710	MDATA679	MDATA678
	7	6	5	4	3	2	1	0
	MDATA677	MDATA676	MDATA675	MDATA674	MDATA673	MDATA672	MDATA671	MDATA670

	7	6	5	4	3	2	1	0
CnMDATA6m	MDATA67	MDATA66	MDATA65	MDATA64	MDATA63	MDATA62	MDATA61	MDATA60

	7	6	5	4	3	2	1	0
CnMDATA7m	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.

After reset: 0000xxxxB R/W Address: refer to *“CAN registers overview“ on page 668*

	7	6	5	4	3	2	1	0
CnMDLCm	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 0000B.
2. Receive data is stored in as many CnMDATAxm register as the number of bytes (however, the upper limit is 8) corresponding to DLC. 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.

After reset: Undefined R/W Address: refer to *“CAN registers overview“ on page 668*

	7	6	5	4	3	2	1	0
CnMCONFm	OWS	RTR	MT2	MT1	MT0	0	0	MA0

OWS	Overwrite control bit
0	The message buffer[Note] that has already received a data frame 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 is overwritten by a newly received data frame.

Note
1. 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.
2. A remote frame is received and stored, regardless of the setting of the OWS and DN bits. A remote frame that satisfies the other conditions (ID matches, the RTR bit = 0, the CnMCTRLm.TRQ bit = 0) is always received and stored in the corresponding message buffer (interrupt generated, DN flag set, the CnMDLCm.MDLC0 to CnMDLCm.MDLC3 bits updated, and recorded to the receive history list).

RTR	Remote frame request bit[Note]
0	Transmit a data frame.
1	Transmit a remote frame.

Note 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).

After reset: Undefined R/W Address: refer to *“CAN registers overview“ on page 668*

	15	14	13	12	11	10	9	8
CnMIDLm	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
	7	6	5	4	3	2	1	0
	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

	15	14	13	12	11	10	9	8
CnMIDHm	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)[Note]
1	Extended format mode (ID28 to ID0: 29 bits)

Note 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 Be sure to write 0 to bits 14 and 13 of the CnMIDHm register.

(23) CnMCTRLm - CANn message control register m

The CnMCTRLm register is used to control the operation of the message buffer.

After reset: 00x000000
00000000B R/W Address: refer to *"CAN registers overview" on page 668*

(a) Read

	15	14	13	12	11	10	9	8
CnMCTRLm	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^{Note}	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).

Note The MUC bit is undefined until the first reception and storage is performed.

MOW	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.

Note The MOW bit is not set to 1 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.

Caution Do not clear the RDY bit (0) during message transmission.

(b) Write

	15	14	13	12	11	10	9	8
CnMCTRLm	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.

Caution Do not set the DN bit to 1 by software. Be sure to write 0 to bit 10.

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.

19.7 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 19-25* 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 19-26*). *Figure 19-25* shows how the values of set bits or clear bits relate to set/clear/no change operations in the corresponding register.

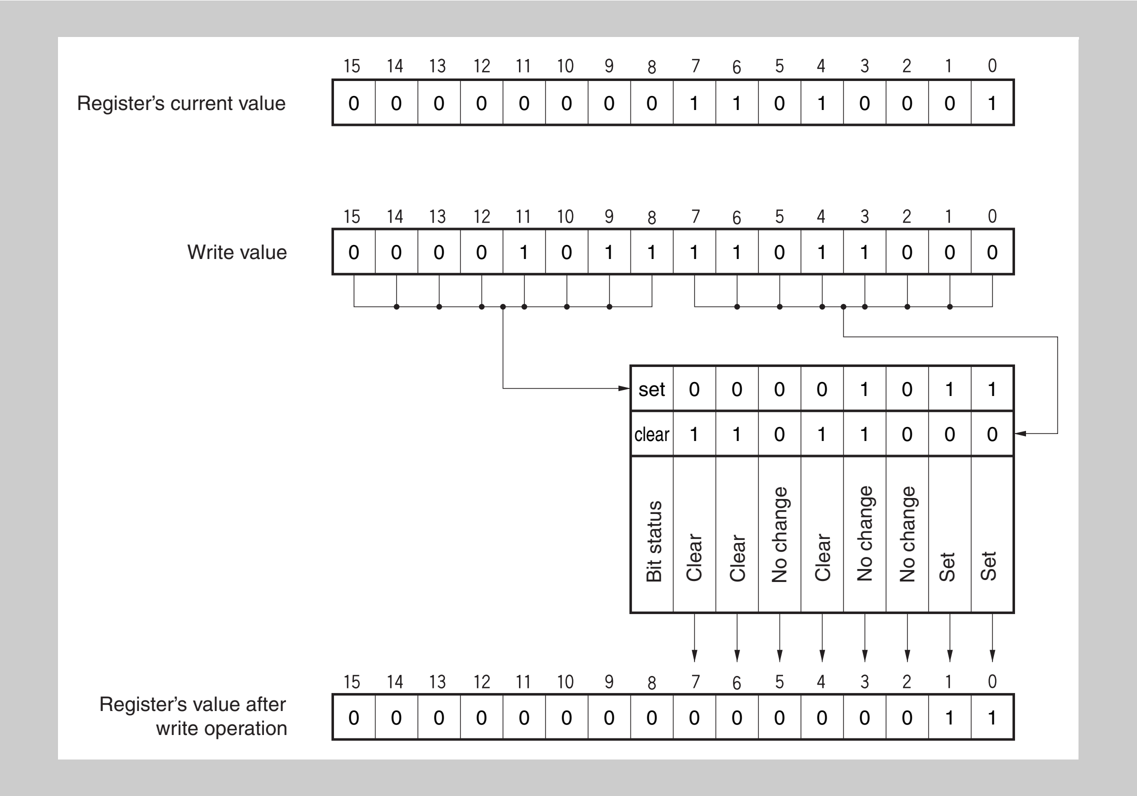


Figure 19-25 Example of bit setting/clearing operations

(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

19.8 CAN Controller Initialization

19.8.1 Initialization of CAN module

Before CAN module operation is enabled, the CAN module system clock needs to be determined by setting the CnGMCS.CCP0 to CnGMCS.CCP3 bits 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 CnGMCTRL.GOM bit.

For the procedure of initializing the CAN module, *“Operation of CAN Controller“ on page 745*.

19.8.2 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 CnMCTRLm.RDY, CnMCTRLm.TRQ, and CnMCTRLm.DN bits to 0.
- Clear the CnMCONFm.MA0 bit to 0.

19.8.3 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 19-37*.

(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 in normal operation mode“ on page 727* and *“Transmission abort in normal operation mode with automatic block transmission (ABT)“ on page 727*). 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.

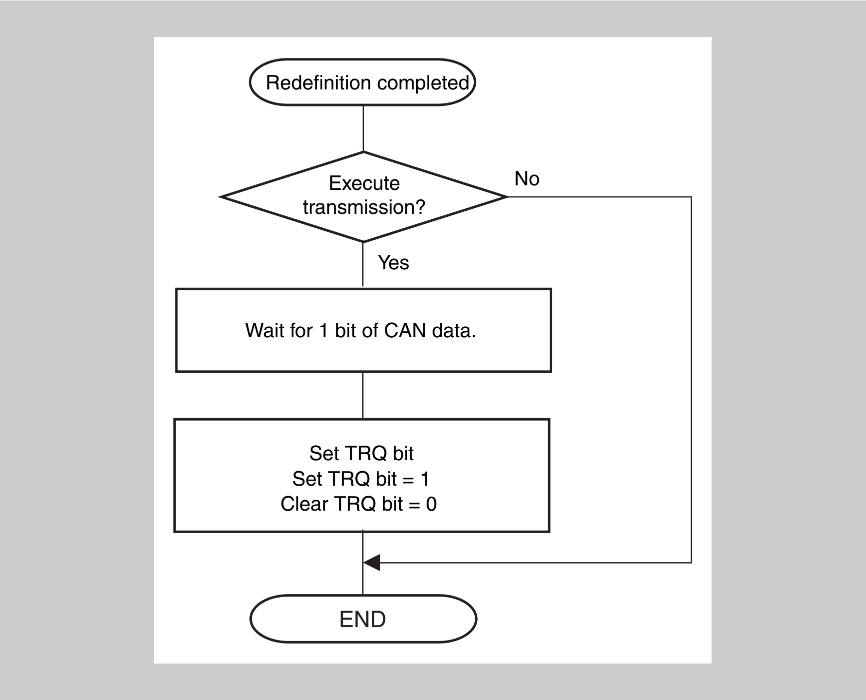


Figure 19-26 Setting transmission request (TRQ) to transmit message buffer after redefinition

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 19-37 on page 748* 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 19-26 on page 712* is not observed, a message with an ID not having the highest priority may be transmitted after redefinition.

19.9 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

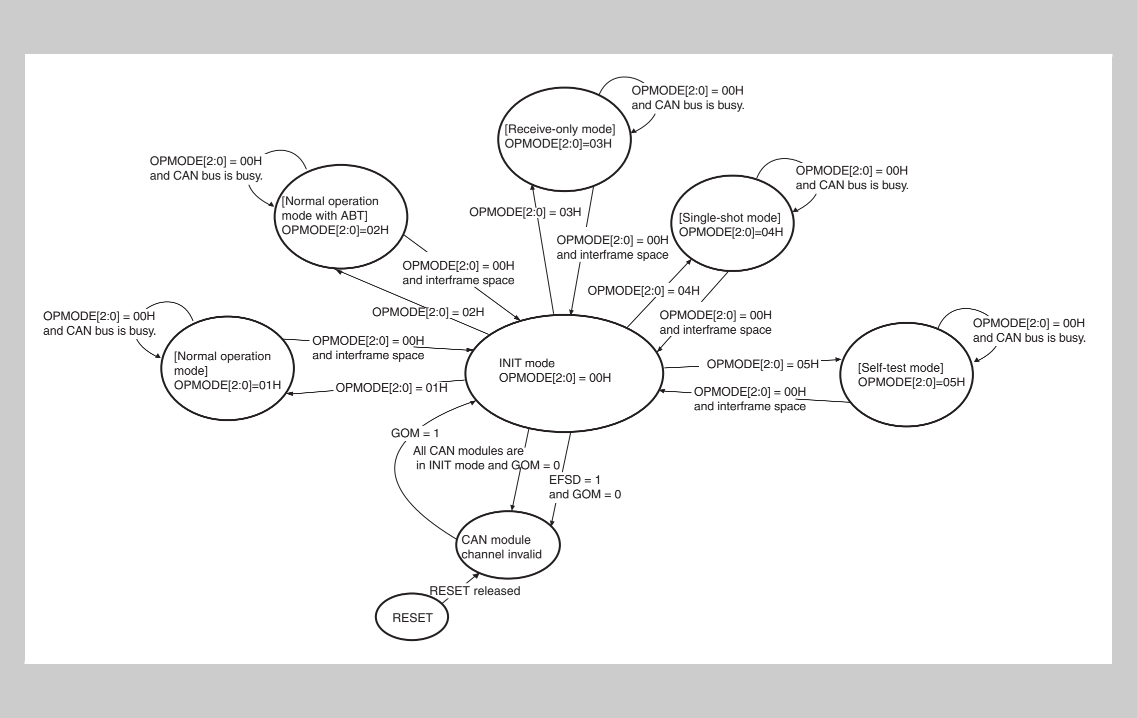


Figure 19-27 Transition to operation modes

The transition from the initialization mode to an operation mode is controlled by the CnCTRL.OPMODE2 to CnCTRL.OPMODE0 bits.

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 OPMODE2 to OPMODE0 bits are changed to 00H). After issuing a request to change the mode to the initialization mode, read the OPMODE2 to OPMODE0 bits until their values become 000B to confirm that the module has entered the initialization mode (see *Figure 19-35 on page 746*).

19.9.1 Resetting error counter CNERC of CAN module

If it is necessary to reset the CnERC and CnINFO registers when re-initialization or forced recovery from the bus-off status is made, set the CnCTRL.CCERC bit to 1 in the initialization mode. When this bit is set to 1, the CnERC and CnINFO registers are cleared to their default values.

19.10 Message Reception

19.10.1 Message reception

In all the operation modes, when a message is received, a message buffer that is to store the message is searched from all the message buffers satisfying the following conditions.

- Used as a message buffer
  (CnMCONFm.MA0 bit is set to 1.)
- Set as a receive message buffer
  (CnMCONFm.MT2 to CnMCONFm.MT0 bits are set to 001B, 010B, 011B, 100B, or 101B.)
- Ready for reception
  (CnMCTRLm.RDY bit 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 message is always stored in the unmasked receive message buffer even if this unmasked receive buffer has already received a message earlier.

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

19.10.2 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 CnRGPT.RHPM bit (receive history list pointer match) 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 CnRGPT.ROVF bit (receive history list overflow) 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 new message. After the ROVF bit has been set (1), therefore, the recorded message buffer numbers in the RHL do not completely reflect the chronological order.

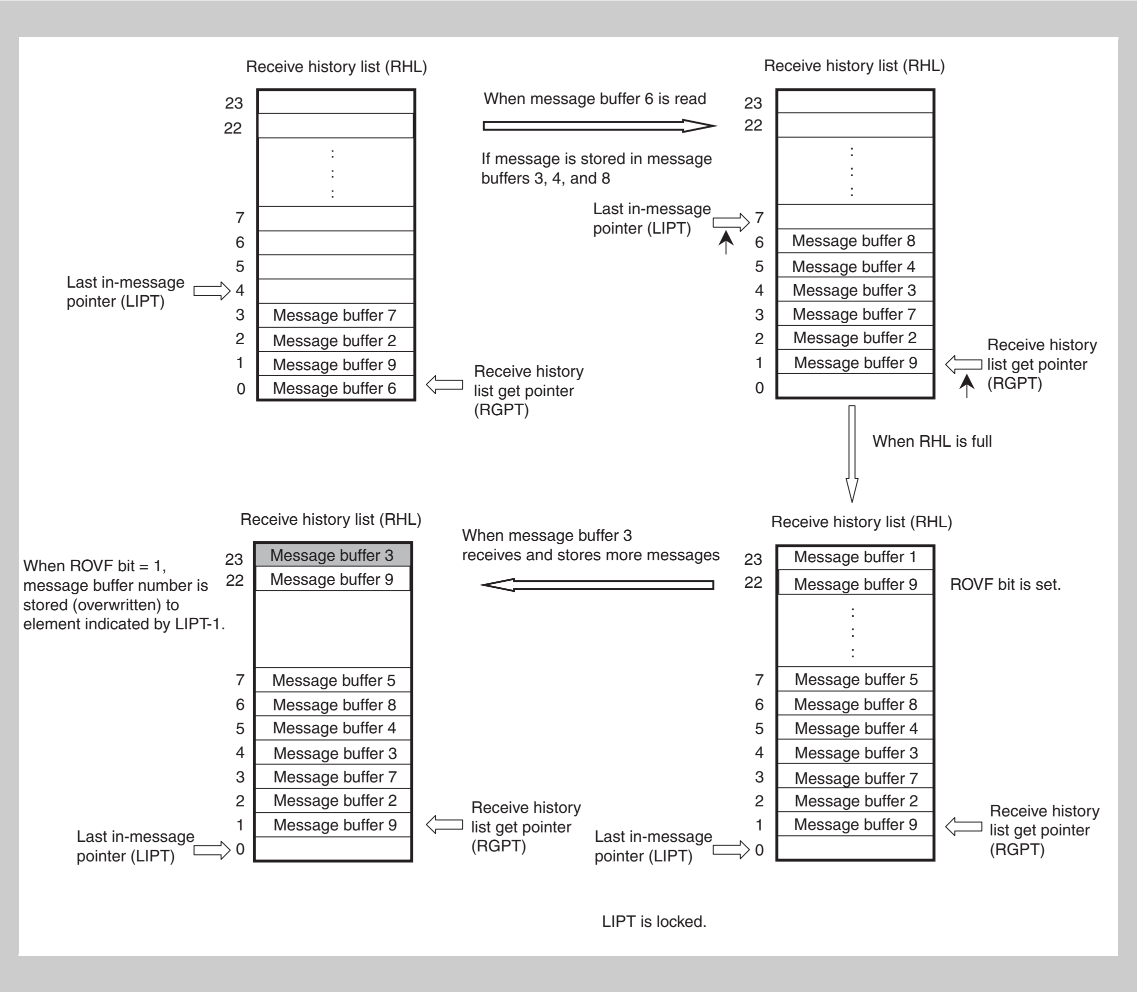


Figure 19-28 Receive history list

19.10.3 Mask function

It can be defined whether masking of the identifier that is set to a message buffer is linked with another message buffer.

By using the mask function, the identifier of a message received from the CAN bus can be compared with the identifier set to a message buffer in advance. Regardless of whether the masked ID is set to 0 or 1, the received message can be stored in the defined message buffer.

While the mask function is in effect, an identifier bit that is defined to be 1 by a mask in the received message is not compared with the corresponding identifier bit in the message buffer.

However, this comparison is performed for any bit whose value is defined as 0 by the mask.

For example, let us assume that all messages that have a standard-format ID, in which bits ID27 to ID25 are 0 and bits ID24 and ID22 are 1, are to be stored in message buffer 14. The procedure for this example is shown below.

<1> Identifier to be stored in message buffer

ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18
x	0	0	0	1	x	1	x	x	x	x

Note x = don’t care

<2> Identifier to be configured in message buffer 14 (example)
(Using C0MIDL14 and C0MIDH14 registers)

ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18
x	0	0	0	1	x	1	x	x	x	x
ID17	ID16	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
x	x	x	x	x	x	x	x	x	x	x
ID6	ID5	ID4	ID3	ID2	ID1	ID0				
x	x	x	x	x	x	x				

Note

1. ID with the ID27 to ID25 bits cleared to 0 and the ID24 and ID22 bits set to 1 is registered (initialized) to message buffer 14.
2. Message buffer 14 is set as a standard format identifier that is linked to mask 1 (CnMCONF14.MT2 to CnMCONF14.MT0 bits are set to 010B).

<3> Mask setting for CAN module 1 (mask 1) (Example)
(Using CAN1 address mask 1 registers L and H (C1MASKL1 and C1MASKH1))

CMID28	CMID27	CMID26	CMID25	CMID24	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18
1	0	0	0	0	1	0	1	1	1	1
CMID17	CMID16	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8	CMID7
1	1	1	1	1	1	1	1	1	1	1
CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0				
1	1	1	1	1	1	1				

1: Not compared (masked)
0: Compared

The CMID27 to CMID24 and CMID22 bits are cleared to 0, and the CMID28, CMID23, and CMID21 to CMID0 bits are set to 1.

19.10.4 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.

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 CnMCTRLm.IE bit 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.

19.10.5 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
  (CnMCONFm.MA0 bit set to 1.)
- Set as a transmit message buffer
  (CnMCONFm.MT2 to CnMCONFm.MT0 bits set to 000B)
- Ready for reception
  (CnMCTRLm.RDY bit set to 1.)
- Set to transmit message
  (CnMCONFm.RTR bit is cleared to 0.)
- Transmission request is not set.
  (CnMCTRLm.TRQ bit is set to 1.)

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 CnMDLCm.DLC3 to CnMDLCm.DLC0 bits store the received DLC value.
- The CnMDATA0m to CnMDATA7m registers in the data area are not updated (data before reception is saved).
- The CnMCTRLm.DN bit is set to 1.
- The CnINTS.CINTS1 bit is set to 1 (if the CnMCTRLm.IE bit of the message buffer that receives and stores the frame is set to 1).
- The receive completion interrupt (INTCnRE) is output (if the IE bit of the message buffer that receives and stores the frame is set to 1 and if the CnIE.CIE1 bit 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 CnMCONFm.OWS bit of the message buffer and the DN bit are not affected.
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.

19.11 Message Transmission

19.11.1 Message transmission

In all the operation modes, if the CnMCTRLm.TRQ bit is set to 1 in a message buffer that satisfies the following conditions, the message buffer that is to transmit a message is searched.

- Used as a message buffer
  (CnMCONFm.MA0 bit set to 1.)
- Set as a transmit message buffer
  (CnMCONFm.MT2 to CnMCONFm.MT0 bits set to 000B.)
- Ready for transmission
  (CnMCTRLm.RDY bit is set to 1.)

The CAN system is a multi-master communication system. In a system like this, the priority of message transmission is determined based on message identifiers (IDs). To facilitate transmission processing by software when there are several messages awaiting transmission, the CAN module uses hardware to check the ID of the message with the highest priority and automatically identifies that message. This eliminates the need for software-based priority control.

Transmission priority is controlled by the identifier (ID).

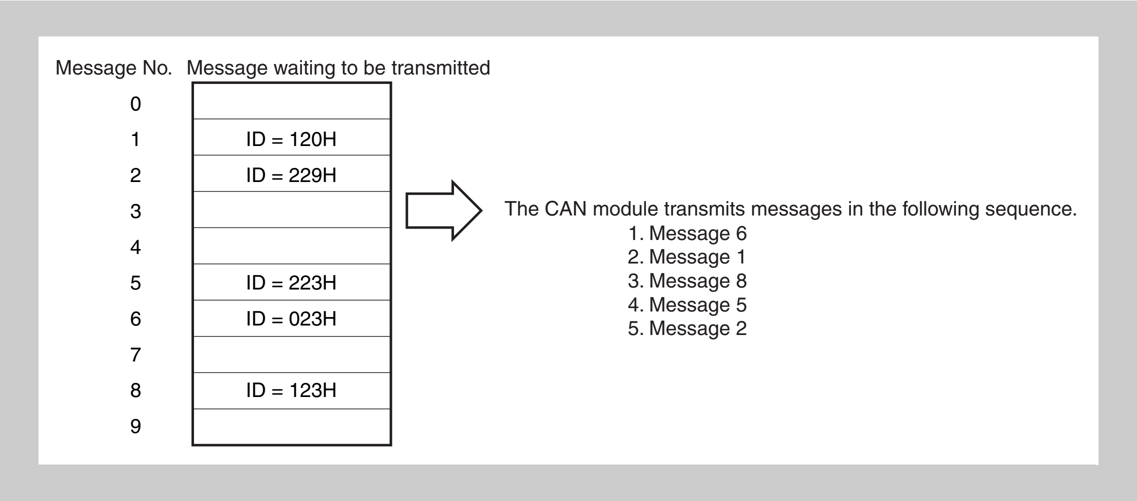


Figure 19-29 Message processing example

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. 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 (CnMCONFm.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 (CnMIDHm.IDE bit is cleared to 0) has a higher priority than a message frame with an extended ID.
4	Value of lower 18 bits of ID [ID17 to ID0]:	If one or more transmission-pending extended ID message frame has equal values in the first 11 bits of the ID and the same frame type (equal RTR bit values), the message frame with the lowest value in the lower 18 bits of its extended ID is transmitted first.
5 (low)	Message buffer number	If two or more message buffers request transmission of message frames with the same ID, the message from the message buffer with the lowest message buffer number is transmitted first.

Note If the automatic block transmission request bit CnGMABT.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 TRQ bit is set to 1 for this buffer and for the message buffers that do not belong to the ABT message buffer group, a conflict occurs. When messages are successively transmitted from the automatic block transmission area (message buffers 0 to 7), therefore, the priority of the transmission ID is not searched, and the messages are transmitted sequentially, starting from the buffer with the lowest number. However, the priority among automatic block transmission messages and message buffers other than those in the automatic block transmission area is in compliance with the above rule.

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 INTRRX1 is output (if the CnIE.CIE0 bit is set to 1 and if the interrupt enable bit (IE) of the corresponding transmit message buffer is set to 1).

19.11.2 Transmit history list function

The transmit history list (THL) function records in the transmit history list the number of the transmit message buffer in which each data frame or remote frame was received and stored. 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 CnTGPT.THPM bit (transmit history list pointer match) 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 number of the message buffer that received and stored the new message. After the TOVF bit has been set (1), therefore, the recorded message buffer numbers in the THL do not completely reflect the chronological order.

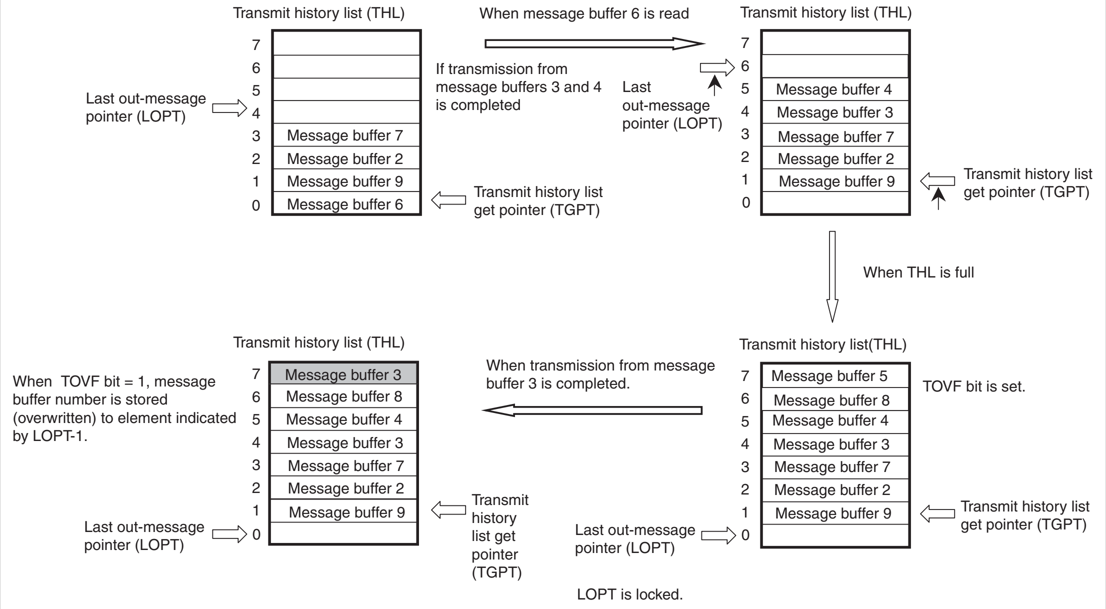


Figure 19-30 Transmit history list

19.11.3 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 CnCTRL.OPMODE2 to CnCTRL.OPMODE0 bits to 010B, “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 CnMCONFm.MA0 bit (1) in all the message buffers used for ABT, and define all the buffers as transmit message buffers by setting the CnMCONFm.MA2 to CnMCONFm.MA0 bits to 000B. Be sure to set the same ID for the message buffers for ATB 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 CnMCTRLm.RDY bit needs to be set (1). In the ABT mode, the CnMCTRLm.TRQ bit does not have to be manipulated by software.

After the data for the ABT message buffers has been prepared, set the CnGMABT.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 bit) 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.

During ABT, the priority of the transmission ID is not searched. The data of message buffers 0 to 7 is 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 CnGMABT.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 CnMCTRLm.IE bit 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 priority of the message to be transmitted 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. 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 = 00H), messages other than ABT messages are transmitted. At this time, transmission does not depend on 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 in the normal operation mode with ABT, the message may be transmitted after the time of one frame has elapsed (when CnGMABTD register = 00H).

19.11.4 Transmission abort process

(1) Transmission abort in normal operation mode

The user can clear the CnMCTRLm.TRQ bit 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 CnCTRL.TSTAT bit and the CnTGPT register, which indicate the transmission status on the CAN bus (for details, refer to the processing in *Figure 19-43 on page 754*).

(2) Transmission abort in normal operation mode with automatic block transmission (ABT)

To abort ABT that is already started, clear the CnGMABT.ABTTRG bit 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 19-44 on page 755*).

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^{Note}	Same message buffer in the ABT area
Cleared (0)	Next message buffer in the ABT area^{Note}	Next message buffer in the ABT area^{Note}

Note 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 CnMCTRLm.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.

19.11.5 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 CnMCONFm.RTR bit. Setting (1) the RTR bit sets remote frame transmission.

19.12 Power Saving Modes

19.12.1 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 01B to the CnCTRL.PSMODE1 and CnCTRL.PSMODE0 bits.

This transition request is only acknowledged only under the following conditions.

(i) 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

(ii) The CAN bus state is bus idle (the 4th bit in the interframe space is recessive)

Note: If the CAN bus is fixed to dominant, the request for transition to the CAN sleep mode is held pending.

(iii) No transmission request is pending

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 PSMODE1 and PSMODE0 bits remain 00B. When the module has entered the CAN sleep mode, the PSMODE1 and PSMODE0 bits are set to 01B.
- 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.

- If a CAN sleep mode request is pending waiting for the CAN bus state to become bus idle while the CAN module is in one of the operation modes, and if a request for transition to the initialization mode is made, the pending CAN sleep mode request becomes disabled, and only the initialization mode request is enabled (in this case, the CAN sleep mode request continues to be held pending).
- If the CAN sleep mode transition request is made while a initialization mode transition request is held pending waiting for completion of communication in one of the operation modes, the CAN sleep mode transition request is ignored and only the initialization mode transition request remains valid (in this case, the CAN sleep mode request continues to be held pending).

(2) Status in CAN sleep mode

The CAN module is in one of the following states 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 PSMODE1 and PSMODE0 bits, but nothing can be written to other CANn module registers or bits.
- The CANn module registers can be read, except for the CnLIPT, CnRGPT, CnLOPT, and CnTGPT registers.
- The CANn message buffer registers cannot be written or read.
- 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 00B to the PSMODE1 and PSMODE0 bits
- A falling edge at the CAN reception pin (CRXDn) (i.e. the CAN bus level shifts from recessive to dominant)

Caution If this falling edge is at the SOF of a receive frame, no receive operation, including returning ACK, is performed on that frame. No receive operation is performed on the subsequent frames either, unless the clock is supplied to the CAN macro.

After releasing the sleep mode, the CAN module returns to the operation mode from which the CAN sleep mode was requested and the PSMODE1 and PSMODE0 bits are reset to 00B. If the CAN sleep mode is released by a change in the CAN bus state, the CnINTS.CINTS5 bit is set to 1, regardless of the CnIE.CIE bit. 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.

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 CPU has to be released from sleep mode by software first before entering the initialization mode.

19.12.2 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 by writing 01B to the CnCTRL.PSMODE1 and CnCTRL.PSMODE0 bits 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 11B to the PSMODE1 and PSMODE0 bits.

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 PSMODE1 and PSMODE0 bits = 01B, 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 one of the following states 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 PSMODE1 and PSMODE0 bits, but nothing can be written to other CANn module registers or bits.
- The CANn module registers can be read, except for the CnLIPT, CnRGPT, CnLOPT, and CnTGPT registers.
- The CANn message buffer registers cannot be written or read.
- 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 01B to the PSMODE1 and PSMODE0 bits.

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.

19.12.3 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 (CnCTRL.PSMODE[1:0] = 01B). 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 CnINTS.CINTS5 is set to 1 and a wakeup interrupt (INTWUP) is generated, provided CnINTS5 is enabled by CnIE.CIE5 = 1.
- The CAN module is automatically released from CAN sleep mode (PSMODE = 00B) and returns to normal operation mode.
- The CPU, in response to INTWUP, can release its own power saving mode and return to normal operation mode.

  To further reduce the power consumption of the CPU, the internal clocks—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 CnINTS.CINTS5 = 1 and generate the wakeup interrupt (INTWUP) 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 INTWUP
  - 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.
- When clock supply is resumed, the CAN module is immediately released from CAN sleep mode and returns to normal operation mode (CnCTRL.PSMODE[1:0] = 00B).

19.13 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 19-26 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[Note]	CnIE	INTCnTRX	Message frame successfully transmitted from message buffer m
2	CINTS1	CnINTS	CIE1[Note]	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)

Note The CnMCTRL.IE bit (message buffer interrupt enable bit) 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).

19.14 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.

19.14.1 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 CnCTRL.VALID bit (1).

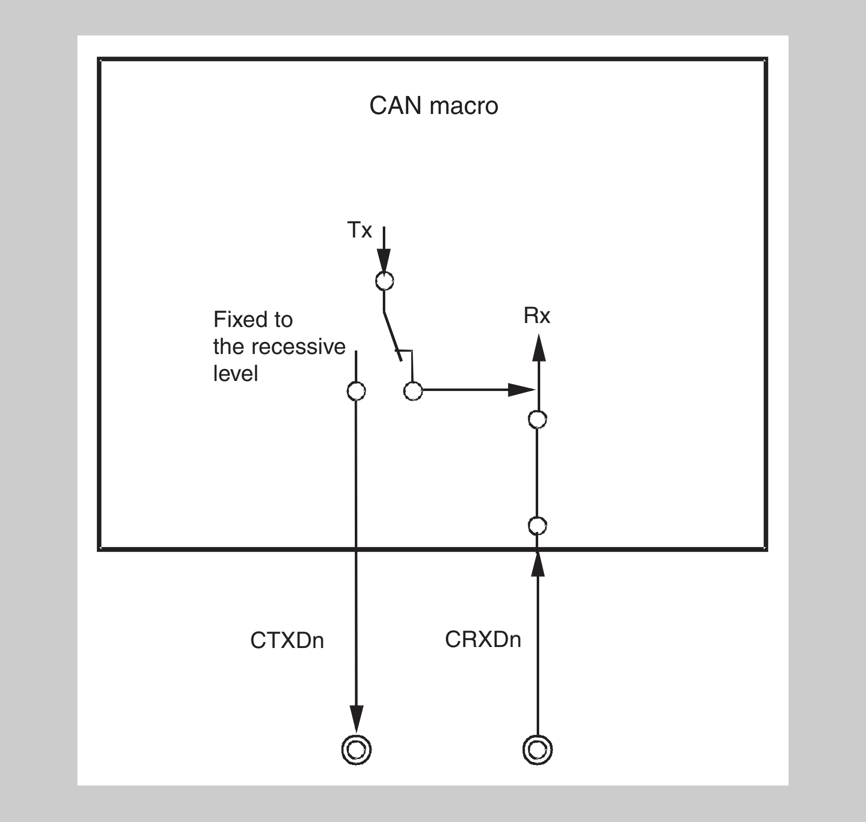


Figure 19-31 CAN module terminal connection in receive-only mode

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, 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). When the message frame is transmitted for the 17th time, 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.

19.14.2 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.)

The single-shot mode disables the re-transmission of an aborted message frame transmission according to the setting of the CnCTRL.AL bit. 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 CnMCTRLm.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 (AL bit = 0)
- Error occurrence while sending the message frame

The events arbitration loss and error occurrence can be distinguished by checking the CnINTS.CINTS4 and CnINTS.CINTS3 bits, and the type of the error can be identified by reading the CnLEC.LEC2 to CnLEC.LEC0 bits of the register.

Upon successful transmission of the message frame, the transmit completion interrupt the CINTS0 bit of the CnINTS register is set to 1. If the CnIE.CIE0 bit 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).

19.14.3 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.

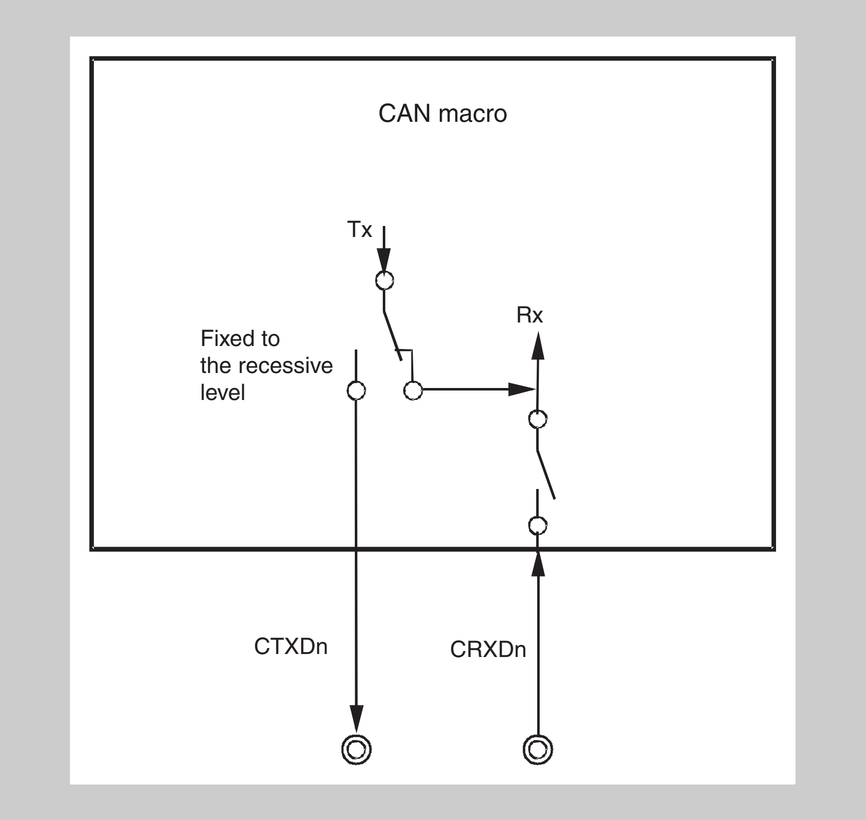


Figure 19-32 CAN module terminal connection in self-test mode

19.15 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 even 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.

19.15.1 Time stamp function

The CAN Controller supports the capturing of timer values triggered by successful reception of a data 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 CnTS.TSSEL bit.

- SOF event (start of frame) (TSSEL bit = 0)
- EOF event (last bit of end of frame) (TSSEL bit = 1)

The TSOUT signal is enabled by setting the CnTS.TSEN bit to 1.

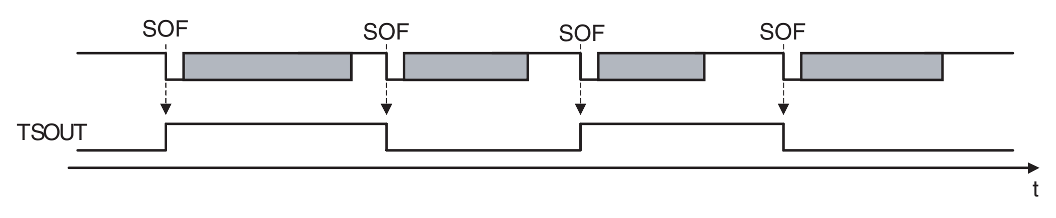


Figure 19-33 Timing diagram of capture signal TSOUT

The TSOUT signal toggles its level upon occurrence of the selected event during data frame reception (in *Figure 19-33*, 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 CnTS.TSLOCK bit. When the TSLOCK bit is cleared to 0, the TSOUT signal toggles upon occurrence of the selected event. If the TSLOCK bit 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 when a data frame is received and stored in message buffer 0. 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.

19.16 Baud Rate Settings

19.16.1 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] (CnBTR.TSEG13 to CnBTR.TSEG10 bits)
3. TSEG2[2:0] (CnBTR.TSEG22 to CnBTR.TSEG20 bits)

Table 19-27 shows the combinations of bit rates that satisfy the above conditions.

Table 19-27 Settable bit rate combinations (1/3)

Valid bit rate setting					CnBTR register setting value		Sampling point (unit %)
DBT length	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG13 to TSEG10	TSEG22 to TSEG20	
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 19-27 Settable bit rate combinations (2/3)

Valid bit rate setting					CnBTR register setting value		Sampling point (unit %)
DBT length	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG13 to TSEG10	TSEG22 to TSEG20	
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 19-27 Settable bit rate combinations (3/3)

Valid bit rate setting					CnBTR register setting value		Sampling point (unit %)
DBT length	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG13 to TSEG10	TSEG22 to TSEG20	
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[Note]	1	2	2	2	0011	001	71.4
7[Note]	1	4	1	1	0100	000	85.7
6[Note]	1	1	2	2	0010	001	66.7
6[Note]	1	3	1	1	0011	000	83.3
5[Note]	1	2	1	1	0010	000	80.0
4[Note]	1	1	1	1	0001	000	75.0

Note Setting with a DBT value of 7 or less is valid only when the value of the CnBRP register is other than 00H.

Caution The values in *Table 19-27* do not guarantee the operation of the network system. Thoroughly check the effect on the network system, taking into consideration oscillation errors and delays of the CAN bus and CAN transceiver.

19.16.2 Representative examples of baud rate settings

Table 19-28 and *Table 19-29* show representative examples of baud rate settings.

Table 19-28 Representative examples of baud rate settings (f_{CANMOD} = 8 MHz) (1/2)

Set baud rate value (unit: kbps)	Division ratio of CnBRP register	CnBRP register set value	Valid bit rate setting (unit: kbps)					CnBTR register setting value		Sampling point (unit: %)
			Length of DBT	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG13 to TSEG10	TSEG22 to TSEG20	
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 19-28 Representative examples of baud rate settings (f_{CANMOD} = 8 MHz) (2/2)

Set baud rate value (unit: kbps)	Division ratio of CnBRP register	CnBRP register set value	Valid bit rate setting (unit: kbps)					CnBTR register setting value		Sampling point (unit: %)
			Length of DBT	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG13 to TSEG10	TSEG22 to TSEG20	
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 19-28* do not guarantee the operation of the network system. Thoroughly check the effect on the network system, taking into consideration oscillation errors and delays of the CAN bus and CAN transceiver.

Table 19-29 Representative examples of baud rate settings (f_{CANMOD} = 16 MHz) (1/2)

Set baud rate value (unit: kbps)	Division ratio of CnBRP register	CnBRP register set value	Valid bit rate setting (unit: kbps)					CnBTR register setting value		Sampling point (unit: %)
			Length of DBT	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG13 to TSEG10	TSEG22 to TSEG20	
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 19-29 Representative examples of baud rate settings (f_{CANMOD} = 16 MHz) (2/2)

Set baud rate value (unit: kbps)	Division ratio of CnBRP register	CnBRP register set value	Valid bit rate setting (unit: kbps)					CnBTR register setting value		Sampling point (unit: %)
			Length of DBT	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG13 to TSEG10	TSEG22 to TSEG20	
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 19-29* do not guarantee the operation of the network system. Thoroughly check the effect on the network system, taking into consideration oscillation errors and delays of the CAN bus and CAN transceiver.

19.17 Operation of CAN Controller

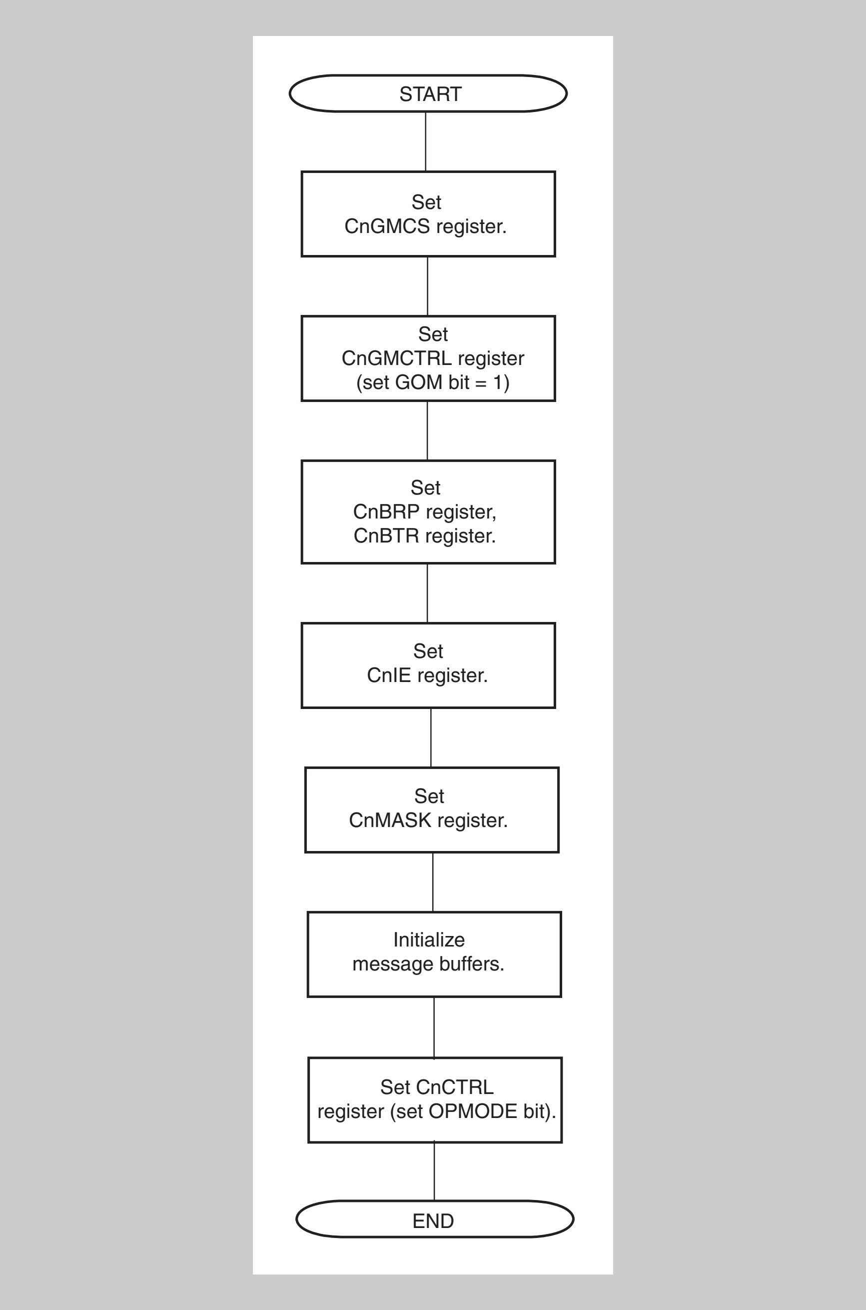


Figure 19-34 **Initialization**

Note OPMODE: Normal operation mode, normal operation mode with ABT, receive-only mode, single-shot mode, self-test mode

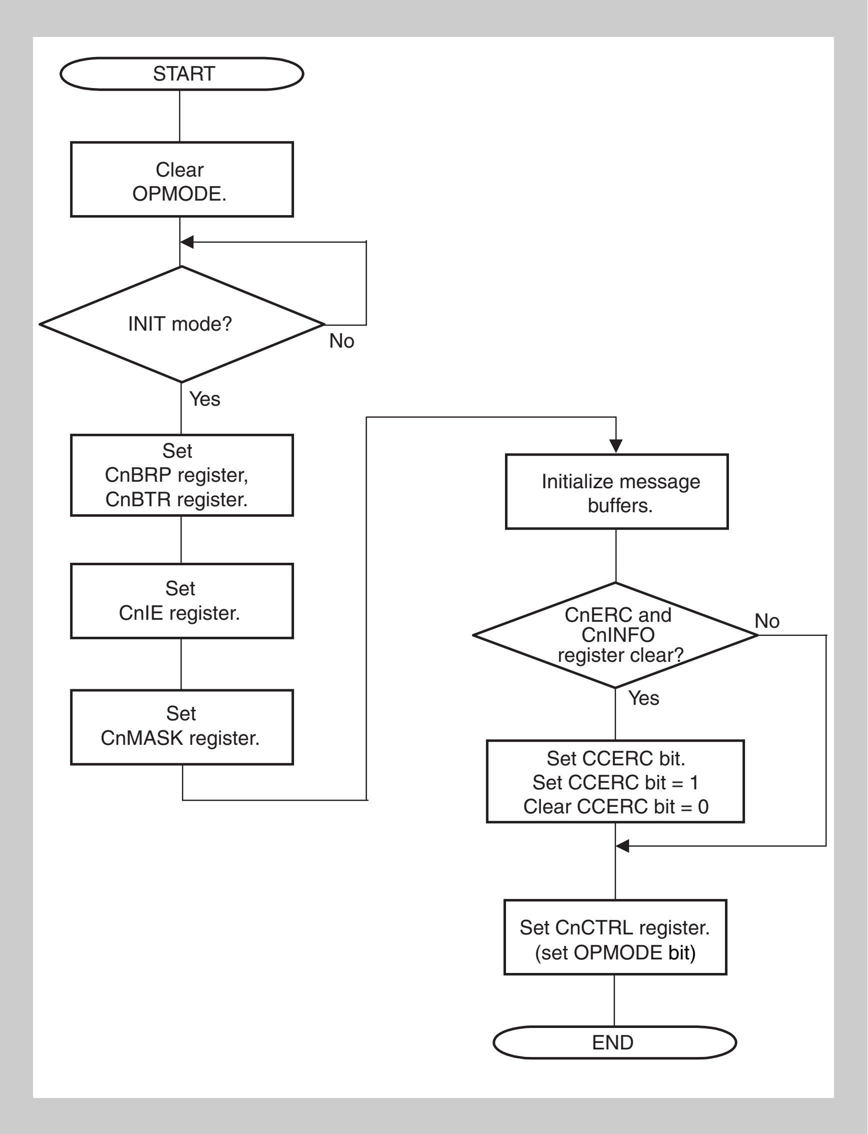


Figure 19-35 Re-initialization

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).

Note OPMODE: Normal operation mode, normal operation mode with ABT, receive-only mode, single-shot mode, self-test mode

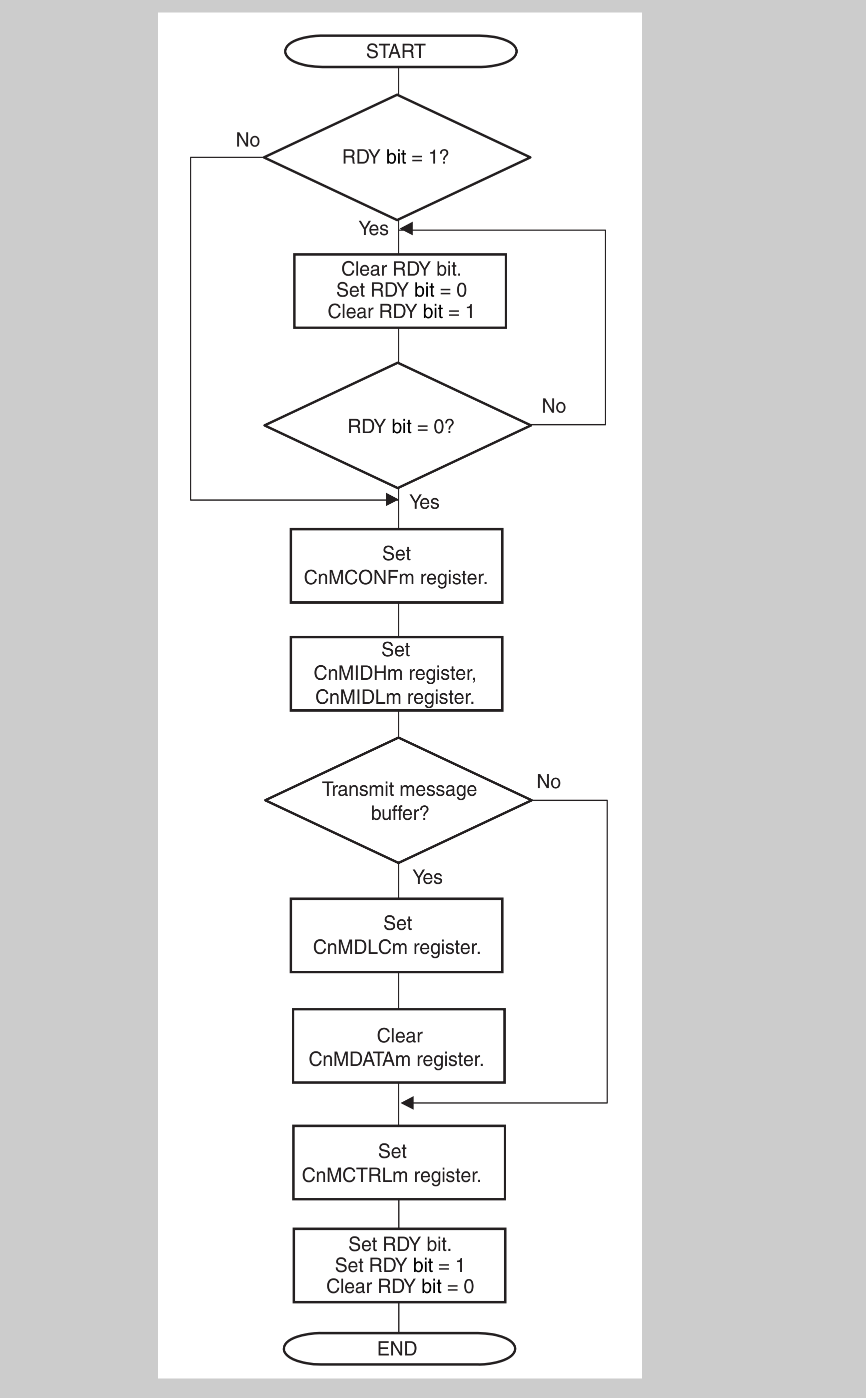


Figure 19-36 **Message buffer initialization**

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 CnMCTRLm.RDY, CnMCTRLm.TRQ, and CnMCTRLm.DN bits to 0.
   - Clear the CnMCONFm.MA0 bit to 0.

Figure 19-37 shows the processing for a receive message buffer (CnMCONFm.MT2 to CnMCONFm.MT0 bits = 001B to 101B).

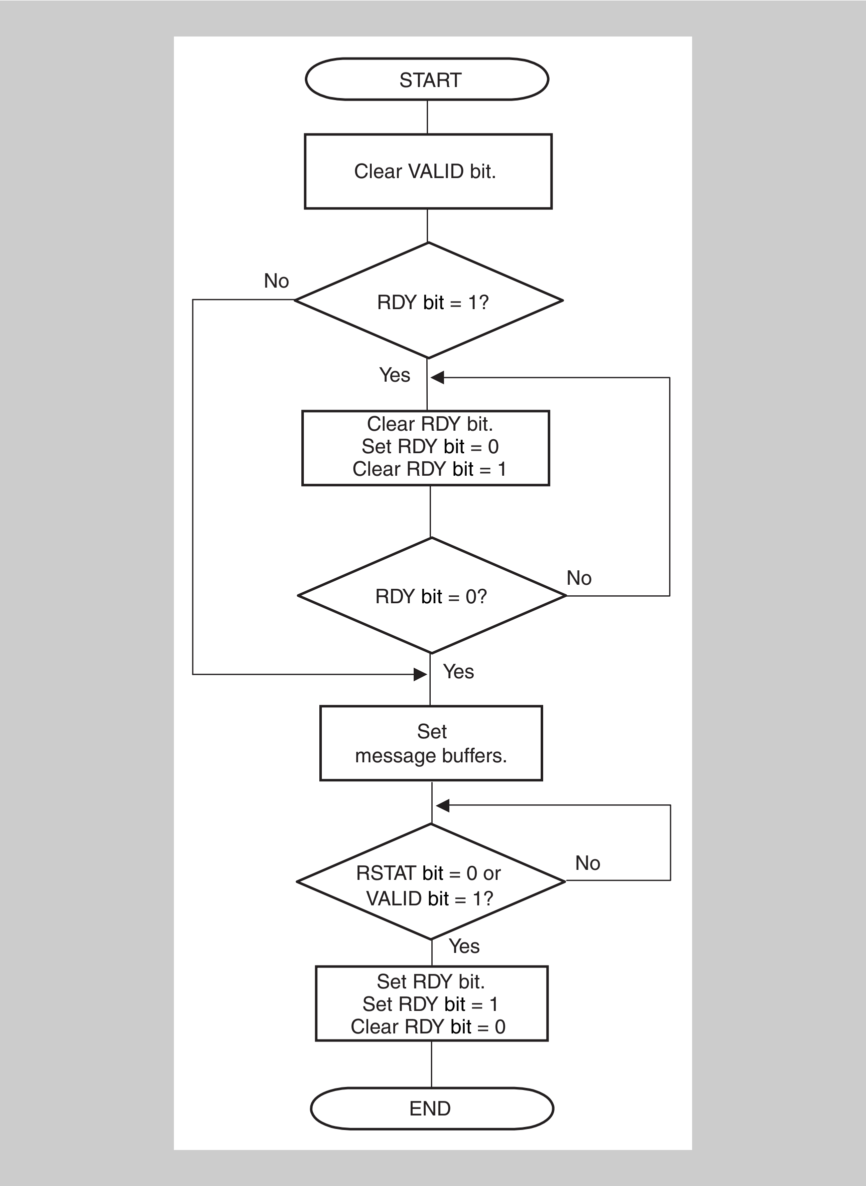


Figure 19-37 Message buffer redefinition

Figure 19-38 shows the processing for a transmit message buffer (CnMCONFm.MT2 to CnMCONFm.MT0 bits = 000B).

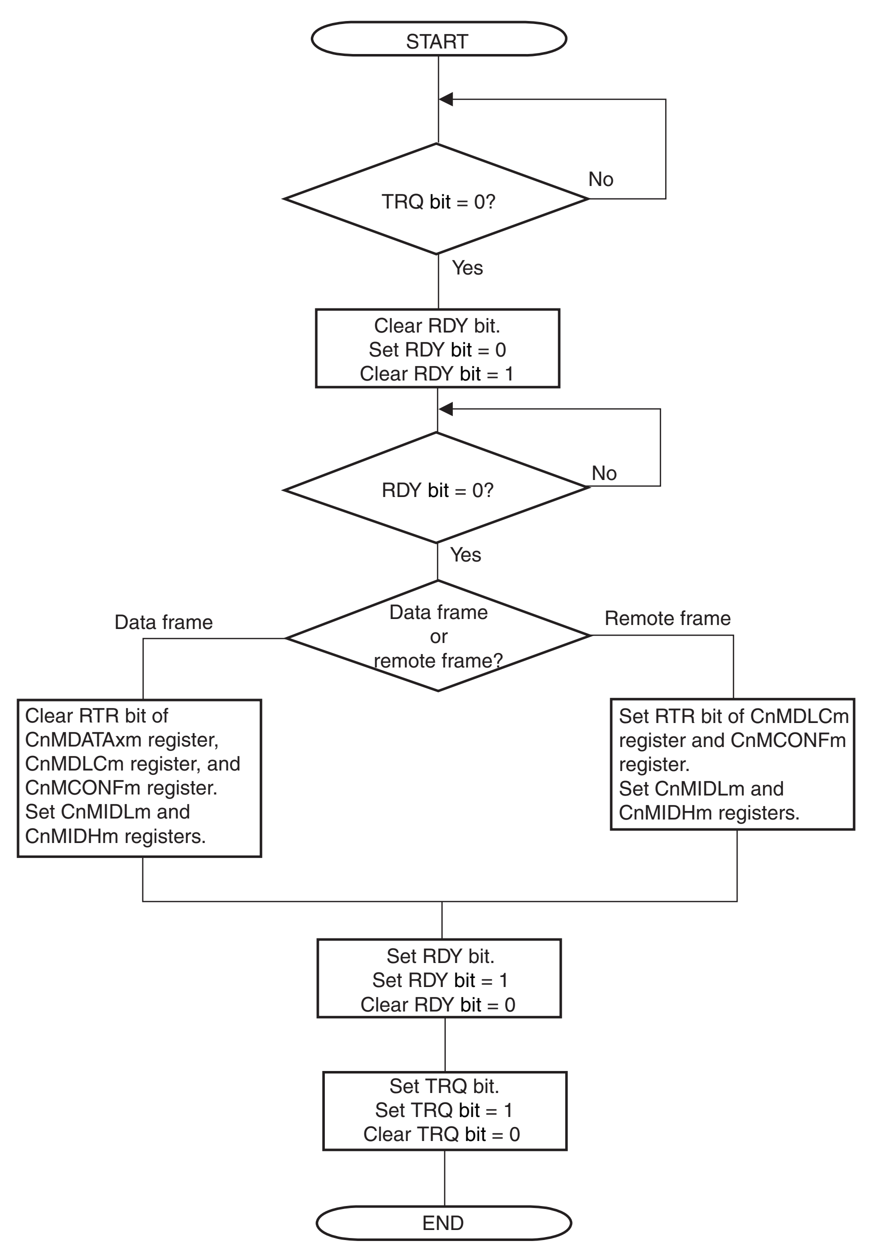


Figure 19-38 Message transmit processing (normal operation mode)

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 19-39 shows the processing for a transmit message buffer (CnMCONFm.MT2 to CnMCONFm.MT0 bits = 000B)

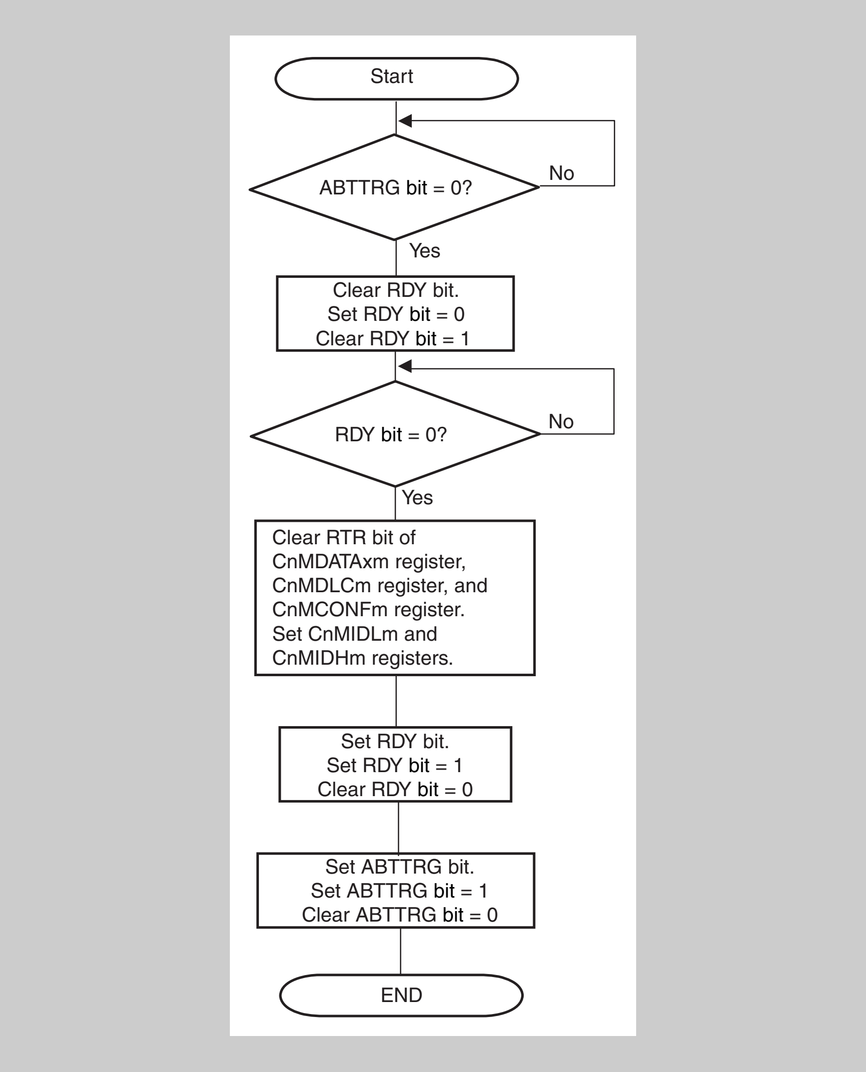


Figure 19-39 **Message transmit processing (normal operation mode with ABT)**

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 19-38 on page 749*.

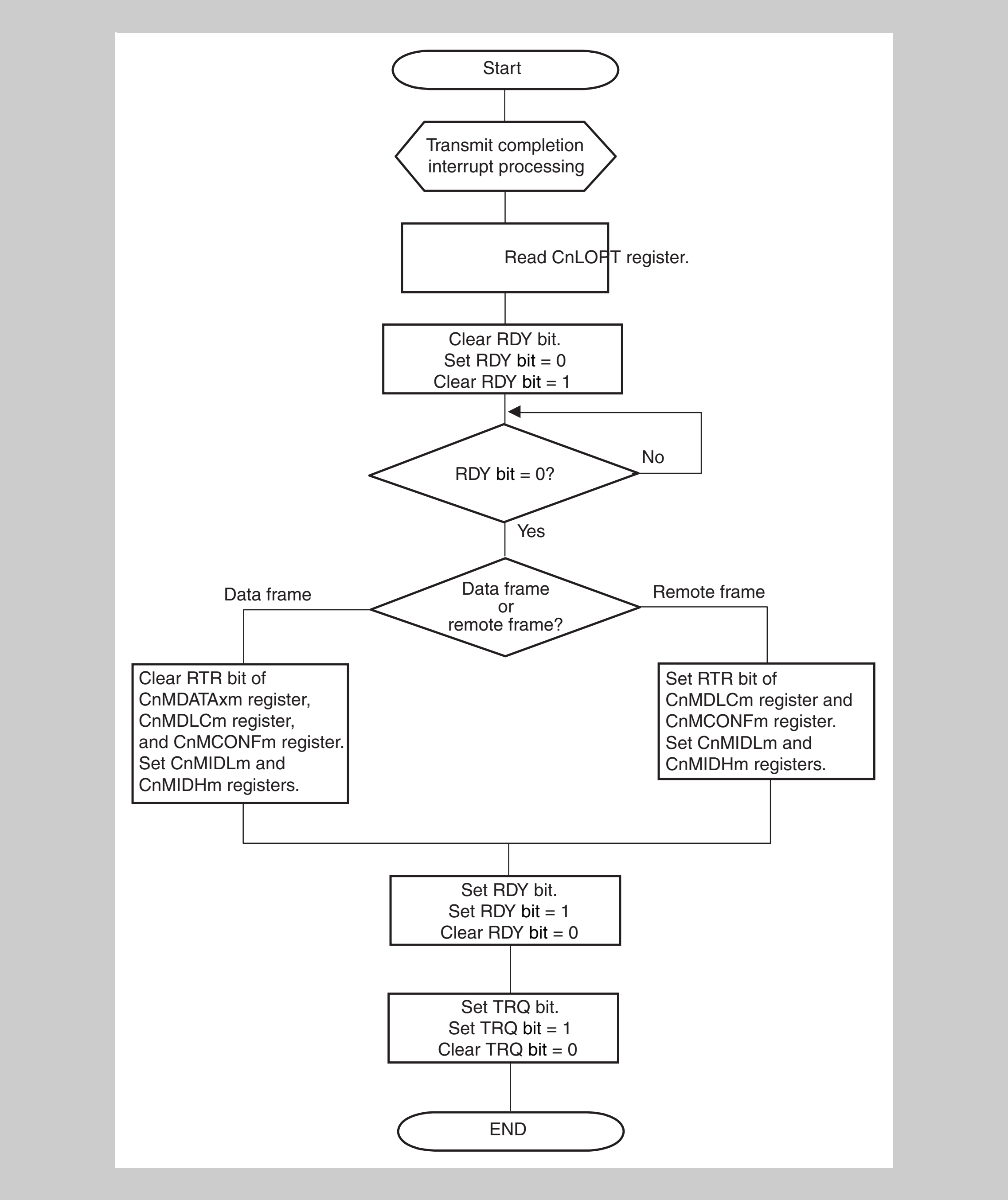


Figure 19-40 Transmission via interrupt (using CnLOPT register)

Caution The TRQ bit should be set after the RDY bit is set.
The RDY bit and TRQ bit should not be set at the same time.

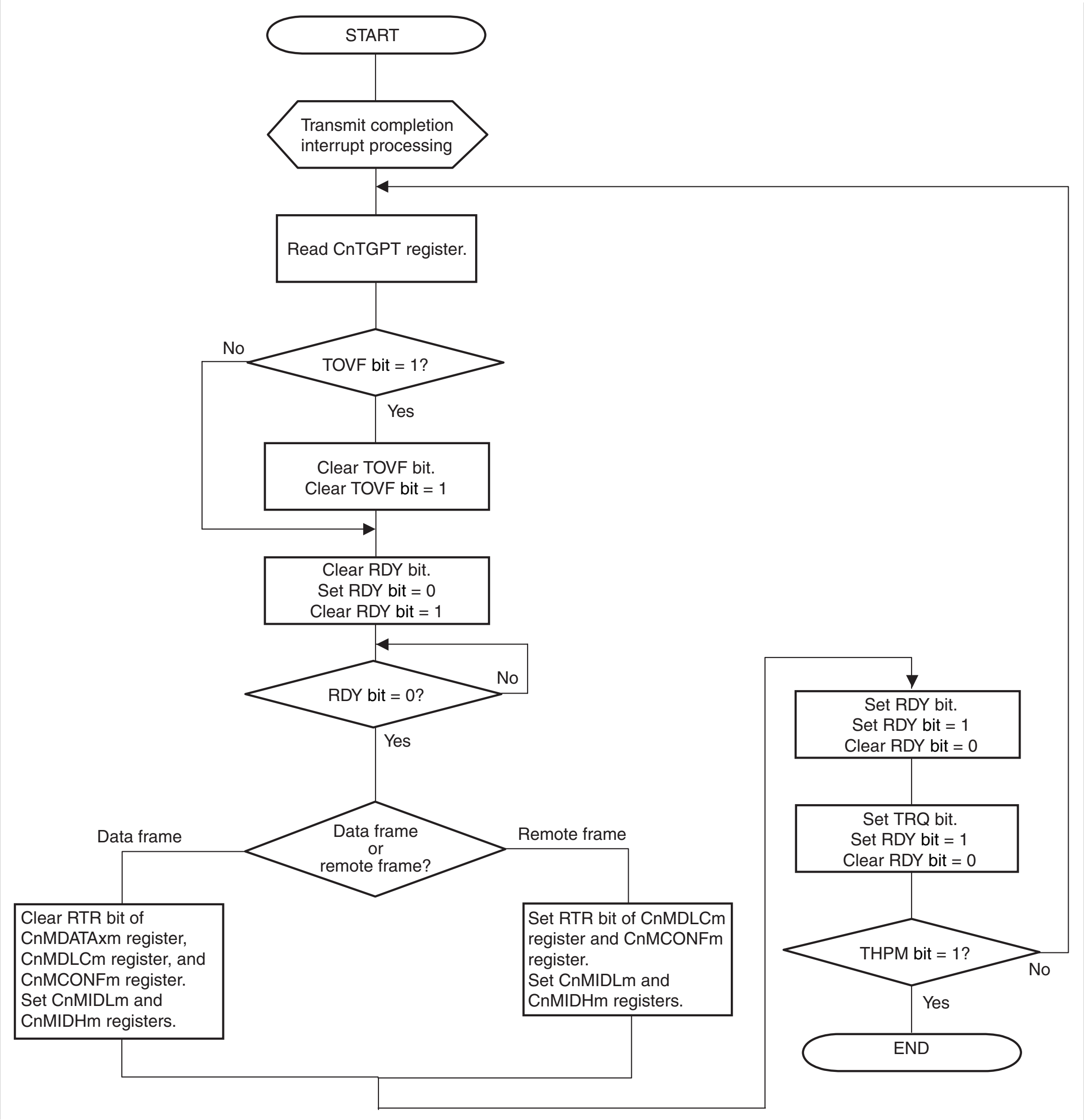


Figure 19-41 **Transmission via interrupt (using CnTGPT register)**

Caution The TRQ bit should be set after the RDY bit is set.
The RDY bit and TRQ bit should not be set at the same time.

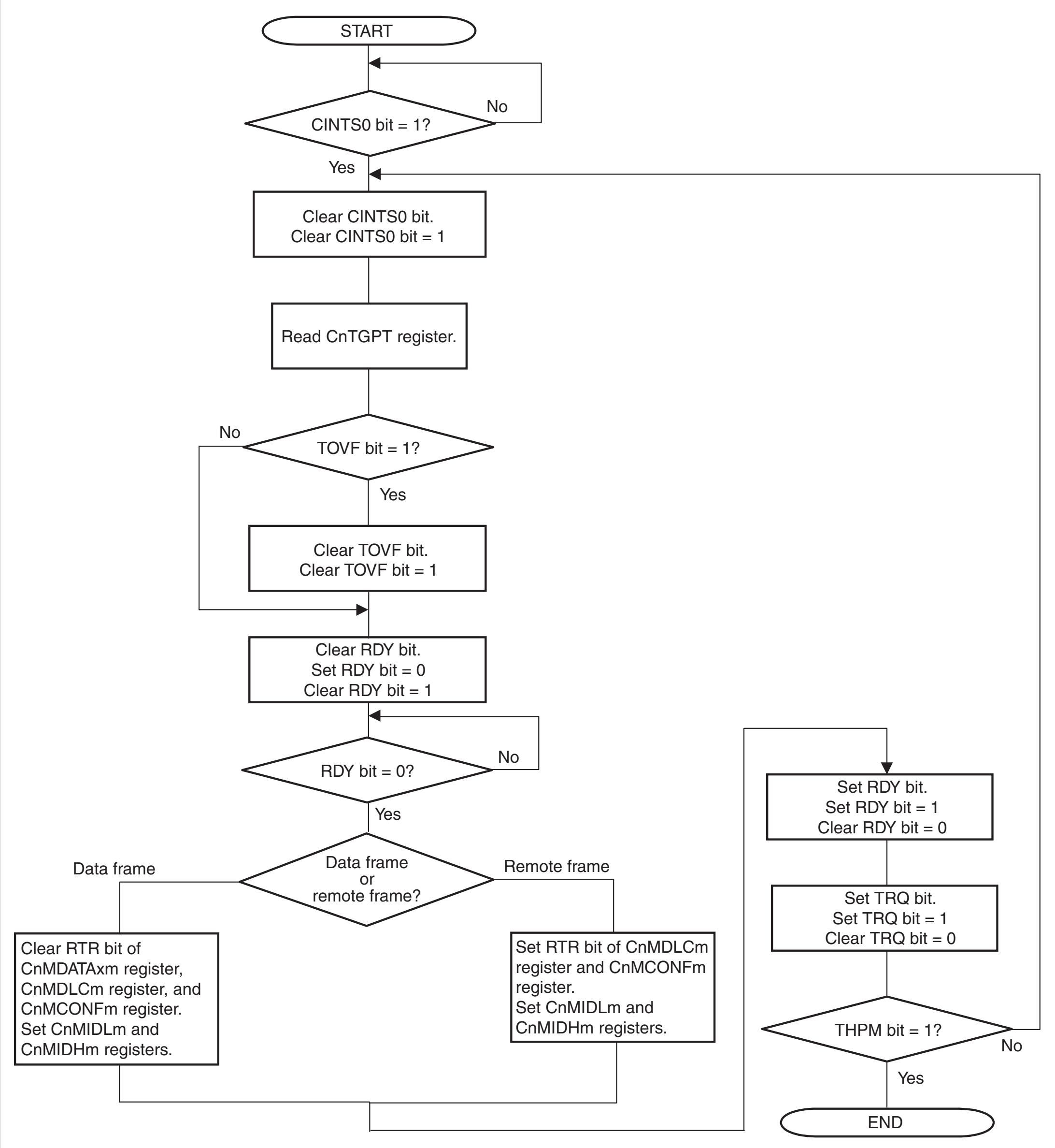


Figure 19-42 Transmission via software polling

Caution The TRQ bit should be set after the RDY bit is set.
The RDY bit and TRQ bit should not be set at the same time.

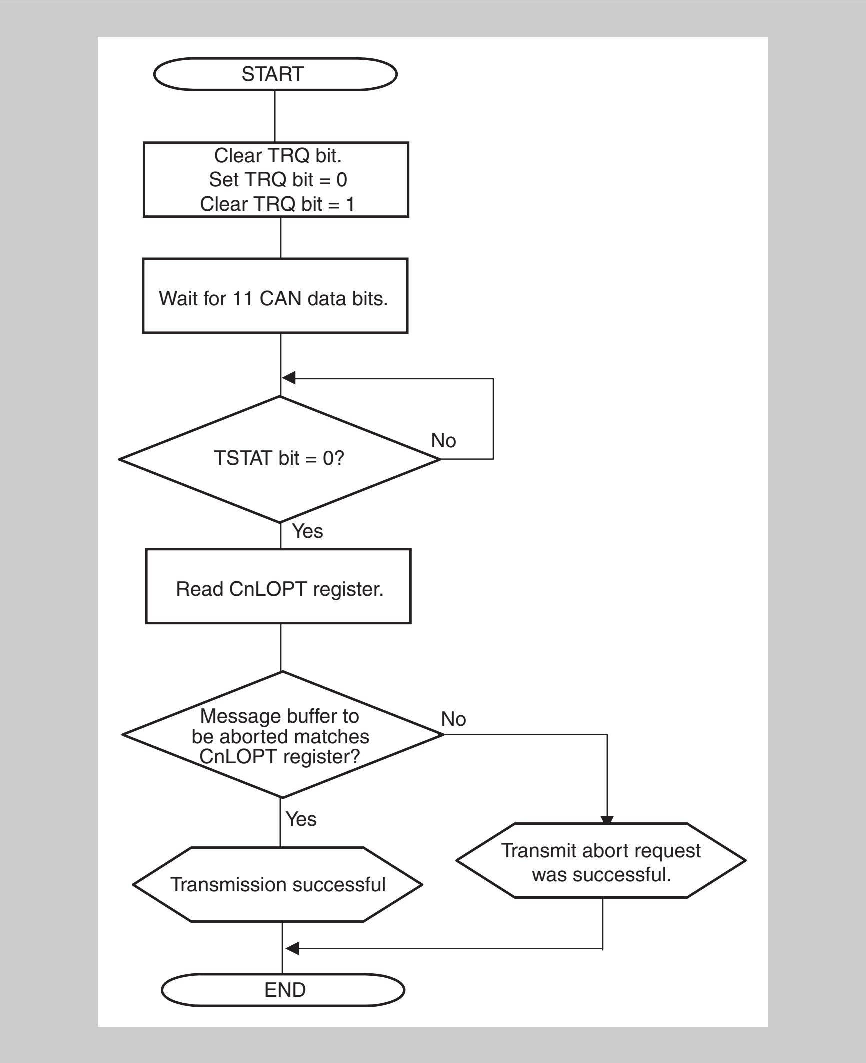


Figure 19-43 Transmission abort processing (normal operation mode)

Caution

1. Execute transmission request abort processing by clearing the TRQ bit, 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.

Figure 19-44 shows the processing to skip resumption of transmitting a message that was stopped when transmission of an ABT message buffer was aborted.

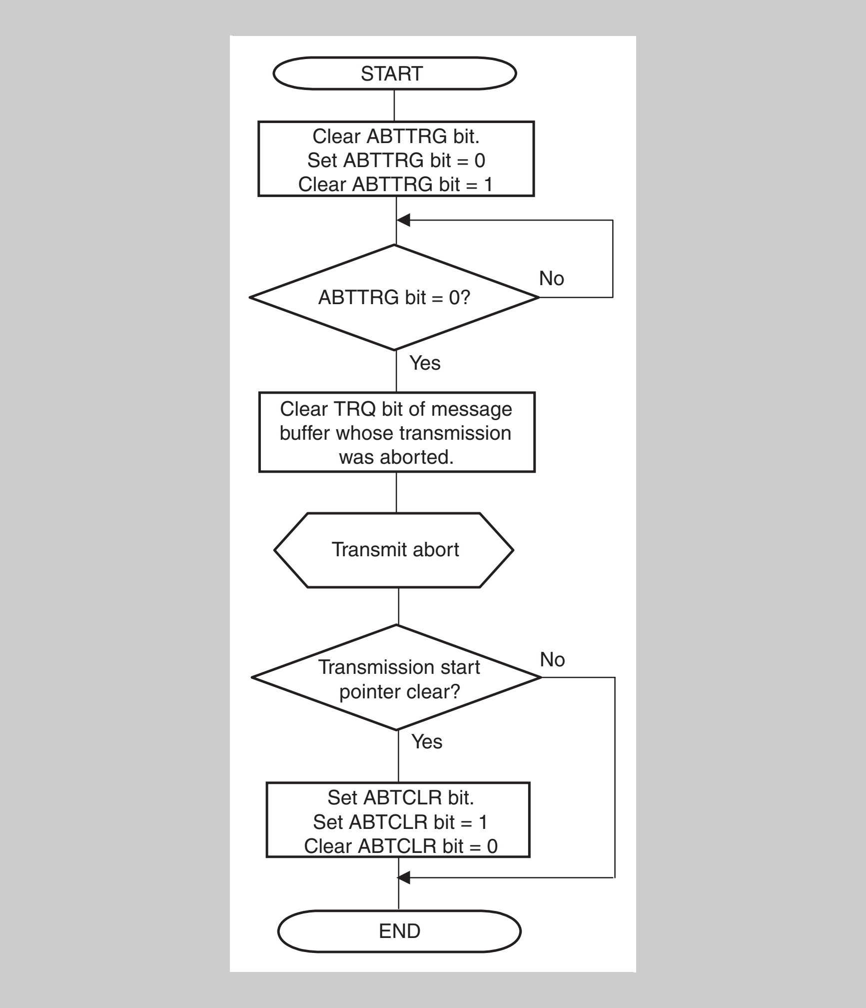


Figure 19-44 **Transmission abort processing (normal operation mode with ABT)**

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 following the procedure shown in *Figure 19-44* or *Figure 19-45*. When clearing a transmission request in an area other than the ABT area, follow the procedure shown in *Figure 19-43 on page 754*.

Figure 19-45 shows the processing to not skip resumption of transmitting a message that was stopped when transmission of an ABT message buffer was aborted.

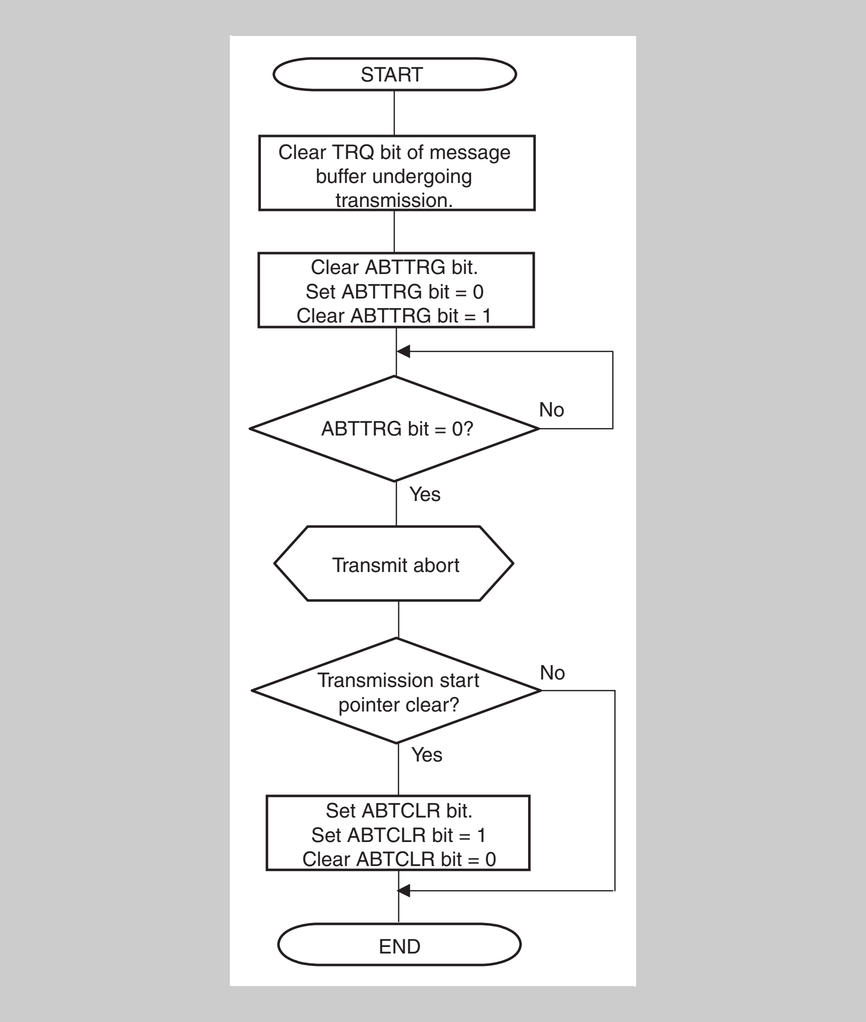


Figure 19-45 Transmission request abort processing (normal operation mode with ABT)

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 following the procedure shown in *Figure 19-44* or *Figure 19-45*. When clearing a transmission request in an area other than the ABT area, follow the procedure shown in *Figure 19-43 on page 754*.

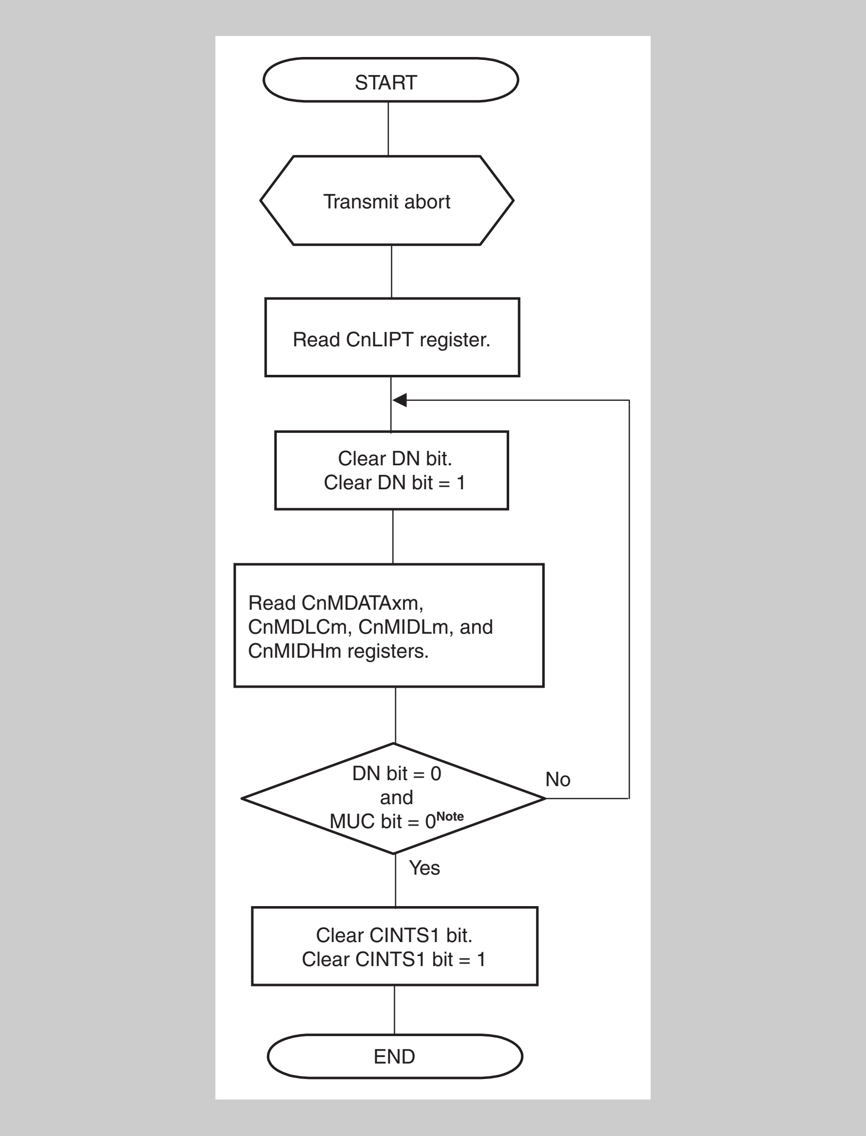


Figure 19-46 **Reception via interrupt (using CnLIPT register)**

Note Check the MUC and DN bits using one read access.

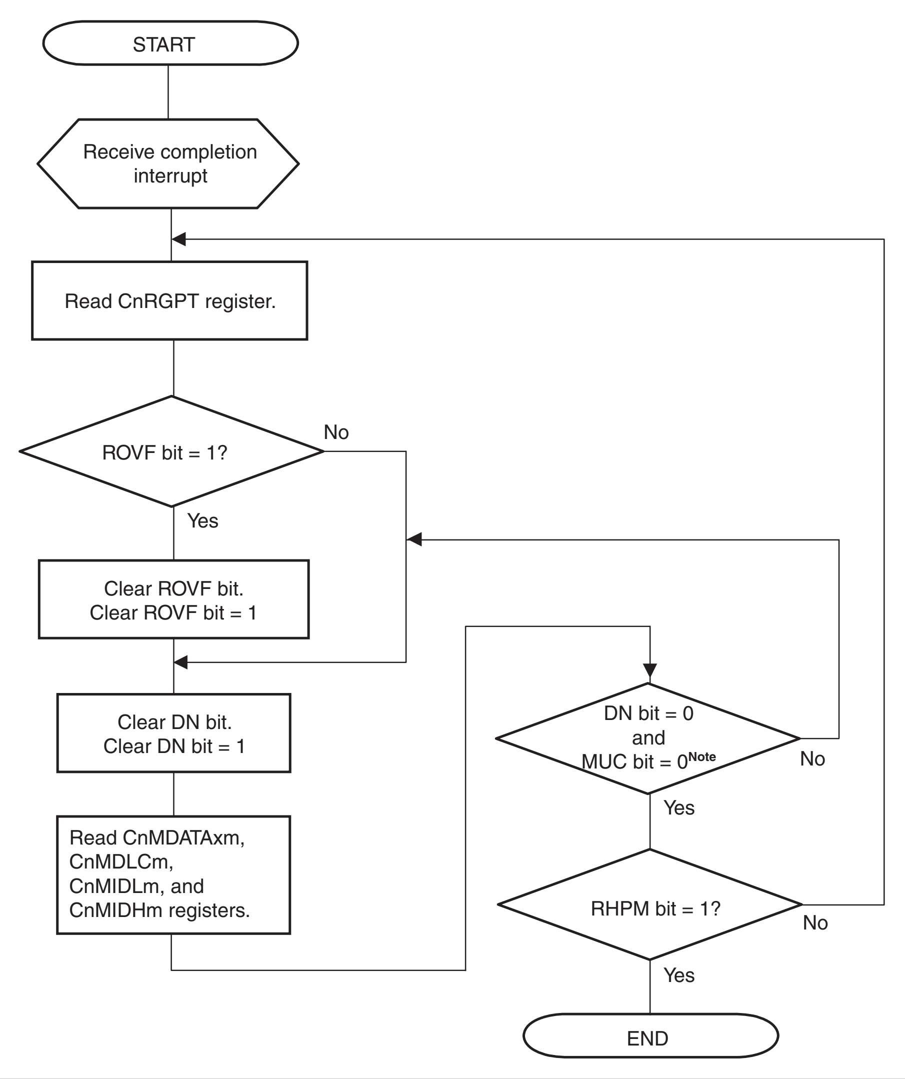


Figure 19-47 **Reception via interrupt (using CnRGPT register)**

Note Check the MUC and DN bits using one read access.

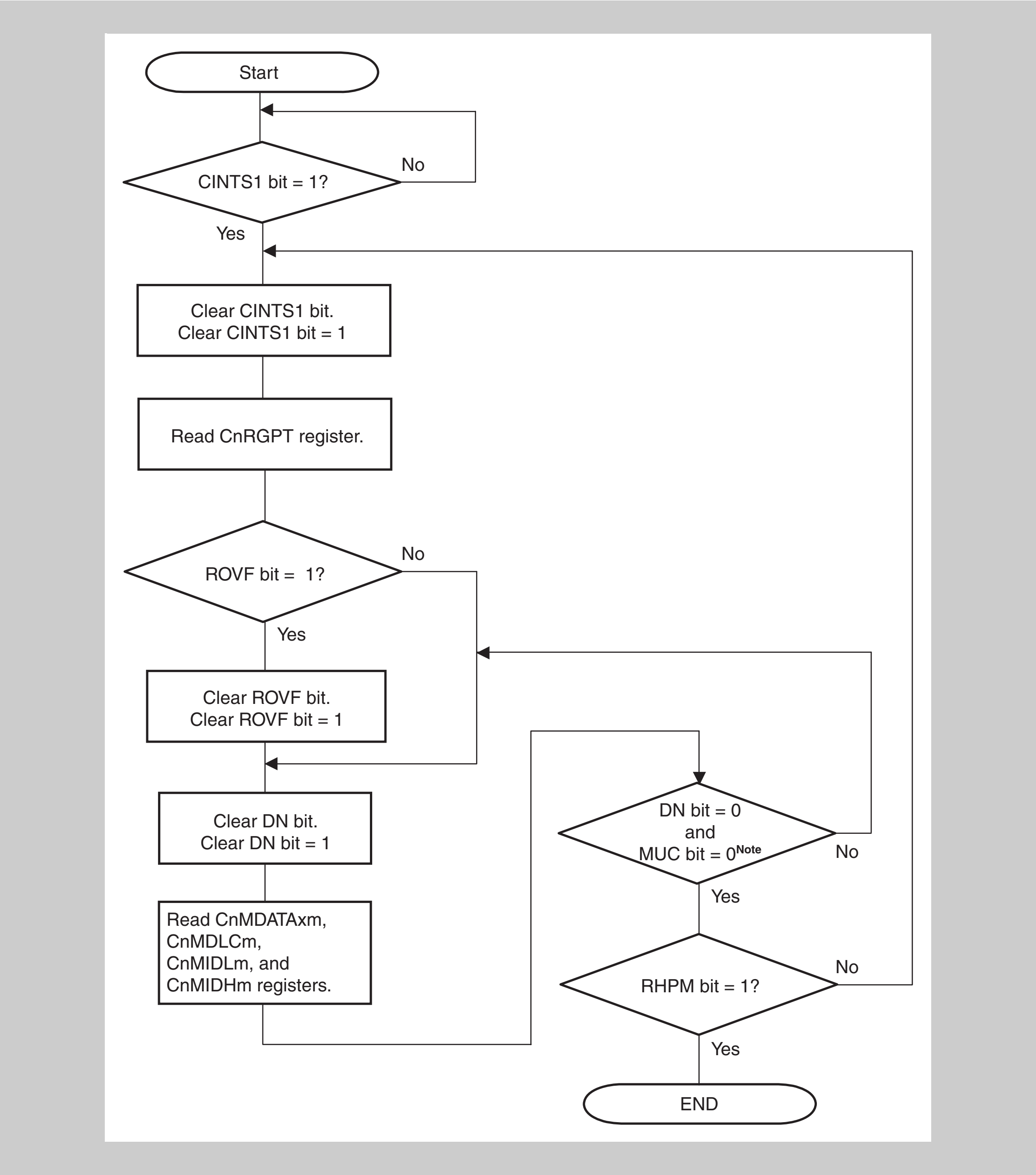


Figure 19-48 **Reception via software polling**

Note Check the MUC and DN bits using one read access.

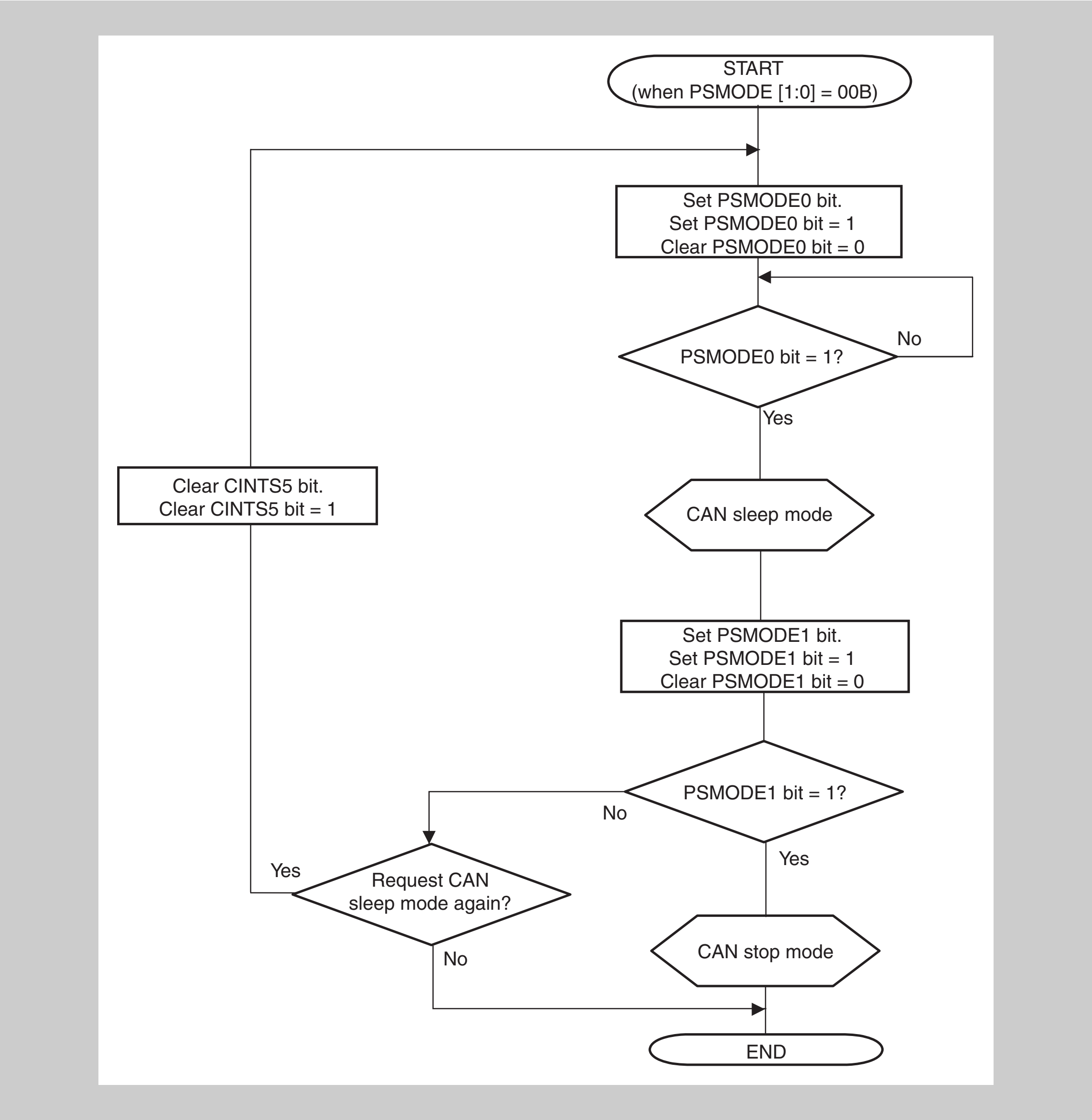


Figure 19-49 Setting CAN sleep mode/stop mode

Caution To abort transmission before making a request for the CAN sleep mode, perform processing according to *Figure 19-43 on page 754* and *Figure 19-44 on page 755*.

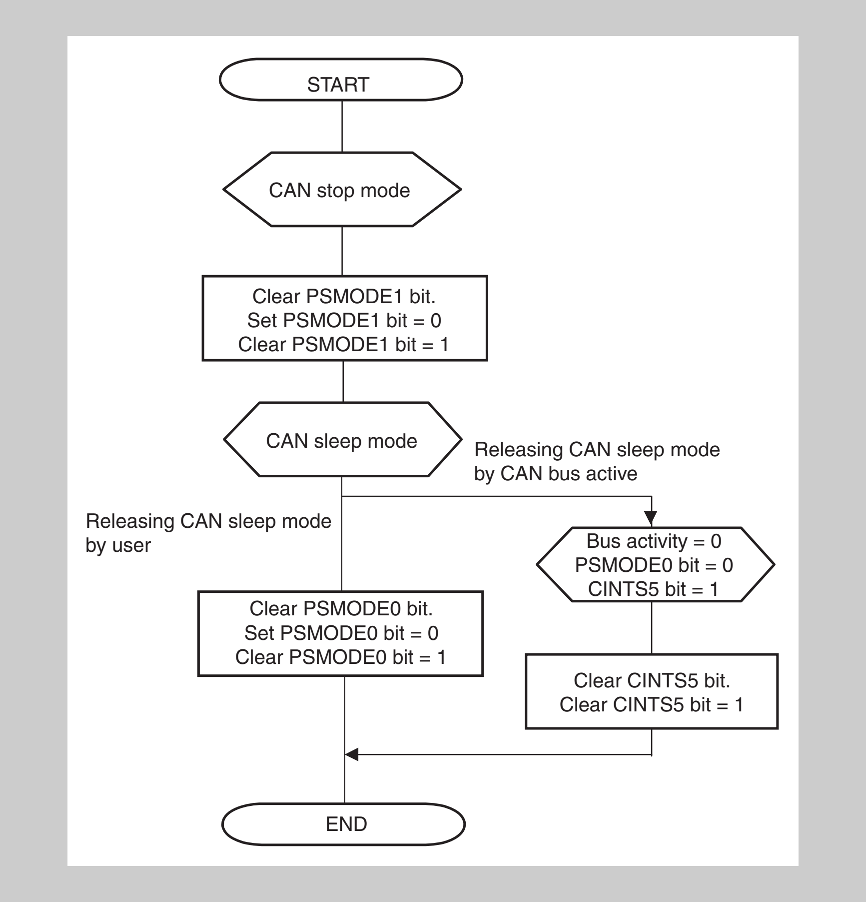


Figure 19-50 Clear CAN sleep/stop mode

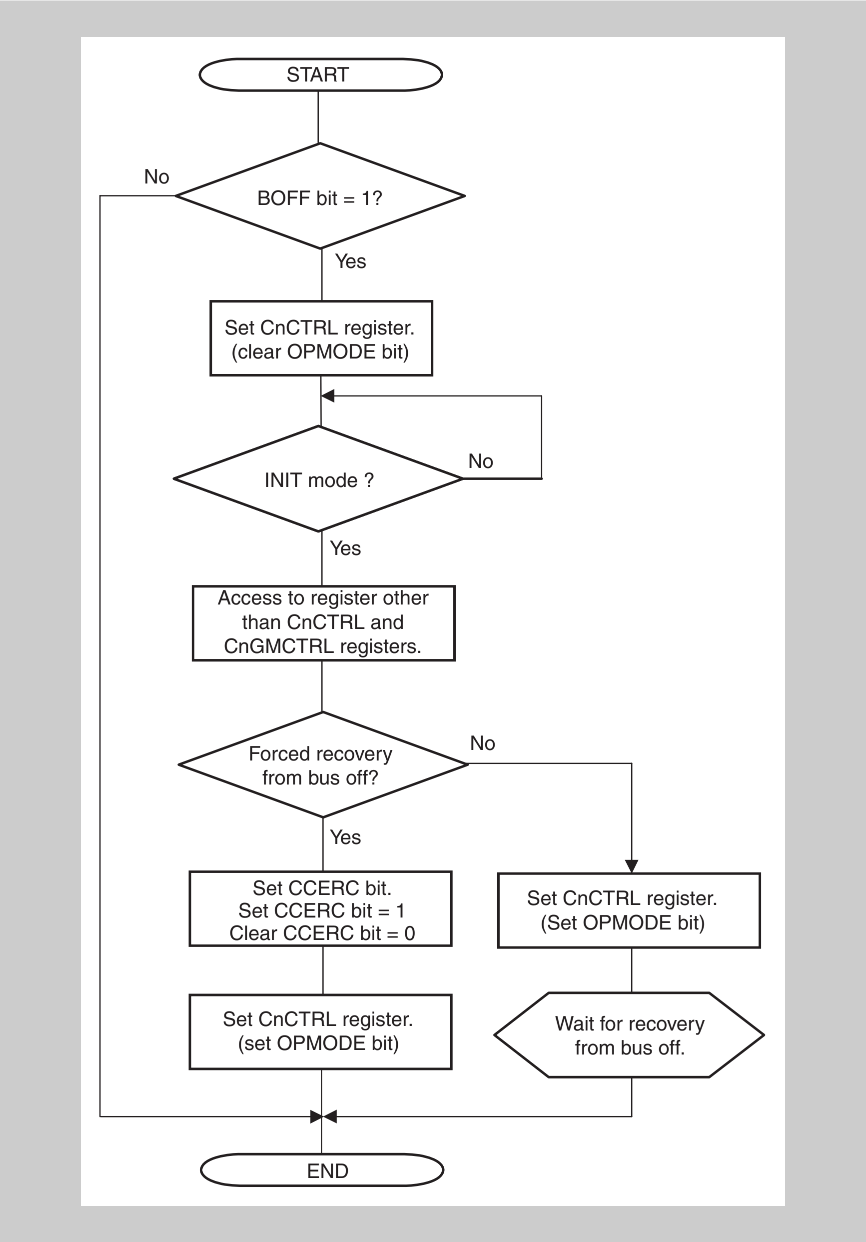


Figure 19-51 Bus-Off recovery

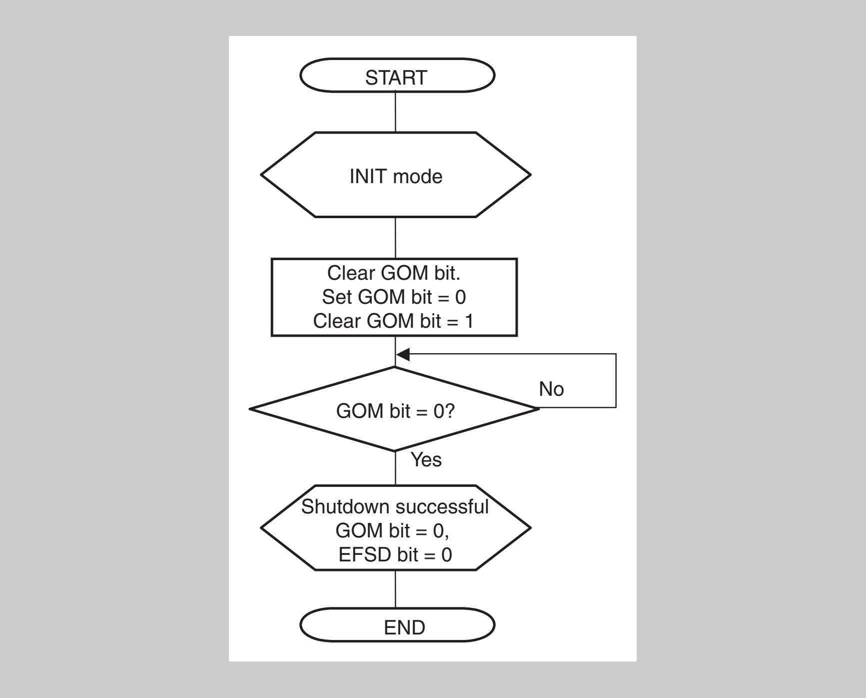


Figure 19-52 Normal shutdown process

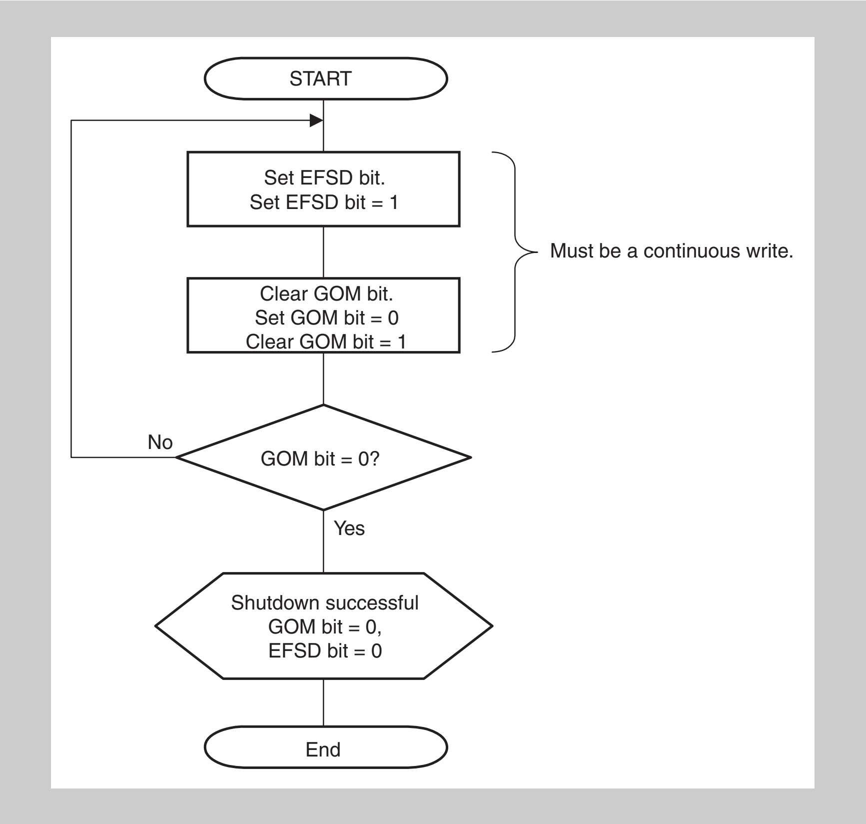


Figure 19-53 **Forced shutdown process**

Caution Do not read- or write-access any registers by software between setting the EFSD bit and clearing the GOM bit.

Note OPMODE: Normal operation mode, normal operation mode with ABT, receive-only mode, single-shot mode, self-test mode

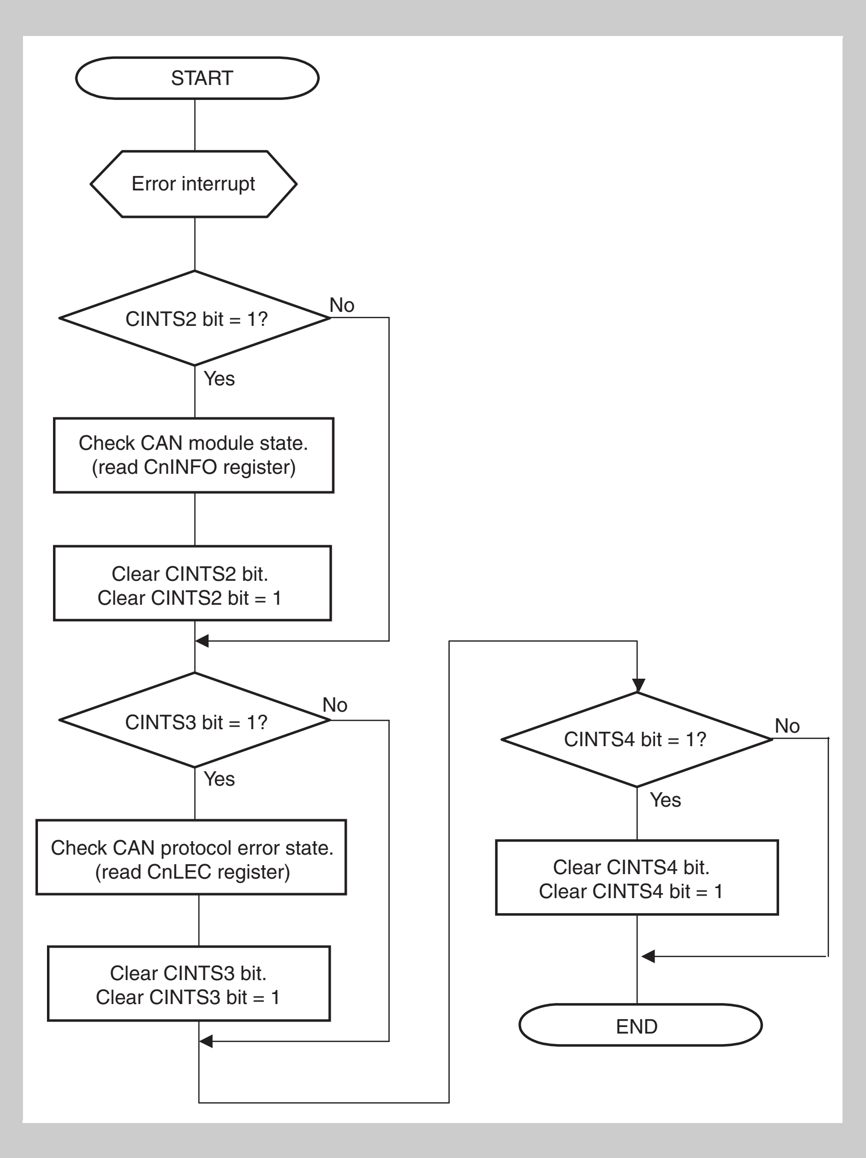


Figure 19-54 Error handling

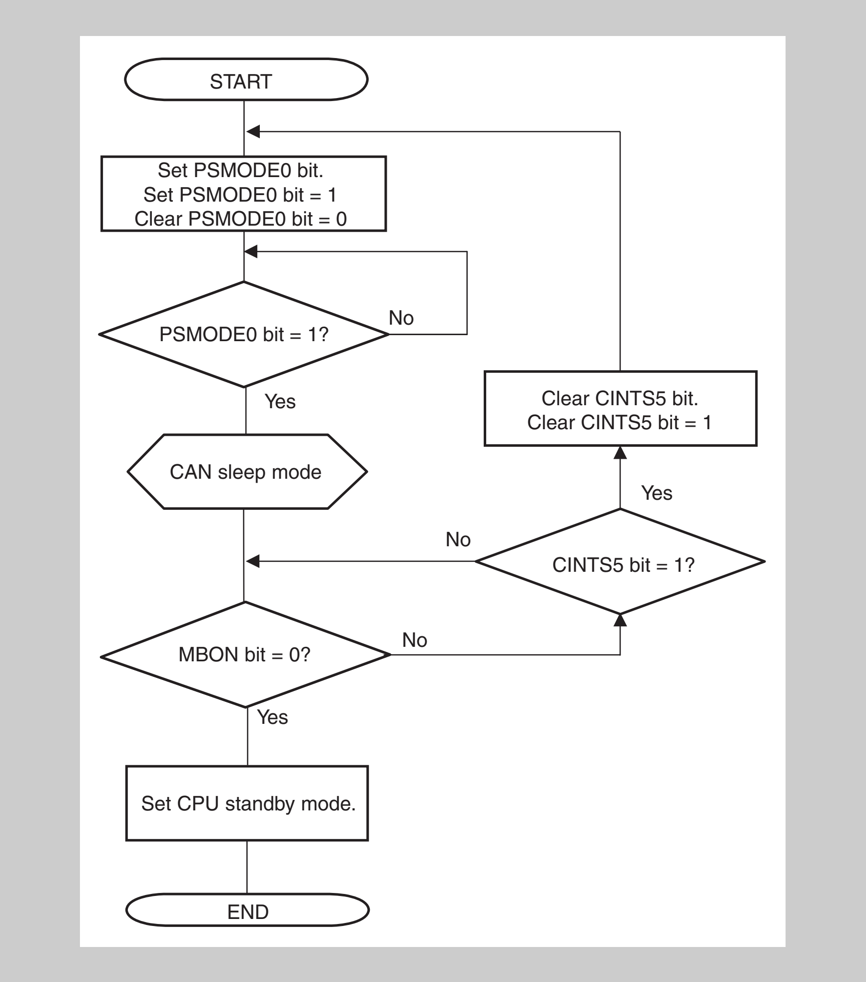


Figure 19-55 Setting CPU stand-by (from CAN sleep mode)

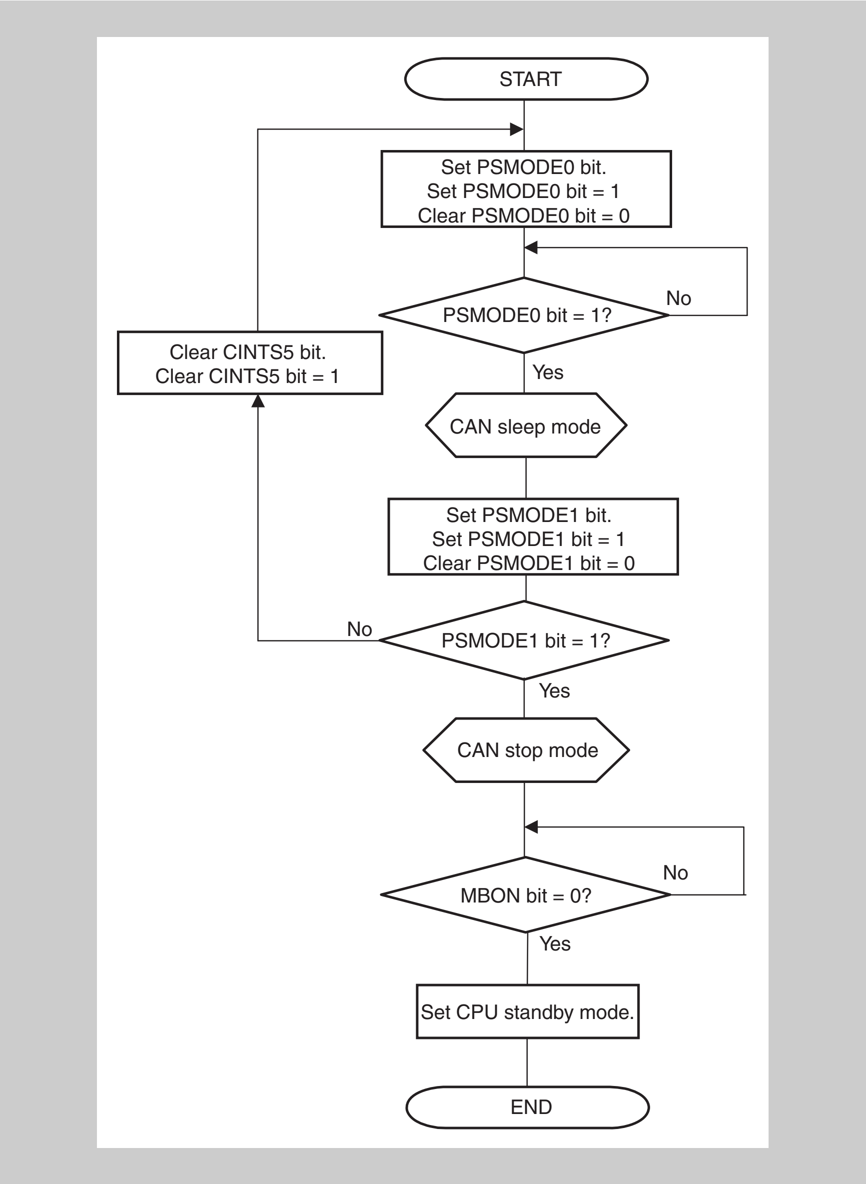


Figure 19-56 Setting CPU stand-by (from CAN stop mode)

19.18 Operating Precautions

19.18.1 Wake-up from sleep mode

(1) Description

When the CAN macro is set into SLEEP mode, it can be waken up by CAN bus activity.

This waking up is asynchronous to the operation of the macro and the CPU. By configuration setting, a WAKEUP interrupt can be generated by the CAN macro on the wakeup event.

While the interrupt is generated asynchronously, the CAN macro may need another dominant edge on the CAN bus, in order to restart its synchronous operation. The necessity of another dominant edge on the CAN bus to wake up depends on the phase between internal clocking of the CAN macro and the signal on the CAN bus.

During the time, after the interrupt already has been indicated, and before the CAN macro has restarted its synchronous operation, the registers of the CAN macro will not operate.

The worst case (maximum length) of this latency time is given by the CAN bus speed and the rule of the CAN bus about the frequency of recessive to dominant edges. Given by the stuffing rule, at least every 10 bits, a recessive to dominant edge must occur.

This means in an example:

- CAN Bus Speed: 1 Mbit/s
  10 CAN Bits within: 10 µs
  Worst case wakeup latency: 10 µs

(2) Software improvement hint

Within the WAKEUP interrupt routine, create a waiting loop, which tests the capability of clearing the WAKEUP interrupt flag within CAN, by checking the actual power save mode.

In the following C-code example, replace the objects in “<>” brackets by the hardware locations within your implementation. Use the appropriate access types, as described in the user’s manual.

```
do
{
     AFCAN_SleepStatus = <CnCTRL_PSMODE>
     if( AFCAN_SleepStatus != 0 )
     {
          /* macro is still in SLEEP mode (waiting for latency time) */
          <CnINTS_CINTS5> = 1; /* repeated trying to clear CINTS5 */
     }
} while( AFCAN_SleepStatus != 0 );
```

Chapter 20 A/D Converter (ADC)

These microcontrollers contain an n-channel 10-bit A/D Converter.

The V850E/Dx3 microcontrollers have following number of the channels:

ADC	**µPD70F3427, µPD70F3426 µPD70F3425, µPD70F3424**	**µPD70F3423, µPD70(F)3422 µPD70(F)3421, µPD70(F)3420**
Instances	16	10

Throughout this chapter, the individual channels of the A/D Converter are identified by “n”, for example ADCR0n for the A/D conversion result register of channel n.

20.1 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
- The following functions are provided as trigger modes.
  - Software trigger mode
  - Timer trigger mode
- Power-fail monitor function (conversion result compare function)

The block diagram of the A/D Converter is shown below.

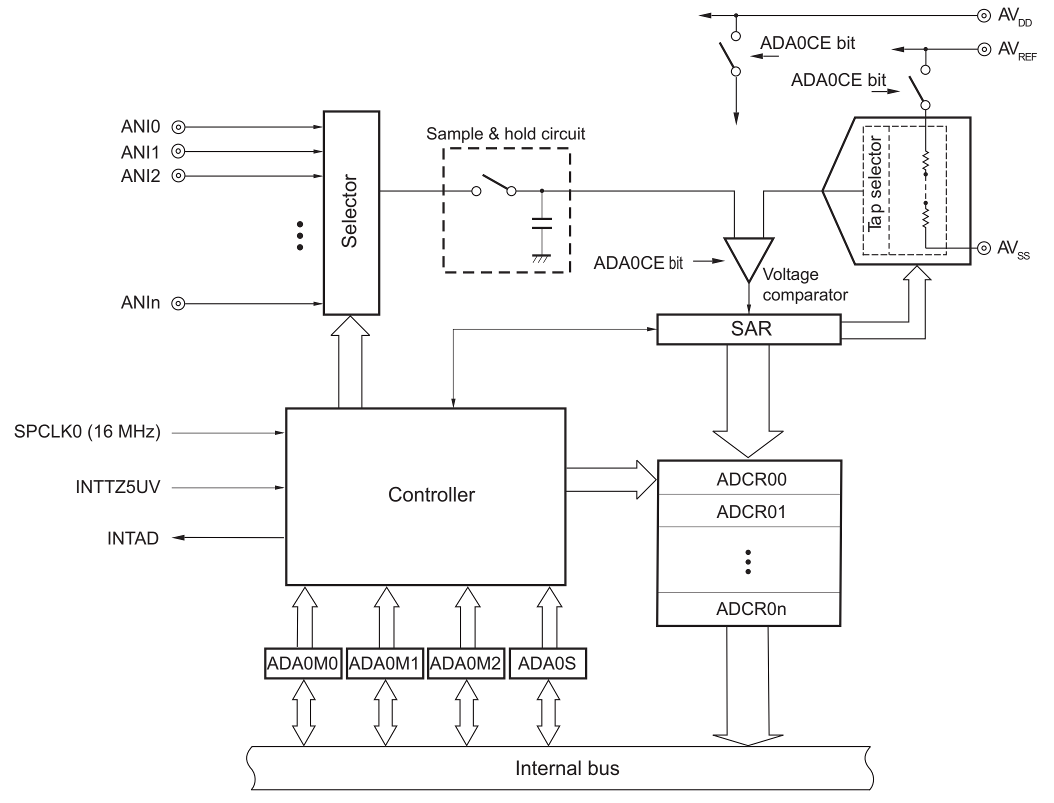


Figure 20-1 Block diagram of A/D Converter

20.2 Configuration

The A/D Converter includes the following hardware.

Table 20-1 Configuration of A/D Converter

Item	Configuration
Analog inputs	ANI0 to ANIn pins
Registers	Successive approximation register (SAR) A/D conversion result registers ADCR00 to ADCR0n A/D conversion result registers ADCR0H0 to ADCR0Hn: only higher 8 bits can be read
Control registers	A/D Converter mode registers 0 to 2 (ADA0M0 to ADA0M2) A/D Converter channel specification register 0 (ADA0S)

Caution It is mandatory to enable the A/D Converter after any reset and to perform a first conversion within a time period of maximum 1 s after reset release. With the execution of the first conversion, the A/D Converter circuit is initialized.

The execution of a first conversion is mandatory independently of whether the A/D Converter is used later on by the user application.

(1) Successive approximation register (SAR)

The SAR register compares the voltage value of the analog input signal with the voltage tap (compare voltage) value from the series resistor string, and holds the comparison result starting from the most significant bit (MSB).

When the comparison result has been held down to the least significant bit (LSB) (i.e., when A/D conversion is complete), the contents of the SAR register are transferred to the ADCR0n register.

(2) A/D conversion result register n (ADCR0n), A/D conversion result register nH (ADCR0Hn)

The ADCR0n register is a 16-bit register that stores the A/D conversion result. ADCR0n consist of 16 registers and the A/D conversion result is stored in the 10 higher bits of the ADCR0n register corresponding to analog input. (The lower 6 bits are fixed to 0.)

The ADCR0n register is read-only, in 16-bit units.

When using only the higher 8 bits of the A/D conversion result, the ADCR0Hn register is read-only, in 8-bit units.

Caution A write operation to the ADA0M0 and ADA0S registers may cause the contents of the ADCR0n register to become undefined. After the conversion, read the conversion result before writing to the ADA0M0 and ADA0S registers. Correct conversion results may not be read if a sequence other than the above is used.

(3) Power-fail compare threshold value register (ADA0PFT)

The ADA0PFT register sets a threshold value that is compared with the value of A/D conversion result register nH (ADCR0Hn). The 8-bit data set to the ADA0PFT register is compared with the higher 8 bits of the A/D conversion result register (ADCR0Hn).

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

Reset input clears this register to 00H.

(4) Sample & hold circuit

The sample & hold circuit samples each of the analog input signals selected by the input circuit and sends the sampled data to the voltage comparator. This circuit also holds the sampled analog input signal voltage during A/D conversion.

(5) Voltage comparator

The voltage comparator compares a voltage value that has been sampled and held with the voltage value of the series resistor string.

(6) Series resistor string

This series resistor string is connected between AV_{REF} and AV_{SS} and generates a voltage for comparison with the analog input signal.

(7) ANIn pins

These are analog input pins for the 16 A/D Converter channels and are used to input analog signals to be converted into digital signals. Pins other than the one selected as the analog input by the ADA0S register can be used as input port pins.

Caution

1. Make sure that the voltages input to the ANIn pins do not exceed the rated values. In particular if a voltage of AV_{REF} or higher is input to a channel, the conversion value of that channel becomes undefined, and the conversion values of the other channels may also be affected.
2. The analog input pins ANIn function also as input port pins. If any of ANIn is selected and A/D converted, do not execute an input instruction this ports during conversion. If executed, the conversion resolution may be degraded.

---

(8) AV_{REF} pin

This is the pin used to input the reference voltage of the A/D Converter. The signals input to the ANIn pins are converted to digital signals based on the voltage applied between the AV_{REF} and AV_{SS} pins.

(9) AV_{SS} pin

This is the ground pin of the A/D Converter. Always make the potential at this pin the same as that at the V_{SS} pin even when the A/D Converter is not used.

20.3 ADC Registers

The A/D Converter is controlled by the following registers:

- A/D Converter mode registers 0, 1, 2 (ADA0M0, ADA0M1, ADA0M2)
- A/D Converter channel specification register 0 (ADA0S)
- Power-fail compare mode register (ADA0PFM)

The following registers are also used:

- A/D conversion result register n (ADCR0n)
- A/D conversion result register nH (ADCR0Hn)
- Power-fail compare threshold value register (ADA0PFT)

(1) ADA0M0 - ADC mode register 0

The ADA0M0 register is an 8-bit register that specifies the operation mode and controls conversion operations.

This register can be read orwritten in 8-bit or 1-bit units. However, bit 0 is read-only.

Reset input clears this register to 00H.

After reset: 00H R/W Address: FFFFF200H

	7	6	5	4	3	2	1	0
ADA0M0	ADA0CE	0	ADA0MD1	ADA0MD0	0	0	ADA0TMD	ADA0EF

ADA0CE	A/D conversion control
0	Stops conversion
1	Starts conversion

ADA0MD1	ADA0MD0	Specification of A/D conversion operation mode
0	0	Continuous select mode
0	1	Continuous scan mode
others		Setting prohibited

ADA0TMD	Trigger mode specification
0	Software trigger mode
1	Timer trigger mode

ADA0EF	A/D converter status display
0	A/D conversion stopped
1	A/D conversion in progress

Caution

1. If bit 0 is written, this is ignored.
2. Changing the ADA0FR3 to ADA0FR0 bits of the ADA0M1 register during conversion (ADA0CE0 bit = 1) is prohibited.
3. When not using the A/D Converter, stop the operation by setting the ADA0CE bit to 0 to reduce the current consumption.

(2) ADA0M1 - ADC mode register 1

The ADA0M1 register is an 8-bit register that controls the conversion time specification.

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

Reset input clears this bit to 00H.

After reset: 00H R/W Address: FFFFF201H

	7	6	5	4	3	2	1	0
ADA0M1	1	0	0	0	ADA0FR3	ADA0FR2	ADA0FR1	ADA0FR0

Caution

1. The bit 7 must be changed to “1” after reset and must not be changed afterwards.
2. Be sure to clear bits 5 and 4 to 0.

For A/D conversion time settings, see *Table 20-2*.

Table 20-2 Conversion time settings

ADA0FR				divider	f_{SPCLK0} = 16 MHz		f_{SPCLK0} = 4 MHz		Stabilization time[a]
3	2	1	0	div	conversion time[b]	sampling time[c]	conversion time[b]	sampling time[c]	
0	0	0	0	1	prohibited		7.75 µs	4.13 µs	$16/f_{SPCLK0}$
0	0	0	1	2	3.88 µs	2.06 µs	15.50 µs	8.25 µs	$31/f_{SPCLK0}$
0	0	1	0	3	5.81 µs	3.09 µs	prohibited		$47/f_{SPCLK0}$
0	0	1	1	4	7.75 µs	4.13 µs	prohibited		$50/f_{SPCLK0}$
0	1	0	0	5	9.69 µs	5.16 µs	prohibited		$50/f_{SPCLK0}$
0	1	0	1	6	11.63 µs	6.12 µs	prohibited		$50/f_{SPCLK0}$
0	1	1	0	7	13.56 µs	7.22 µs	prohibited		$50/f_{SPCLK0}$
0	1	1	1	8	15.50 µs	8.25 µs	prohibited		$50/f_{SPCLK0}$
1	x	x	x	prohibited					

a) **When A/D conversion is started by ADA0M0.ADA0CE = 0 → 1 the first sampling of the ANIn input is delayed by the given stabilization time. This ensures compliance with the necessary stabilization time. The stabilization time applies only prior to the first sampling.**

b) **The conversion time is calculated by (31 x div) / f_{SPCLK0}.**

c) **The sampling time is calculated by (16.5 x div) / f_{SPCLK0}.**

Note Note that the given times in *Table 20-2* do not regard the dithering of the A/D converter supply clock. Using a dithering supply clock does not impact the A/D converter's operation.

(3) ADA0M2 - ADC mode register 2

The ADA0M2 register specifies the hardware trigger mode.

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

Reset input clears this register to 00H.

After reset: 00H R/W Address: FFFFF203H

	7	6	5	4	3	2	1	0
ADA0M2	0	0	0	0	0	0	ADA0TMD1	ADA0TMD0

ADA0TMD1	ADA0TMD0	Specification of trigger mode
0	0	No trigger
0	1	INTTZ5UV trigger
Others		Setting prohibited

Caution Be sure to clear bits 7 to 1.

(4) ADA0S - ADC channel specification register 0

The ADA0S register specifies the pin that inputs the analog voltage to be converted into a digital signal.

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

Reset input clears this register to 00H.

After reset: 00H R/W Address: FFFFF202H

	7	6	5	4	3	2	1	0
ADA0S	0	0	0	ADA0S4	ADA0S3	ADA0S2	ADA0S1	ADA0S0

ADA0S4	ADA0S3	ADA0S2	ADA0S1	ADA0S0	Select mode	Scan mode
0	0	0	0	0	ANI0	ANI0
0	0	0	0	1	ANI1	ANI0, ANI1
0	0	0	1	0	ANI2	ANI0 to ANI2
0	0	0	1	1	ANI3	ANI0 to ANI3
0	0	1	0	0	ANI4	ANI0 to ANI4
0	0	1	0	1	ANI5	ANI0 to ANI5
0	0	1	1	0	ANI6	ANI0 to ANI6
0	0	1	1	1	ANI7	ANI0 to ANI7
0	1	0	0	0	ANI8	ANI0 to ANI8
0	1	0	0	1	ANI9	ANI0 to ANI9
0	1	0	1	0	ANI10	ANI0 to ANI10
0	1	0	1	1	ANI11	ANI0 to ANI11
0	1	1	0	0	ANI12	ANI0 to ANI12
0	1	1	0	1	ANI13	ANI0 to ANI13
0	1	1	1	0	ANI14	ANI0 to ANI14
0	1	1	1	1	ANI15	ANI0 to ANI15
Other than above					Setting prohibited	

(5) ADCR0n, ADCR0Hn - ADC conversion result registers

The ADCR0n and ADCR0Hn registers store the A/D conversion results.

These registers are read-only, in 16-bit or 8-bit units. However, specify the ADCR0n register for 16-bit access and the ADCR0Hn register for 8-bit access. The 10 bits of the conversion result are read from the higher 10 bits of the ADCR0n register, and 0 is read from the lower 6 bits. The higher 8 bits of the conversion result are read from the ADCR0Hn register.

After reset: undefined R

Address: ADCR00	FFFFF210H,	ADCR01	FFFFF212H
ADCR02	FFFFF214H,	ADCR03	FFFFF216H
ADCR04	FFFFF218H,	ADCR05	FFFFF21AH
ADCR06	FFFFF21CH,	ADCR07	FFFFF21EH
ADCR08	FFFFF220H,	ADCR09	FFFFF222H
ADCR010	FFFFF224H,	ADCR011	FFFFF226H
ADCR012	FFFFF228H,	ADCR013	FFFFF22AH
ADCR014	FFFFF22CH,	ADCR015	FFFFF22EH

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCR0n	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	0	0	0	0	0	0

After reset: undefined R

Address: ADCR0H0	FFFFF211H,	ADCR0H1	FFFFF213H
ADCR0H2	FFFFF215H,	ADCR0H3	FFFFF217H
ADCR0H4	FFFFF219H,	ADCR0H5	FFFFF21BH
ADCR0H6	FFFFF21DH,	ADCR0H7	FFFFF21FH
ADCR0H8	FFFFF221H,	ADCR0H9	FFFFF223H
ADCR0H10	FFFFF225H,	ADCR0H11	FFFFF227H
ADCR0H12	FFFFF229H,	ADCR0H13	FFFFF22BH
ADCR0H14	FFFFF22DH,	ADCR0H15	FFFFF22FH

	7	6	5	4	3	2	1	0
ADCR0Hn	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2

The relationship between the analog voltage input to the analog input pins (ANI0 to ANI11) and the A/D conversion result (of A/D conversion result register n (ADCR0n)) is as follows:

$$ADCR0 \;=\; INT(\frac{V_{IN}}{AV_{REF}} \bullet 1024 + 0{,}5)$$

or

$$(ADCR0 - 0{,}5) \bullet \frac{AV_{REF}}{1024} \le V_{IN} < (ADCR0 + 0{,}5) \bullet \frac{AV_{REF}}{1024}$$

INT( ): Function that returns the integer of the value in ( )

V_{IN}: Analog input voltage

AV_{REF}: AV_{REF} pin voltage

ADCR0: Value of A/D conversion result register n (ADCR0n)

Figure 20-2 shows the relationship between the analog input voltage and the A/D conversion results.

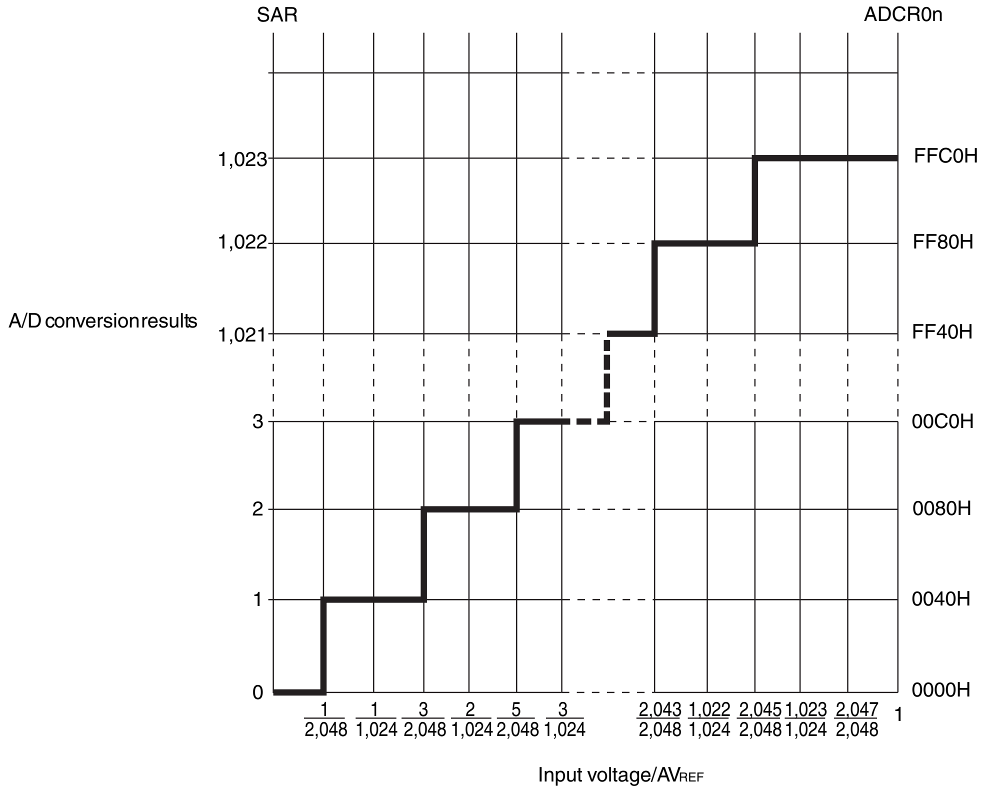


Figure 20-2 Relationship between analog input voltage and A/D conversion results

(6) ADA0PFM - ADC power-fail compare mode register

The ADA0PFM register is an 8-bit register that sets the power-fail compare mode.

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

After reset: 00H R/W Address: FFFFF204H

	7	6	5	4	3	2	1	0
ADA0PFM	ADA0PFE	ADA0PFC0	0	0	0	0	0	0

ADA0PFE	Power-fail compare enable/disable
0	Power-fail compare disabled
1	Power-fail compare enabled

ADA0PFC	Power-fail compare mode
0	Generates interrupt request INTAD if ADA0CRnH ≥ ADA0PFT
1	Generates interrupt request INTAD if ADA0CRnH < ADA0PFT

Note In continuous select mode the conversion result of ADC channel ANIn, selected by ADA0S, is observed.
In continuous scan mode the conversion result of ADC channel ANI0 is observed.
For further details, refer to *“Power-fail compare mode“ on page 785*.

(7) ADA0PFT - ADC power-fail compare threshold value register

The ADA0PFT 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.

After reset: 00H R/W Address: FFFFF205H

	7	6	5	4	3	2	1	0
ADA0PFT								

20.4 Operation

20.4.1 Basic operation

1. Set the operation mode, trigger mode, and conversion time for executing A/D conversion by using the ADA0M0, ADA0M1, ADA0M2, and ADA0S registers. When the ADA0CE bit of the ADA0M0 register is set, conversion is started in the software trigger mode and the A/D Converter waits for a trigger in the external or timer trigger mode.
2. When A/D conversion is started, the voltage input to the selected analog input channel is sampled by the sample & hold circuit.
3. When the sample & hold circuit samples the input channel for a specific time, it enters the hold status, and holds the input analog voltage until A/D conversion is complete.
4. Set bit 9 of the successive approximation register (SAR). The tap selector selects (1/2) AV_{REF} as the voltage tap of the series resistor string.
5. The voltage difference between the voltage of the series resistor string and the analog input voltage is compared by the voltage comparator. If the analog input voltage is higher than (1/2) AV_{REF}, the MSB of the SAR register remains set. If it is lower than (1/2) AV_{REF}, the MSB is reset.
6. Next, bit 8 of the SAR register is automatically set and the next comparison is started. Depending on the value of bit 9, to which a result has been already set, the voltage tap of the series resistor string is selected as follows:

   –Bit 9 = 1: (3/4) AV_{REF}

   –Bit 9 = 0: (1/4) AV_{REF}

   This voltage tap and the analog input voltage are compared and, depending on the result, bit 8 is manipulated as follows.

   Analog input voltage ≥ Voltage tap: Bit 8 = 1

   Analog input voltage ≤ Voltage tap: Bit 8 = 0
7. This comparison is continued to bit 0 of the SAR register.
8. When comparison of the 10 bits is complete, the valid digital result is stored in the SAR register, which is then transferred to and stored in the ADCR0n register. At the same time, an A/D conversion end interrupt request signal (INTAD) is generated.

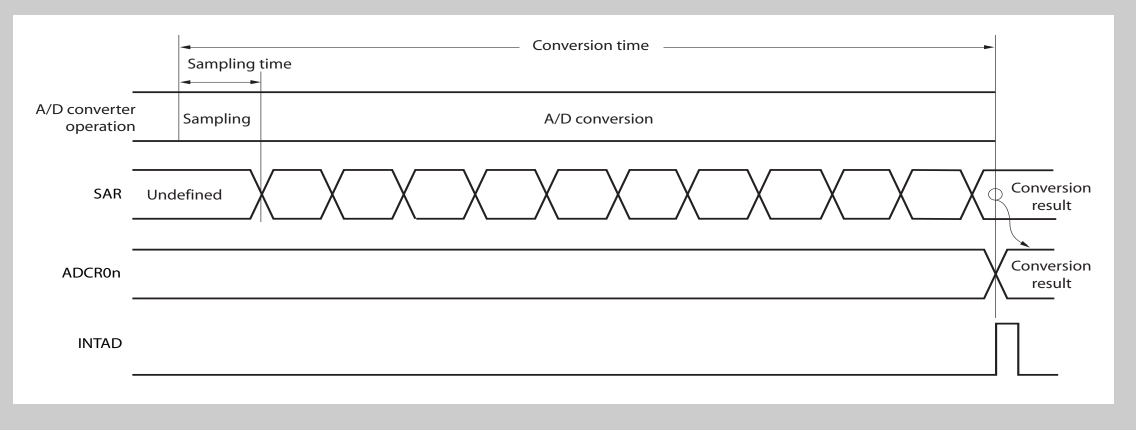


Figure 20-3 A/D Converter basic operation

20.4.2 Trigger mode

The timing of starting the conversion operation is specified by setting a trigger mode. The trigger mode includes a software trigger mode and hardware trigger modes. The hardware trigger modes include timer trigger modes 0 and 1, and external trigger mode. The ADA0TMD bit of the ADA0M0 register is used to set the trigger mode. In timer trigger mode set ADA0M2.ADA0TMD[1:0] = 01.

(1) Software trigger mode

When the ADA0CE bit of the ADA0M0 register is set to 1, the signal of the analog input pin ANIn specified by the ADA0S register is converted. When conversion is complete, the result is stored in the ADCR0n register. At the same time, the A/D conversion end interrupt request signal (INTAD) is generated.

If the operation mode specified by the ADA0MD1 and ADA0MD0 bits of the ADA0M0 register is the continuous select/scan mode, the next conversion is started, unless the ADA0CE bit is cleared to 0 after completion of the first conversion.

When conversion is started, the ADA0EF bit is set to 1 (indicating that conversion is in progress).

If the ADA0M0, ADA0M2, ADA0S, ADA0PFM, or ADA0PFT register is written during conversion, the conversion is aborted and started again from the beginning.

(2) Timer trigger mode

In this mode, converting the signal of the analog input pin ANIn specified by the ADA0S register is started by the Timer Z underflow interrupt signal.

Make sure to set ADA0M2.ADA0TMD[1:0] = 01_B.

When conversion is completed, the result of the conversion is stored in the ADCR0n register. At the same time, the A/D conversion end interrupt request signal (INTAD) is generated, and the A/D Converter waits for the trigger again.

When conversion is started, the ADA0EF bit is set to 1 (indicating that conversion is in progress). While the A/D Converter is waiting for the trigger, however, the ADA0EF bit is cleared to 0 (indicating that conversion is stopped). If the valid trigger is input during the conversion operation, the conversion is aborted and started again from the beginning.

If the ADA0M0, ADA0M2, ADA0S, ADA0PFM, or ADA0PFT register is written during conversion, the conversion is stopped and the A/D Converter waits for the trigger again.

20.4.3 Operation modes

Two operation modes are available as the modes in which to set the ANIn pins: continuous select mode and continuous scan mode.

The operation mode is selected by the ADA0MD1 and ADA0MD0 bits of the ADA0M0 register.

(1) Continuous select mode

In this mode, the voltage of one analog input pin selected by the ADA0S register is continuously converted into a digital value.

The conversion result is stored in the ADCR0n register corresponding to the analog input pin. In this mode, an analog input pin corresponds to an ADCR0n register on a one-to-one basis. Each time A/D conversion is completed, the A/D conversion end interrupt request signal (INTAD) is generated. After completion of conversion, the next conversion is started, unless the ADA0CE bit of the ADA0M0 register is cleared to 0.

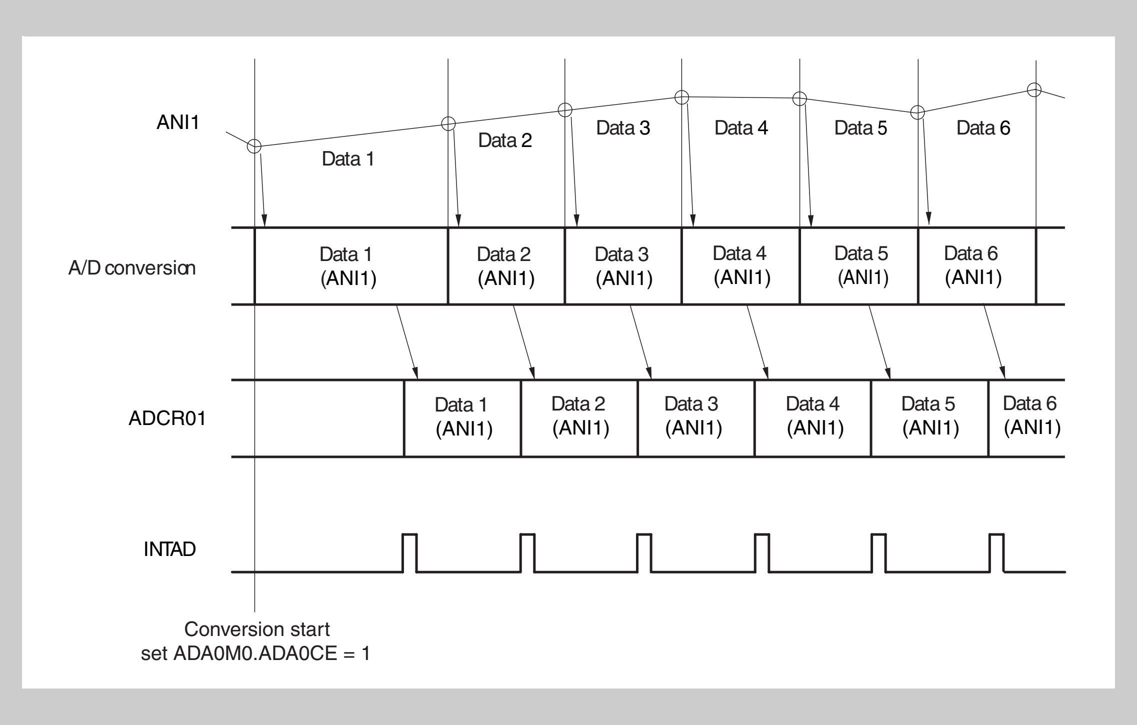


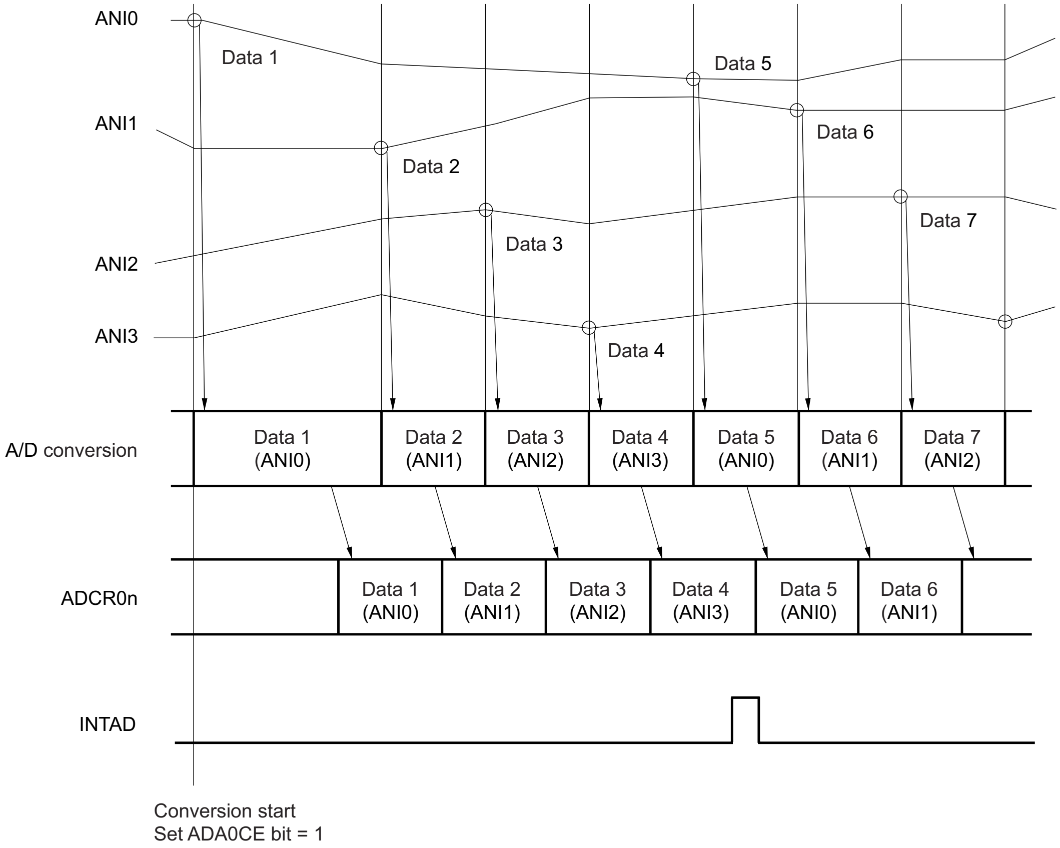
Figure 20-4 Timing example of continuous select mode operation (ADA0S = 01H)

(2) Continuous scan mode

In this mode, analog input pins are sequentially selected, from the ANI0 pin to the pin specified by the ADA0S register, and their values are converted into digital values.

The result of each conversion is stored in the ADCR0n register corresponding to the analog input pin. When conversion of the analog input pin specified by the ADA0S register is complete, the A/D conversion end interrupt request signal (INTAD) is generated, and A/D conversion is started again from the ANI0 pin, unless the ADA0CE bit of the ADA0M0 register is cleared to 0.

(a) Timing example



(b) Block diagram

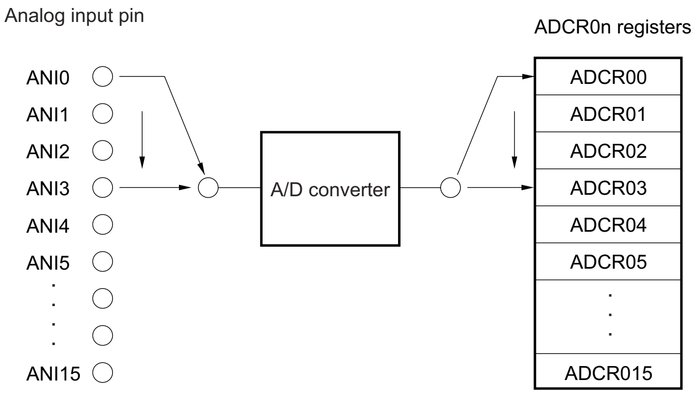


Figure 20-5 Timing example of continuous scan mode operation (ADA0S register = 03H)

20.4.4 Power-fail compare mode

The A/D conversion end interrupt request signal (INTAD) can be controlled as follows by the ADA0PFM and ADA0PFT registers.

- If the power-fail compare mode is disabled (ADA0PFM.ADA0PFE = 0), the INTAD signal is generated each time conversion is completed.
- If the power-fail compare mode is enabled (ADA0PFM.ADA0PFE = 1) and ADA0PFM.ADA0PFC = 0, the value of the ADCR0Hn register is compared with the value of the ADA0PFT register when conversion is completed, and the INTAD signal is generated only if ADCR0H0 ≥ ADA0PFT.
- If the power-fail compare mode is enabled (ADA0PFM.ADA0PFE = 1) and ADA0PFM.ADA0PFC = 1, the value of the ADCR0Hn register is compared with the value of the ADA0PFT register when conversion is completed, and the INTAD signal is generated only if ADCR0H0 < ADA0PFT.

In the power-fail compare mode, two modes are available as modes in which to set the ANIn pins: continuous select mode and continuous scan mode.

(1) Continuous select mode

In this mode, the higher 8 bits of conversion result of the ANIn channel in ADA0CR0Hn, specified by ADA0S, is compared with the value of the ADA0PFT register.

If the result of power-fail comparison matches the condition set by the ADA0PFM.ADA0PFC bit, INTAD is generated.

In any case the next conversion is started.

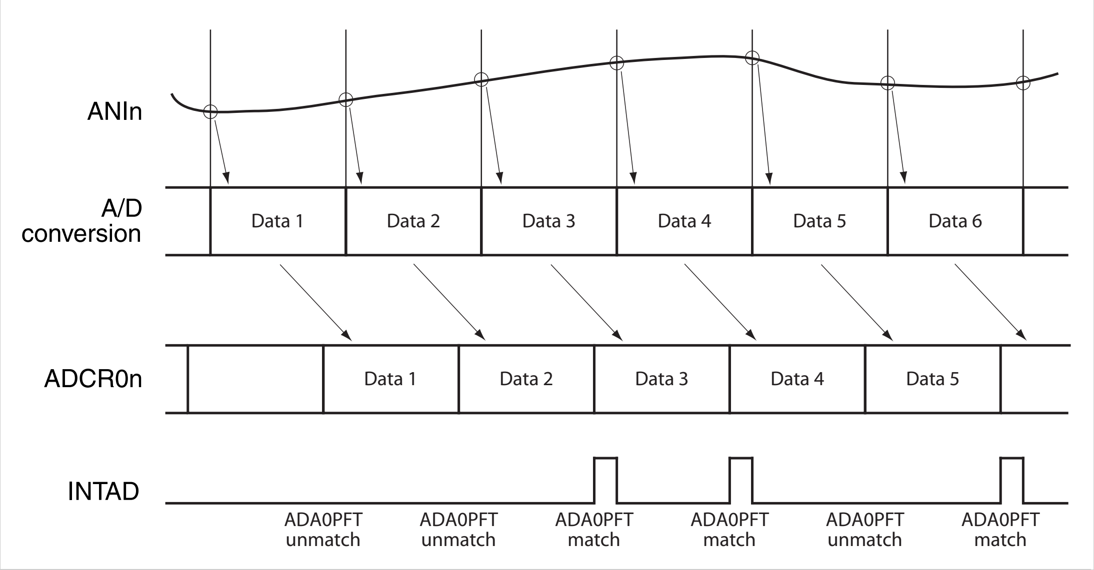


Figure 20-6 Timing example of continuous select mode operation whit power-fail comparison

(2) Continuous scan mode

In this mode, the ADC channels starting from ANI0 to the one specified by the ADA0S register are sequentially converted and the conversion results are stored in the ADCR0n registers.

Note In continuous scan mode power-fail comparison is performed only on ANI0.

After each conversion of ANI0, the higher 8 bits of conversion result in ADA0CR0H0 is compared with the value of the ADA0PFT register.

If the result of power-fail comparison matches the condition set by the ADA0PFM.ADA0PFC bit, INTAD is generated.

In any case conversion of the remaining ADC channels continuous.

Thus it is possible to catch a snapshot of the other analog inputs ANIn in case of power-fail.

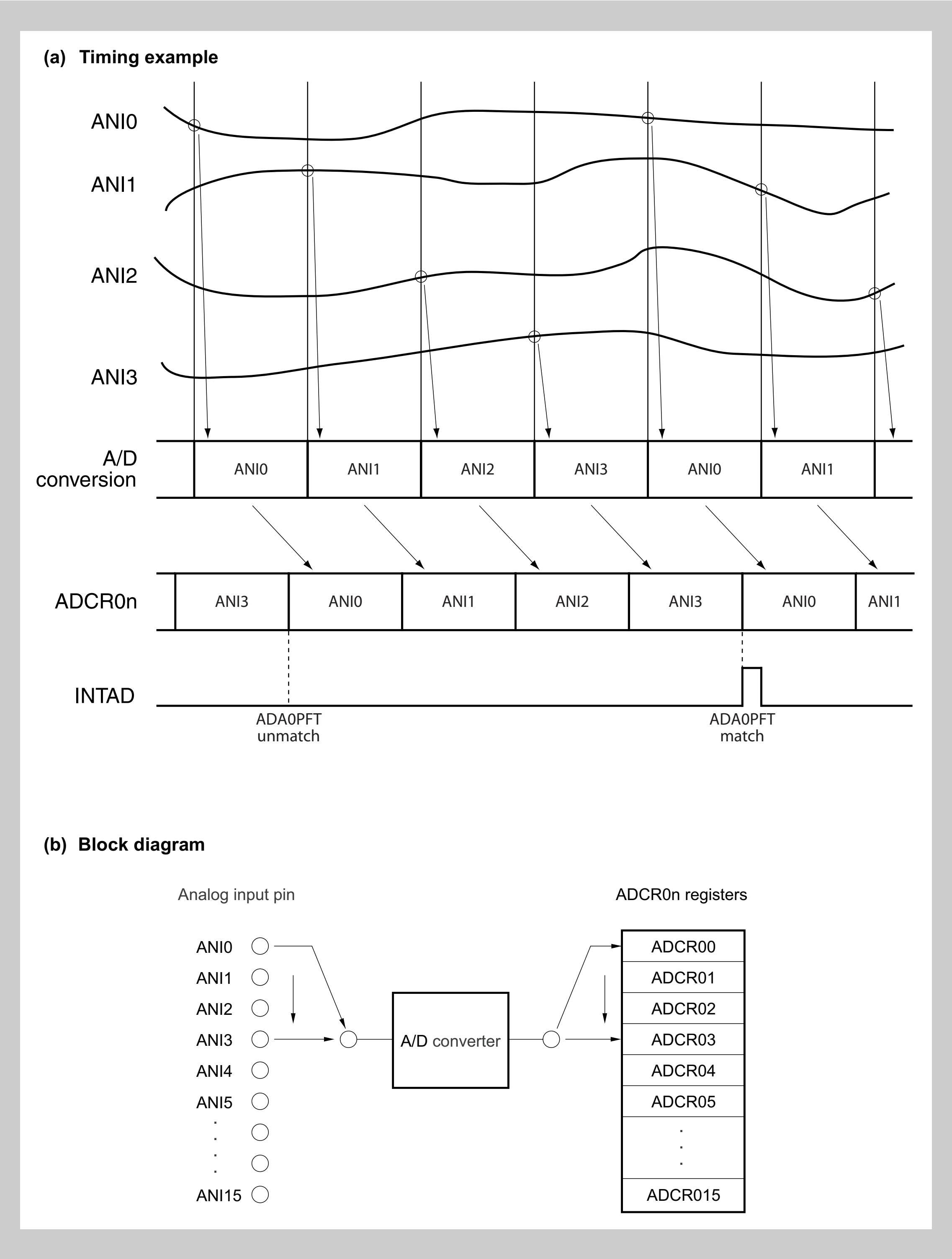


Figure 20-7 Timing example of continuous scan mode operation with power-fail comparison (ADA0S = 03_H)

20.5 Cautions

(1) When A/D Converter is not used

When the A/D Converter is not used, the power consumption can be reduced by clearing the ADA0CE bit of the ADA0M0 register to 0.

(2) Input range of ANIn pins

Input the voltage within the specified range to the ANIn pins. If a voltage equal to or higher than AV_{REF} or equal to or lower than AV_{SS} (even within the range of the absolute maximum ratings) is input to any of these pins, the conversion value of that channel is undefined.

(3) Countermeasures against noise

To maintain the 10-bit resolution, the ANIn pins must be effectively protected from noise. The influence of noise increases as the output impedance of the analog input source becomes higher. To lower the noise, connecting an external capacitor as shown in *Figure 20-8* is recommended.

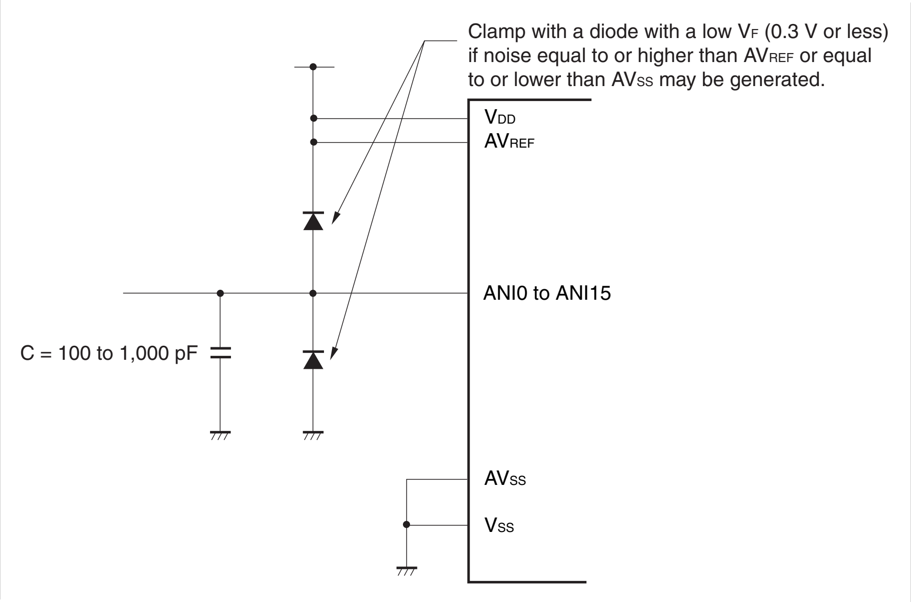


Figure 20-8 Processing of analog input pin

(4) Alternate I/O

The analog input pins ANIn function alternately as port pins. When selecting one of the ANIn pins to execute A/D conversion, do not execute an instruction to read an input port or write to an output port during conversion as the conversion resolution may drop.

If a digital pulse is applied to a pin adjacent to the pin whose input signal is being converted, the A/D conversion value may not be as expected due to the influence of coupling noise. Therefore, do not apply a pulse to a pin adjacent to the pin undergoing A/D conversion.

(5) Interrupt request flag (ADIF)

The interrupt request flag (ADIF) is not cleared even if the contents of the ADA0S register are changed. If the analog input pin is changed during A/D conversion, therefore, the result of converting the previously selected analog input signal may be stored and the conversion end interrupt request flag may be set immediately before the ADA0S register is rewritten. If the ADIF flag is read immediately after the ADA0S register is rewritten, the ADIF flag may be set even though the A/D conversion of the newly selected analog input pin has not been completed. When A/D conversion is stopped, clear the ADIF flag before resuming conversion.

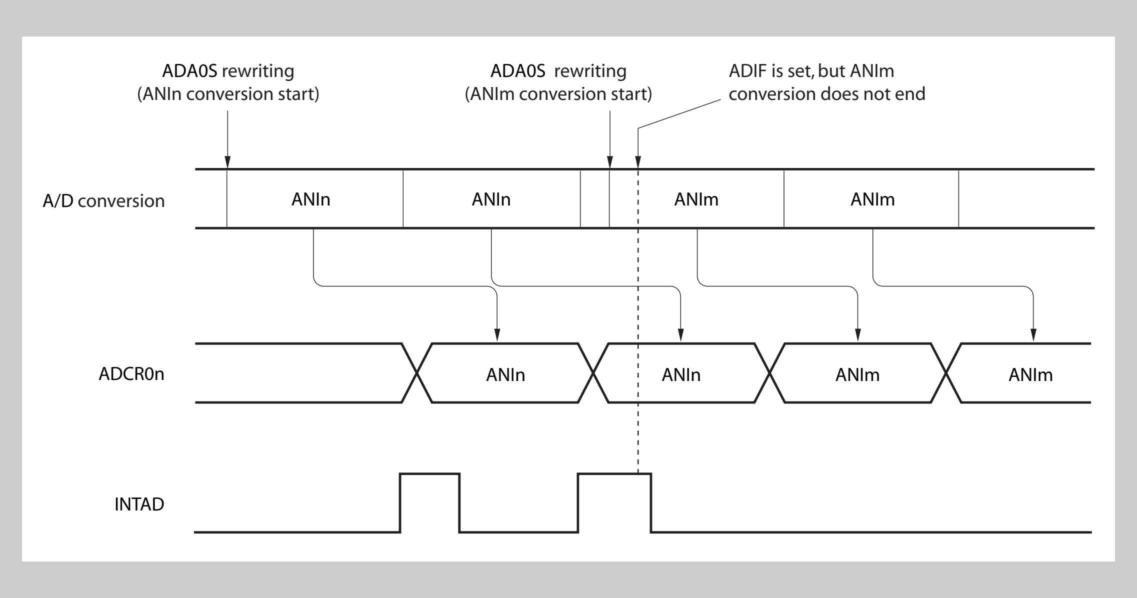


Figure 20-9 Generation timing of A/D conversion end interrupt request

(6) Reading ADCR0n register

When the ADA0M0 to ADA0M2 or ADA0S register is written, the contents of the ADCR0n register may be undefined. Read the conversion result after completion of conversion and before writing to the ADA0M0 to ADA0M2 and ADA0S registers. The correct conversion result may not be read at a timing different from the above.

20.6 How to Read A/D Converter Characteristics Table

This section describes the terms related to the A/D Converter.

(1) Resolution

The minimum analog input voltage that can be recognized, i.e., the ratio of an analog input voltage to 1 bit of digital output is called 1 LSB (least significant bit). The ratio of 1 LSB to the full scale is expressed as %FSR (full-scale range). %FSR is the ratio of a range of convertible analog input voltages expressed as a percentage, and can be expressed as follows, independently of the resolution.

1%FSR = (Maximum value of convertible analog input voltage – Minimum value of convertible analog input voltage)/100
= $(AV_{REF} – 0)/100$
= $AV_{REF}/100$

When the resolution is 10 bits, 1 LSB is as follows:

1 LSB = $1/2^{10} = 1/1,024$
= 0.098%FSR

The accuracy is determined by the overall error, independently of the resolution.

(2) Overall error

This is the maximum value of the difference between an actually measured value and a theoretical value. It is a total of zero-scale error, full-scale error, linearity error, and a combination of these errors.

The overall error in the characteristics table does not include the quantization error.

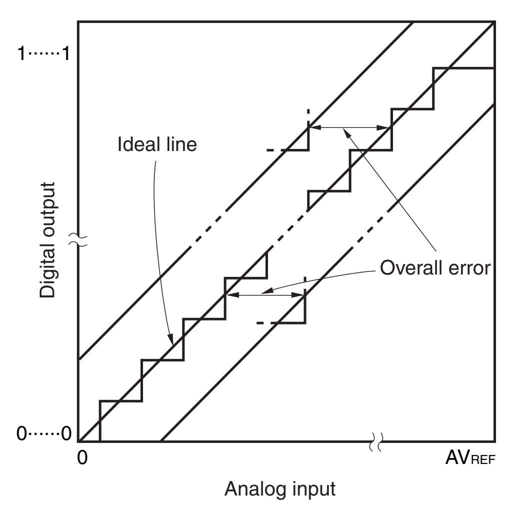


Figure 20-10 Overall error

(3) Quantization error

This is an error of ±1/2 LSB that inevitably occurs when an analog value is converted into a digital value. Because the A/D Converter converts analog input voltages in a range of ±1/2 LSB into the same digital codes, a quantization error is unavoidable.

This error is not included in the overall error, zero-scale error, full-scale error, integral linearity error, or differential linearity error in the characteristics table.

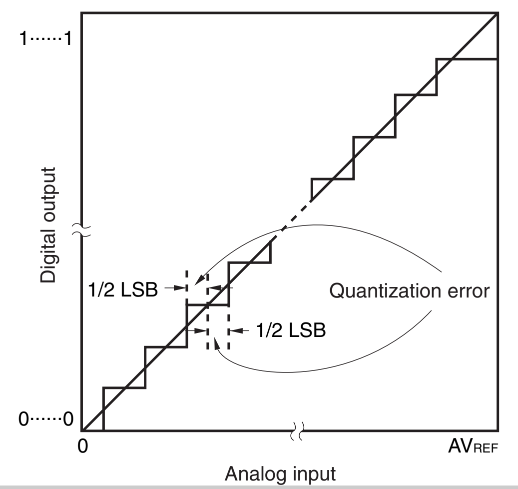


Figure 20-11 Quantization error

(4) Zero-scale error

This is the difference between the actually measured analog input voltage and its theoretical value when the digital output changes from 0…000 to 0…001 (1/2 LSB).

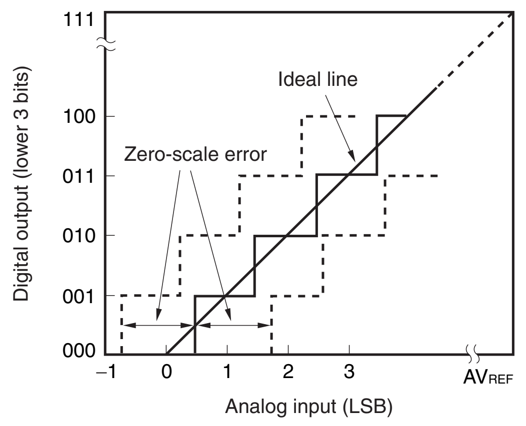


Figure 20-12 Zero-scale error

(5) Full-scale error

This is the difference between the actually measured analog input voltage and its theoretical value when the digital output changes from 1…110 to 0…111 (full scale - 3/2 LSB).

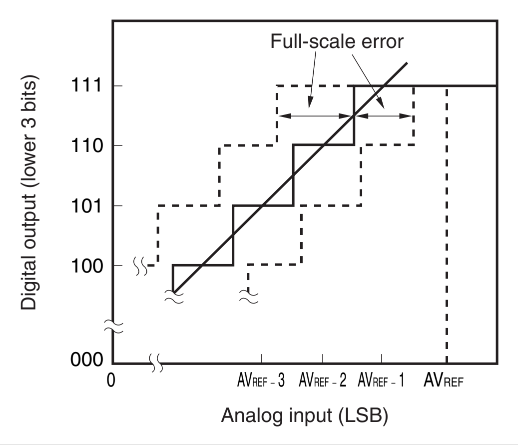


Figure 20-13 Full-scale error

(6) Differential linearity error

Ideally, the width to output a specific code is 1 LSB. This error indicates the difference between the actually measured value and its theoretical value when a specific code is output.

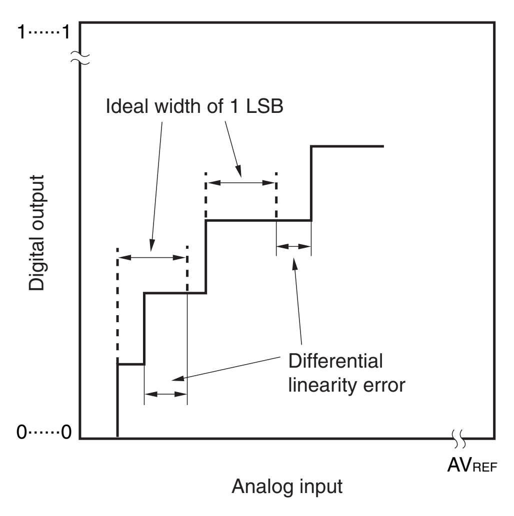


Figure 20-14 Differential linearity error

(7) Integral linearity error

This error indicates the extent to which the conversion characteristics differ from the ideal linear relationship. It indicates the maximum value of the difference between the actually measured value and its theoretical value where the zero-scale error and full-scale error are 0.

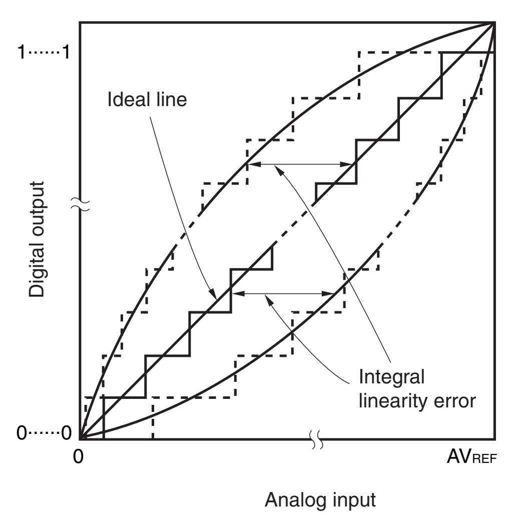


Figure 20-15 Integral linearity error

(8) Conversion time

This is the time required to obtain a digital output after an analog input voltage has been assigned.

The conversion time in the characteristics table includes the sampling time.

(9) Sampling time

This is the time for which the analog switch is ON to load an analog voltage to the sample & hold circuit.

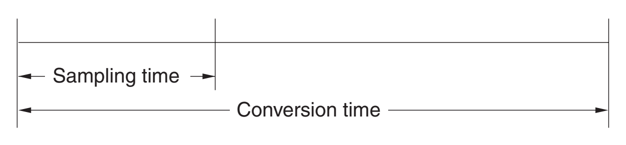


Figure 20-16 Sampling time

Chapter 21 Stepper Motor Controller/Driver (Stepper-C/D)

The Stepper Motor Controller/Driver module is comprised of six drivers (k = 1 to 6) for external 360° type meters or for bipolar and unipolar stepper motors. The V850E/Dx3 microcontrollers have following instances of the Stepper Motor Controller/Driver:

Stepper-C/D	All devices
Instances	1

Throughout this chapter, the individual instances of Stepper-C/D are identified by “n”, for example MCNTCn0, or MCNTCn1 for the timer mode control registers.

The Stepper Motor Controller/Driver module can be separated into two sub-modules. Throughout this chapter, the individual sub-modules are identified by “m” (m = 0, 1).

21.1 Overview

The Stepper Motor Controller/Driver module generates pulse width modulated (PWM) output signals. Each driver generates up to four output signals.

Features summary

The generated output signals have the following features:

- Pulse width of 8 bits precision
- 1-bit addition function enables an average pulse width precision of 1/2 bit, resulting in a pseudo 9-bit precision
- PWM frequency up to 32 KHz
- automatic PWM phase shift for reducing fluctuation on power supply and for reducing the susceptibility to electromagnetic interference

21.1.1 Driver overview

A stepper motor is driven by PWM signals. The PWM signals are generated by comparing the contents of compare registers with the actual value of a free running up counter.

The Stepper Motor Controller/Driver module can be separated into two sub-modules - each sub-module contains one counter and assigned compare registers and control registers. In the following, the two sub-modules are called Stepper Motor Controller/Driver 0 sub-module and Stepper Motor Controller/Driver 1 sub-module.

The following figures show the main components of the Stepper Motor Controller/Driver 0 sub-module (*Figure 21-1*) and of the Stepper Motor Controller/Driver 1 sub-module (*Figure 21-2*).

The Stepper Motor Controller/Driver 0 sub-module is comprised of 4 drivers (k = 1 to 4), Stepper Motor Controller/Driver 1 sub-module is comprised of 2 drivers (k = 5 to 6). Each Stepper Motor Controller/Driver sub-module includes a free running up counter (CNTm). The counter is controlled by a timer mode control register (MCNTCnm).

Each of the six drivers consists of two compare registers, MCMPnk0 and MCMPnk1, respectively. Their contents define the pulse widths for the sine and the cosine side of the meters. The MCMPnk0/MCMPnk1 registers comprise a master-slave register combination. This allows to re-write the master register while the slave register is actually used for comparison with the counter CNTm.

The compare control register MCMPCnk defines whether or not enhanced pulse width precision by one-bit addition is enabled, and it routes the output signals to the corresponding output pins (SMk1 to SMk4).

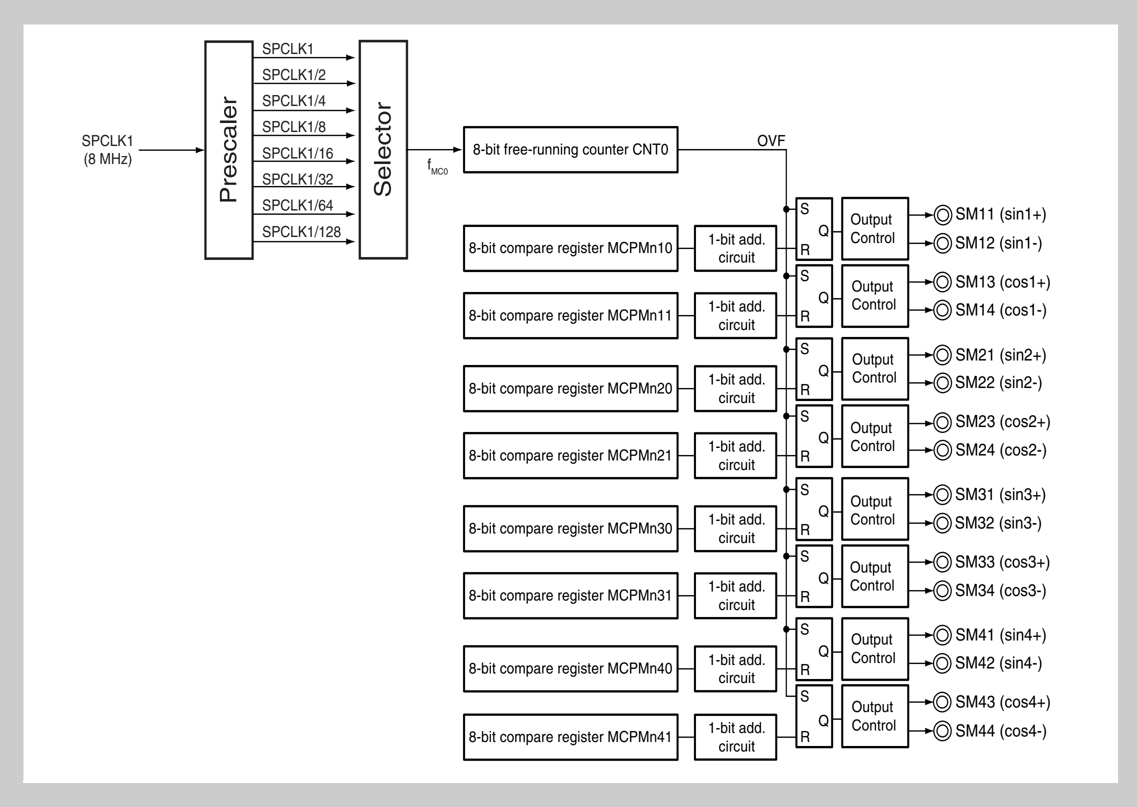


Figure 21-1 Stepper Motor Controller/Driver 0 block diagram

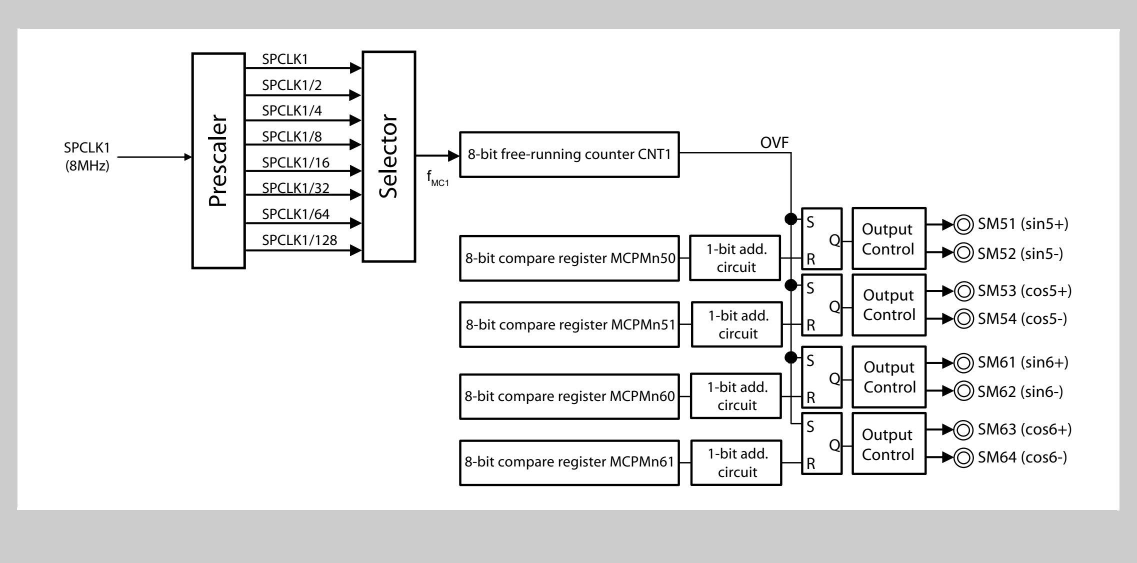


Figure 21-2 Stepper Motor Controller/Driver 1 block diagram

The external signals are listed in the following table.

Table 21-1 Stepper Motor Controller/Driver external connections

Signal name	I/O	Active level	Reset level	Pins	Function
SM[1:6]1	O	–	L	SM11 to SM61	driver signal, sine side (+)
SM[1:6]2	O	–	L	SM12 to SM62	driver signal, sine side (–)
SM[1:6]3	O	–	L	SM13 to SM63	driver signal, cosine side (+)
SM[1:6]4	O	–	L	SM14 to SM64	driver signal, cosine side (–)

21.2 Stepper Motor Controller/Driver Registers

The Stepper Motor Controller/Driver is controlled and operated by means of the following registers:

Table 21-2 Stepper Motor Controller/Driver registers overview (1/2)

Register name	Shortcut	Address
Timer mode control registers	MCNTCn0	<base>
	MCNTCn1	<base> + 14_H

Table 21-2 Stepper Motor Controller/Driver registers overview (2/2)

Register name	Shortcut	Address
Compare registers	MCMPnk0 (k = 1 to 6)	<base> + 2_H, 4_H, 6_H, 8_H, 16_H, 18_H
	MCMPnk1 (k = 1 to 6)	<base> + 3_H, 5_H, 7_H, 9_H, 17_H, 19_H
	MCMPnkHW (k = 1 to 6)	<base> + 2_H, 4_H, 6_H, 8_H, 16_H, 18_H
Compare control registers	MCMPCnk (k = 1 to 6)	<base> + A_H, C_H, E_H, 10_H, $1A_H$, $1C_H$

The base address of the Stepper Motor Controller/Driver is <base> = FFFF $F5C0_H$.

(1) MCNTCn0, MCNTCn1 - Timer mode control registers

The 8-bit MCNTCnm registers control the operation of the free running up counters CNTm.

Access These registers can be read/written in 8-bit or 1-bit units.

Address MCNTCn0: <base>
MCNTCn1: <base> + 14_H

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

7	6	5	4	3	2	1	0
CAE[a]	0	FULL	PCE	0	SMCL2	SMCL1	SMCL0
R/W[b]	R	R/W	R/W	R	R/W	R/W	R/W

[a] Bit CAE refers only to register MCNTCn0. In register MCNTCn1, this bit is set to 0.
[b] In register MCNTCn0, this bit is read only (R)

Table 21-3 MCNTCnm register contents

Bit position	Bit name	Function
7	CAE[a]	Stepper Motor Controller/Driver control 0: Stepper Motor Controller/Driver operation is disabled. 1: Stepper Motor Controller/Driver operation is enabled. This bit switches both Stepper Motor Controller/Driver 0 and Stepper Motor Controller/Driver 1.
5	FULL	Sets the count range of the timer counter 0: count range from 01_H to FF_H 1: count range from 00_H to FF_H The initial start value is 00_H in both cases. For the impact of this bit on duty factor and PWM cycle time, see also *"Duty Factor" on page 804*.
4	PCE	Timer operation control 0: Timer counteris stopped. 1: Timer counter is enabled.
2 to 0	SMCL[2:0]	Sets the timer count clock for the timer counter

SMCL2	SMCL1	SMCL0	Selected timer count clock
0	0	0	SPCLK1
0	0	1	SPCLK1 / 2
0	1	0	SPCLK1 / 4
0	1	1	SPCLK1 / 8
1	0	0	SPCLK1 / 16
1	0	1	SPCLK1 / 32
1	1	0	SPCLK1 / 64
1	1	1	SPCLK1 / 128

[a] Bit CAE refers only to register MCNTCn0. In register MCNTCn1, this bit is set to 0.

Caution In register MCNTCn0, bits 3 and 6 must be 0.
In register MCNTCn1, bits 3, 6 and 7 must be 0.

Power save mode preparation Before entering any power save mode the Stepper-C/D must be shut down in advance in order to minimize power consumption.

Apply following sequence to shut down the Stepper-C/D:

1. Stop the counter CNT1 by setting MCNTCn1.PCE = 0.
2. Stop the counter CNT0 by setting MCNTCn0.PCE = 0.
3. Disable the Stepper-C/D operation by setting MCNTCn0.CAE = 0.

Note that the MCNTCn0.PCE and MCNTCn0.CAE bits must not be cleared to 0 by a single write instruction. Perform two write instructions as shown above.

(2) MCMPnk0 - Compare registers for sine side (k = 1 to 6)

The 8-bit MCMPnk0 registers hold the values that define the PWM pulse width for the sine side of the connected meters.

The contents of the registers are continuously compared to the timer counter value:

- Registers MCMP10 to MCMP40 are compared to CNT0.
- Registers MCMP50 to MCMP60 are compared to CNT1.

When the register contents match the timer counter contents, a match signal is generated. Thus a PWM pulse with a pulse width corresponding to the MCMPnk0 register contents is output to the sine side of the connected meter.

Access These registers can be read/written in 8-bit units.

Address <base> + 2_H, 4_H, 6_H, 8_H, 16_H, 18_H

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

7	6	5	4	3	2	1	0
sine DATA							
R/W							

Note

1. New data must only be written to registers MCMPnk0 if the corresponding bit MCMPCnk.TEN = 0.
2. Don't write to the compare register MCMPnk0, until the corresponding bit MCMPCnk.TEN has been reset to 0 automatically.
3. To enable master-to-slave register copy upon next CNTm overflow set MCMPCnk.TEN = 1.

(3) MCMPnk1 - Compare registers for cosine side (k = 1 to 6)

The 8-bit MCMPnk1 registers hold the values that define the PWM pulse width for the cosine side of the connected meters.

The contents of the registers are continuously compared to the timer counter value:

- Registers MCMP11 to MCMP41 are compared to CNT0.
- Registers MCMP51 to MCMP61 are compared to CNT1.

When the register contents match the timer counter contents, a match signal is generated. Thus a PWM pulse with a pulse width corresponding to the MCMPnk1 register contents is output to the sine side of the connected meter.

Access These registers can be read/written in 8-bit units.

Address <base> + 3_H, 5_H, 7_H, 9_H, 17_H, 19_H

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

7	6	5	4	3	2	1	0
cosine DATA							
R/W							

Note

1. New data must only be written to registers MCMPnk1 if the corresponding bit MCMPCnk.TEN = 0.
2. Don't write to the compare register MCMPnk1, until the corresponding bit MCMPCnk.TEN has been reset to 0 automatically.
3. To enable master-to-slave register copy upon next CNTm overflow set MCMPCnk.TEN = 1.

(4) MCMPnkHW - Combined compare registers (k = 1 to 6)

The 16-bit MCPMnkHW registers combine the sine and cosine registers MCMPnk0 and MCMPnk1. Via these registers it is possible to read or write the contents of MCMPnk0 and MCMPnk1 in a single instruction.

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

Address <base> + 2_H, 4_H, 6_H, 8_H, 16_H, 18_H

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cosine DATA								sine DATA							
R/W															

Note

1. New data must only be written to registers MCMPnk1 if the corresponding bit MCMPCnk.TEN = 0.
2. Don't write to the compare register MCMPnk1, until the corresponding bit MCMPCnk.TEN has been reset to 0 automatically.
3. To enable master-to-slave register copy upon next CNTm overflow set MCMPCnk.TEN = 1.

(5) MCMPCnk - Compare control registers (k = 1 to 6)

The 8-bit MCMPCnk registers control the operation of the corresponding compare registers and the output direction of the PWM pin.

Access These registers can be read/written in 8-bit units.

Address <base> + A_H, C_H, E_H, 10_H, $1A_H$, $1C_H$

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

7	6	5	4	3	2	1	0
AOUT	0[a]	0[b]	TEN	ADB1	ADB0	DIR1	DIR0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W

[a] Do not change this bit.
[b] This bit may be written, but writing is ignored.

Table 21-4 MCMPCnk register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>AOUT</td><td>Selects the output pins for sine and cosine signals
0: The PWM signals for sine and cosine side are output to those pins that are selected by bits DIR0 and DIR1. At all other pins, the output signal is 0 (SMV_{SS} level).
1: The PWM signal for the sine side is output to pins SMk1 and SMk2. The PWM signal for the cosine side is output to pins SMk3 and SMk4.</td></tr>
<tr><td>4</td><td>TEN</td><td>Transfer enable control bit
0: MCMPnk0/MCMPnk1 master-to-slave register copy is disabled. New data can be written to compare registers MCMPnk0 or MCMPnk1.
1: MCMPnk0/MCMPnk1 master-to-slave register copy is enabled. The copy process will take place when CNT0 or CNT1, respectively, overflows. Don't write to compare registers MCMPnk0 or MCMPnk1 while MCMPCnk.TEN = 1.
Note: This bit functions as a control bit and status flag. It is automatically reset to zero upon the next timer counter overflow.</td></tr>
<tr><td>3</td><td>ADB1</td><td>Sets 1-bit addition function for cosine side
0: no 1-bit addition to PWM signal
1: 1-bit addition to PWM signal</td></tr>
<tr><td>2</td><td>ADB0</td><td>Sets 1-bit addition function for sine side
0: no 1-bit addition to PWM signal
1: 1-bit addition to PWM signal</td></tr>
<tr><td>1 to 0</td><td>DIR[1:0]</td><td>Selects the output pins for the PWM signals.
Bits DIR1 and DIR0 address the quadrant to be activated by sine and cosine. The PWM signal is routed to the specific pin with respect to the sin/cos of each quadrant.
<table>
<tr><th>DIR1</th><th>DIR0</th><th>Selected output pins</th></tr>
<tr><td>0</td><td>0</td><td>Quadrant 1: SMk1 (sin +), SMk3 (cos +)</td></tr>
<tr><td>0</td><td>1</td><td>Quadrant 2: SMk1 (sin +), SMk4 (cos –)</td></tr>
<tr><td>1</td><td>0</td><td>Quadrant 3: SMk2 (sin –), SMk4 (cos –)</td></tr>
<tr><td>1</td><td>1</td><td>Quadrant 4: SMk2 (sin –), SMk3 (cos +)</td></tr>
</table>
At the other output pins, the output level is SMV_{SS}.
Note: These bits are only considered if bit AOUT is set to 0.</td></tr>
</table>

21.3 Operation

In the following, the operation of the Stepper Motor Controller/Driver module as a driver for external meters is described.

21.3.1 Stepper Motor Controller/Driver operation

This section describes the generation of PWM signals of the driver k for driving external meters. Further, the achievable duty factor is explained and how advanced precision can be gained by 1-bit addition.

(1) Driving Meters

External meters can be driven both in H-bridge configuration and in half bridge configuration:

- Driving meters in H-bridge configuration

  Deflection of the needle of a meter in H-bridge configuration is determined by the sine and cosine value of its desired angle. Since the PWM signals do not inherit a sign, separate signals for positive and negative sine and cosine values are generated.
  The four signals at pins SMk1 to SMk4 of the driver k are:
  – sine side, positive (sin +)
  – sine side, negative (sin –)
  – cosine side, positive (cos +)
  – cosine side, negative (cos –)

  Two output control circuits select which signal (sign) for sine side and cosine side is output (bits MCMPCnk.DIR[1:0]). At the remaining two output pins, the signal is set to low level.
  To drive meter k in full bridge mode, set bit MCMPCnk.AOUT to 0.

- Driving meters in half bridge configuration

  In this mode, the same signal is sent to both sine pins (SMk1 and SMk2) and both cosine pins (SMk3 and SMk4), respectively. The setting of output control bits MCMPCnk.DIR[1:0] is neglected.
  To drive meter k in half bridge mode, set bit MCMPCnk.AOUT to 1.

(2) Generation of PWM signals

Bit data corresponding to the length of the PWM pulses has to be written to the compare registers MCMPnk0 (sine side) and MCMPnk1 (cosine side).

A timer counter is counting up. The rising edge of the PWM pulse is initiated at the overflow of the counter. The falling edge of the PWM pulse is initiated when the counter value equals the contents of the compare register.

The absolute pulse length in seconds is defined by the timer count clock (f_{MC0} and f_{MC1}, respectively). Various cycle times can be set via the timer mode control registers MCNTCn0 and MCNTCn1.

Instruction When writing data to compare registers, proceed as follows:

1. Confirm that MCMPCnk.TEN = 0.
2. Write 8-bit PWM data to MCMPnk0 and MCMPnk1.
3. Set MCMPCnk.ADB0 and MCMPCnk.ADB1 as desired.
4. Set MCMPCnk.TEN = 1 to start the counting operation.

   The data in MCMPnk0/MCMPnk1 will automatically be copied to the compare slave register when the counter overflows. The new pulse width is valid immediately.

   Bit MCMPCnk.TEN is automatically cleared to 0 by hardware.

(3) Duty Factor

The minimum pulse width that can be generated is zero (output signal is low) and the maximum pulse width is 255 clock cycles (maximum value of 8-bit compare registers).

The count range of the timer counter defines the duty factor. It can be set by bit MCNTCnm.FULL:

- count range 01_H to FF_H (MCNTCnm.FULL = 0)

  Formula for the duty cycle:
  PWM duty = MCMPki / 255 with k = 1 to 6 and i = 0, 1

  One count cycle is comprised of 255 clock cycles. A PWM signal with maximum pulse length is a steady high level signal. The duty factor is 100%.

- count range 00_H to FF_H (MCNTCnm.FULL = 1)

  Formula for the duty cycle:
  PWM duty = MCMPki / 256 with k = 1 to 6 and i = 0, 1

  One count cycle is comprised of 256 clock cycles. A PWM signal with maximum pulse length is comprised of 255 clock cycles at high level and one clock cycle at low level. The duty factor is 255/256 *100% = 99.6%.

(4) Advanced precision by 1-bit addition

The precision of the angle of a needle is implicitly defined by the number of bits of the compare registers MCMPnk0 and MCMPnk1 (8 bit).

If the 1-bit addition circuit is enabled, every second pulse of the PWM signal is extended by one bit (one clock cycle). In average, a pulse width precision of 1/2 bit (1/2 clock) can be achieved.

The following figures show the timing of PWM output signals with 1-bit addition disabled and enabled. (See *“Advanced precision by 1-bit addition“ on page 804*)

Note

1. The PWM pulse is not generated until the first overflow occurs after the counting operation has been started.
2. The PWM signal is two cycle counts delayed compared to the overflow signal and the match signal. This is not depicted in the figures.

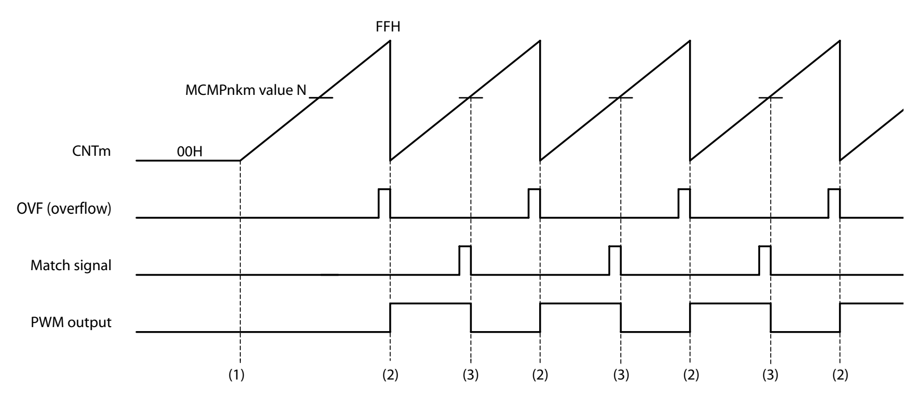


Figure 21-3 Output timing without 1-bit addition

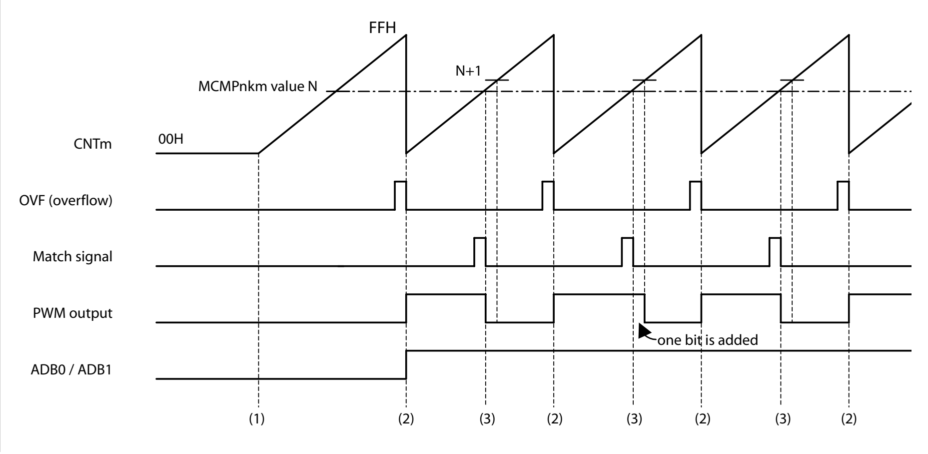


Figure 21-4 Output timing with 1-bit addition

Sequence

1. Start of counting (MCNTCnm.PCE is set to 1)
2. Generation of overflow signal (start of PWM pulse)
3. Generation of match signal (timer counter CNTm matches compare register, end of PWM pulse)

21.4 Timing

This section starts with the timing of the timer counter and general output timing behaviour. Then, examples of output signal generation with and without 1-bit addition are presented.

21.4.1 Timer counter

The free running up counter is clocked by the timer count clock selected in register MCNTCnm.

The counting operation is enabled or disabled by the MCNTCnm.PCE bit.

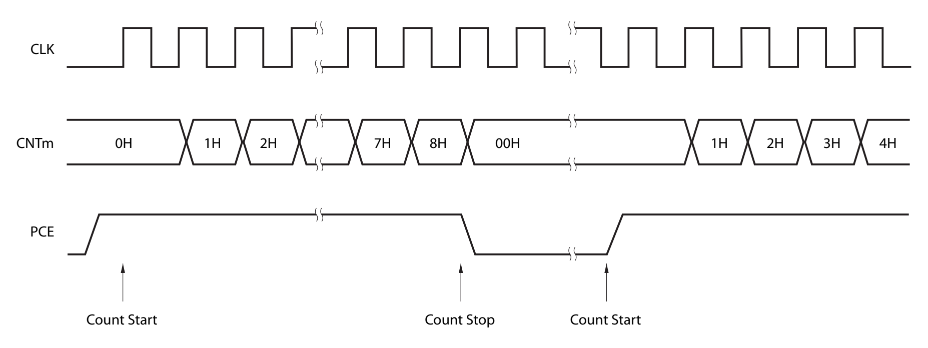


Figure 21-5 Restart Timing after Count Stop (Count Start—Count Stop—Count Start)

Sequence

- Count Start:
  - Enable counting operation (MCNTCnm.PCE = 1)
  - Timer counter starts with value 00_H. Depending on bit MCNTCnm.FULL, all following counter cycles start with 00_H or 01_H, respectively.
- Count Stop:
  - Disable counting operation (MCNTCnm.PCE = 0)
  - Counting is stopped and timer counter is set to 00_H.

21.4.2 Automatic PWM phase shift

Simultaneous switching of sine and cosine output could lead to a fluctuation of the power supply and increase the susceptibility to electromagnetic interference. To prevent this for drivers 1 to 4, the output signals are automatically shifted by one timer count clock cycle defined in MCNTCn0.

The same accounts for the output signals of drivers 5 and 6. They are controlled by the timer count clock defined in MCNTCn1.

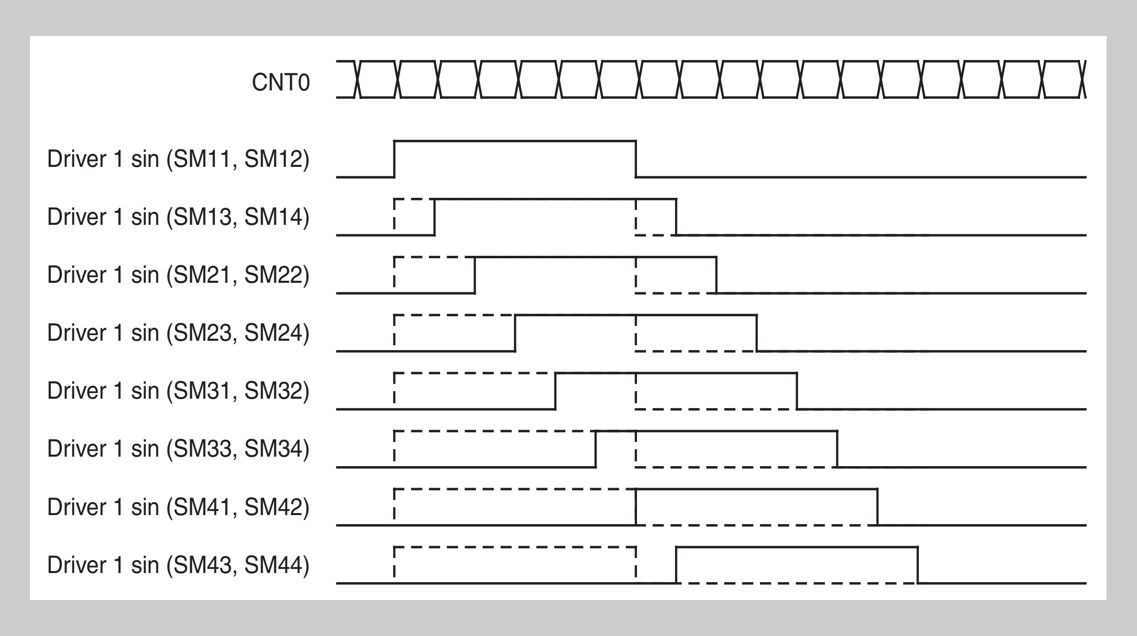


Figure 21-6 Output timing of signals SM11 to SM44

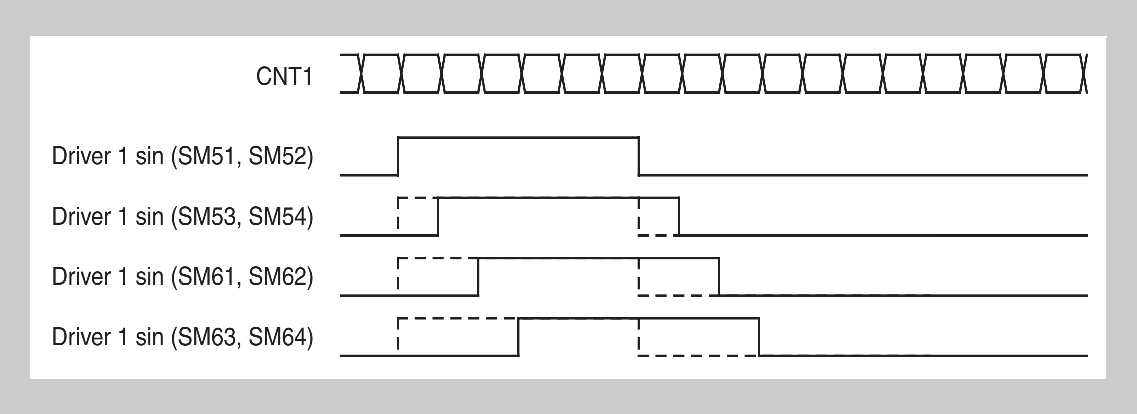


Figure 21-7 Output timing of signals SM51 to SM64

Chapter 22 LCD Controller/Driver (LCD-C/D)

The LCD Controller/Driver is provided with the µPD70(F)3420, µPD70(F)3421, µPD70(F)3422 and µPD70F3423 microcontrollers only.

This LCD Controller/Driver is suitable for LC displays with up to 160 segments. The supported addressing method of the LCD is multiplex addressing.

22.1 Overview

The LCD Controller/Driver generates the signals that are necessary for driving an LCD panel.

Features summary

The LCD Controller/Driver provides:

- Maximum of 40 segment signal outputs (SEG0 to SEG39)
- 4 common signal outputs (COM0 to COM3)
- Display mode: 1/4 duty (1/3 bias)
- Wide range of selectable frame frequencies
- Edge enhancement

22.1.1 Description

The following figure shows the main components of the LCD Controller/Driver:

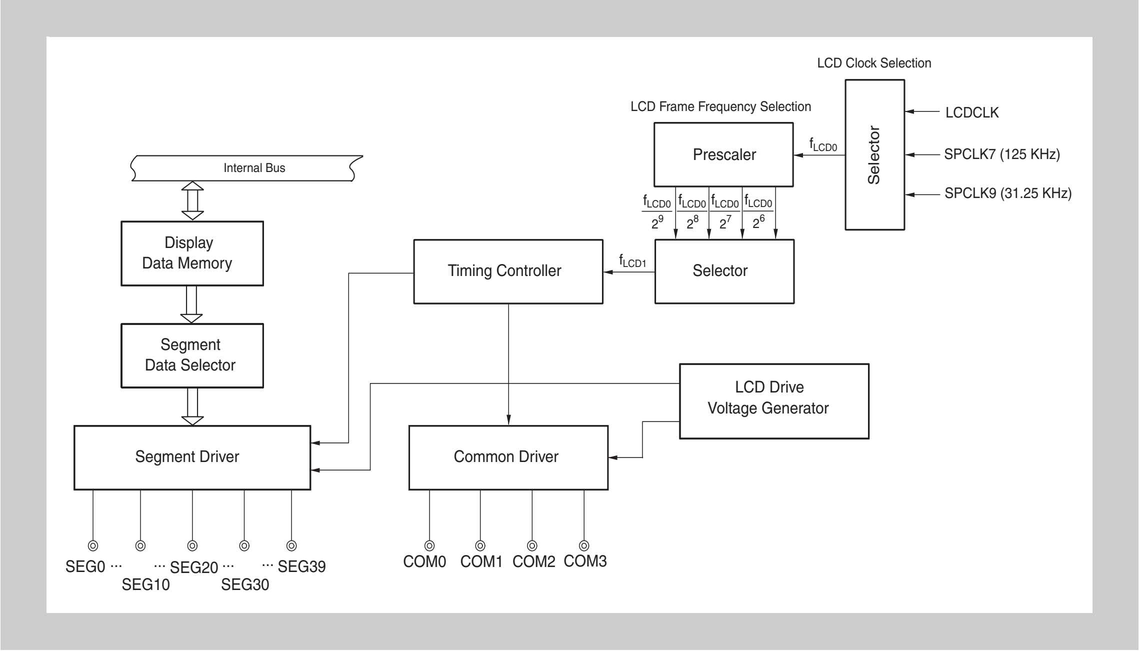


Figure 22-1 LCD Controller/Driver block diagram

The pattern that is to be displayed on the LCD panel has to be mapped to bit data. The bit data is stored in the display control registers SEGREGk (k = 0 to 39). The LCD Controller/Driver generates the corresponding output signals for driving the LCD panel.

The update rate of the LC display is determined by the frame frequency. It can be adjusted via the clock control register LCDC.

The external signals are listed in the following table.

Table 22-1 LCD Controller/Driver external connections

Signal name	I/O	Pins	Function
SEG[0:39]	O	SEG0 to SEG39	Segment signals
COM[0:3]	O	COM0 to COM3	Common signals

22.1.2 LCD panel addressing

Each individual segment of an LCD panel is addressed by a signal pair: a segment signal and a common signal. The segment becomes visible when the potential difference of the corresponding common signal and the segment signal reaches or exceeds the LCD drive voltage V_{LCD}.

Example *Figure 22-2* shows how the eight LCD segments of a digit are allocated to

- two segment signals (SEG_{2n} and SEG_{2n+1}, n = 0 to 19)
- four common signals

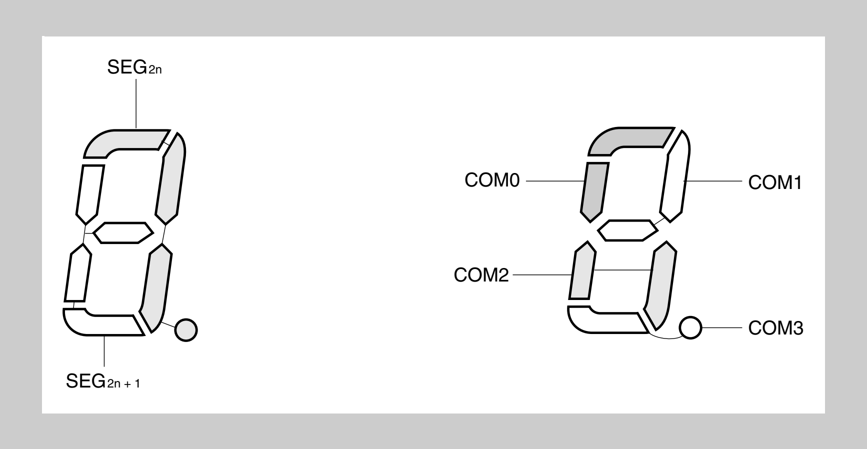


Figure 22-2 Allocation of segment signals and common signals to LCD segments (4-time-division)

Every combination of a segment and a common signal addresses a single element. The middle horizontal bar, for example, becomes visible if the potential difference of signals SEG_{2n+1} and COM1 exceeds V_{LCD}.

To display a desired pattern on the LCD panel:

1. Check what combination of segment and common signals form the desired display pattern.
2. Write bit data with the pattern to be displayed to registers SEGREGk.

The LCD Controller/Driver generates the corresponding segment and common signals.

See also the *“Display Example“ on page 818*.

Connections At the LCD panel, the signals are connected as follows:

Table 22-2 Signals and connections of LCD Controller/Driver

Signals	Connection at LCD panel
segment signals	front surface electrodes
common signals	rear surface electrodes

Caution The LCD panel is driven by AC voltage. The performance of the LCD deteriorates if DC voltage is applied in the common and segment signals. That means contrast and brightness of the display may decrease. The display may even be damaged.

22.2 LCD-C/D Registers

The LCD Controller/Driver is controlled by means of the following registers:

Table 22-3 LCD Controller/Driver registers overview

Register name	Shortcut	Address
LCD clock control register	LCDC0	FFFF $FB00_H$
LCD mode control register	LCDM0	FFFF $FB01_H$
LCD display control registers	SEGREG0k, k= 0 to 39	FFFF $FB20_H$ to FFFF $FB47_H$

(1) LCDC0 - LCD clock control register

The 8-bit LCDC0 register determines the duty cycle frequency f_{LCD1}.

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

Address FFFF $FB00_H$

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

7	6	5	4	3	2	1	0
0	0	0	0	LCDC03	LCDC02	LCDC01	LCDC00
R	R	R	R/W	R/W	R/W	R/W	R/W

Table 22-4 LCDC0 register contents

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>3 to 2</td><td>LCDC0[3:2]</td><td>Selects the LCD clock
<table>
<tr><th>LCDC03</th><th>LCDC02</th><th>Selected LCD clock (f_{LCD0})</th></tr>
<tr><td>0</td><td>0</td><td>LCDCLK</td></tr>
<tr><td>0</td><td>1</td><td>SPCLK7</td></tr>
<tr><td>1</td><td>0</td><td>SPCLK9</td></tr>
<tr><td>1</td><td>1</td><td>reserved</td></tr>
</table></td></tr>
<tr><td>1 to 0</td><td>LCDC0[1:0]</td><td>Selects the duty cycle frequency
<table>
<tr><th>LCDC01</th><th>LCDC00</th><th>Selected duty cycle frequency (f_{LCD1})</th></tr>
<tr><td>0</td><td>0</td><td>LCD clock divided by 2^6</td></tr>
<tr><td>0</td><td>1</td><td>LCD clock divided by 2^7</td></tr>
<tr><td>1</td><td>0</td><td>LCD clock divided by 2^8</td></tr>
<tr><td>1</td><td>1</td><td>LCD clock divided by 2^9</td></tr>
</table></td></tr>
</table>

Caution

1. Bit 4 must always be 0.
2. Changing the root clock source for LCDLCK will also change the Watch Timer clock WTCLK. For details refer to the *“Clock Generator“ on page 129*.

Note The frequency of LCDCLK is determined in the Clock Generator.
The root clock for LCDCLK can be selected from the main, sub, or ring oscillator. It can be identical with the clock source or it can be a fraction thereof.

Possible frame frequencies

Table 22-5 lists the possible frame frequencies. The values in *Table 22-5* are only examples. Check *“Clock Generator“ on page 129* for details.

Selection of the following LCD clocks is provided:

- LCDC0.LCDC0[3:2] = 00_B
  LCD clock = LCDCLK = f_0 / d, with
  - f_0 = root clock for LCDCLK
    It can be selected from f_{main} (4 MHz), f_{sub} (32.768 KHz), or f_{ring} (200 KHz).
  - d = divider
    LCDCLK is gained by dividing the root clock by d. Divider d can be selected from 2^0 to 2^7.
- LCDC0.LCDC0[3:2] = 01_B
  LCD clock = SPCLK7 = SPCLK0 / 2^7 = 125 KHz
- LCDC0.LCDC0[3:2] = 10_B
  LCD clock = SPCLK9 = SPCLK0 / 2^9 = 31.25 KHz

Table 22-5 **Example settings for frame frequency and duty cycle**

LCDC03	LCDC02	LCDC01	LCDC00	LCD clock[a]	Duty cycle frequency	Frame frequency
0	1	0	1	SPCLK7 = 125 KHz	977 Hz	244 Hz
0	1	1	0	SPCLK7 = 125 KHz	488 Hz	122 Hz
0	1	1	1	SPCLK7 = 125 KHz	244 Hz	61 Hz
1	0	0	0	SPCLK9 = 31.25 KHz	488 Hz	122 Hz
1	0	0	1	SPCLK9 = 31.25 KHz	244 Hz	61 Hz
0	0	0	0	LCDCLK = 32.768 KHz (with $f_0 = f_{sub}$ and d = 2^0)	512 Hz	128 Hz
0	0	0	1	LCDCLK = 32.768 KHz	256 Hz	64 Hz
0	0	0	1	LCDCLK = 100 KHz (with $f_0 = f_{ring}$ and d = 2^1)	781 Hz	195 Hz
0	0	1	0	LCDCLK = 100 KHz	391 Hz	98 Hz

a) The frequency of the LCD clock is determined in the clock generator.

(2) LCDM0 - LCD mode control register

The 8-bit LCDM0 register enables/disables the LCD operation, activates edge enhancement and selects the power supply.

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

Address FFFF FB01$_H$

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

7	6	5	4	3	2	1	0
LCDON0	0	0	LIPS0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 22-6 LCDM0 register contents

Bit position	Bit name	Function
7	LCDON0	Enables/disables LCD display 0: Display disabled No segment of the display is visible. The contents of the SEGREG0k registers are disregarded. The output is at non-selection level. 1: Display enabled
4	LIPS0	Selects the power supply 0: LCD Controller/Driver is not powered 1: LCD Controller/Driver is powered

Caution Bits 0, 1, 2, 3, 5, 6 must always be 0.

(3) SEGREG0k - LCD display control register (k = 0 to 39)

The 8-bit registers contain the data that is displayed on the LCD. Each register contains the data for one of the 40 segments.

Access These registers can be read/written in 8-bit or 1-bit units.

Address FFFF FB20$_H$ to FFFF FB47$_H$

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

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

Table 22-7 SEGREG0k register contents (k = 0 to 39)

Bit position	Bit name	Function
3 to 0	SEGREG0k[3:0]	Status of the LCD segment that is controlled by segment signal k and the common signal, that corresponds to the bit position. 0: Display off 1: Display on, if corresponding common signal is active

The bits 4 to 7 are ignored. They should be set to zero.

22.3 Operation

The following describes the timing of common and segment signals, the activation of an LCD segment and how edge enhancement can be applied.

22.3.1 Common signals and segment signals

This section describes the timing of common signals and segment signals and at which conditions an individual LCD segment becomes visible.

(1) Common Signals

Common signals COM0 to COM3 are generated internally. Together with the segment signals, they define which LCD segment is activated in the current cycle.

Figure 22-3 shows the common signal wave form for COM0, 1/4 duty (1/3 bias). 1/4 duty means each signal COMn is in selection level for one quarter of a frame.

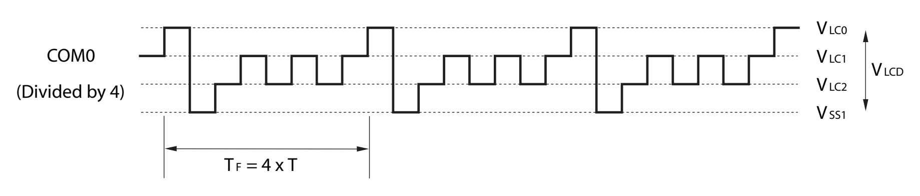


Figure 22-3 Common signal wave form (1/4 duty, 1/3 bias)

- T_F = frame cycle time.
  $T_F = 4 \times T$
  T corresponds to the duty cycle frequency f_{LCD1} and is thus determined by register LCDC.
- T = duty cycle time.
  Each frame cycle T_F is comprised of 4 duty cycles (1/4 duty), one duty cycle for each signal COMn.

Each LCD segment is allocated to one of the common signals. The LCD segment can only be activated in a duty cycle, in which the common signal is at selection level.

Figure 22-4 shows the selection and non-selection level of common signals.

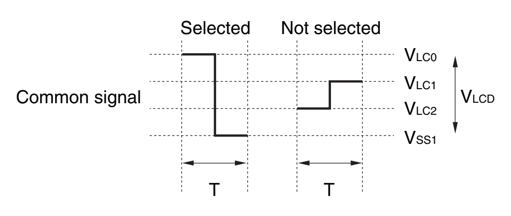


Figure 22-4 Selection level and non-selection level of common signals

T = duty cycle time.

(2) Segment Signals

Segment signals correspond to the contents of the 40 LCD display control registers SEGREG0k. Bits 0 to 3 of these registers are read in synchronization with the common signals COM0 to COM3, this means bit 0 is read in synchronization with common signal COM0 and so on.

- If the value of the bit is 1 while the common signal is at selection level, the corresponding segment signal is set to selection level.
- If the value of the bit is 0 while the common signal is at selection level, the corresponding segment signal is set to non-selection level.

Figure 22-5 shows the selection and non-selection level of segment signals.

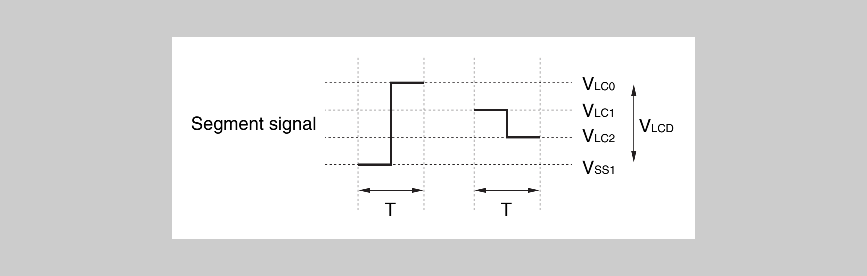
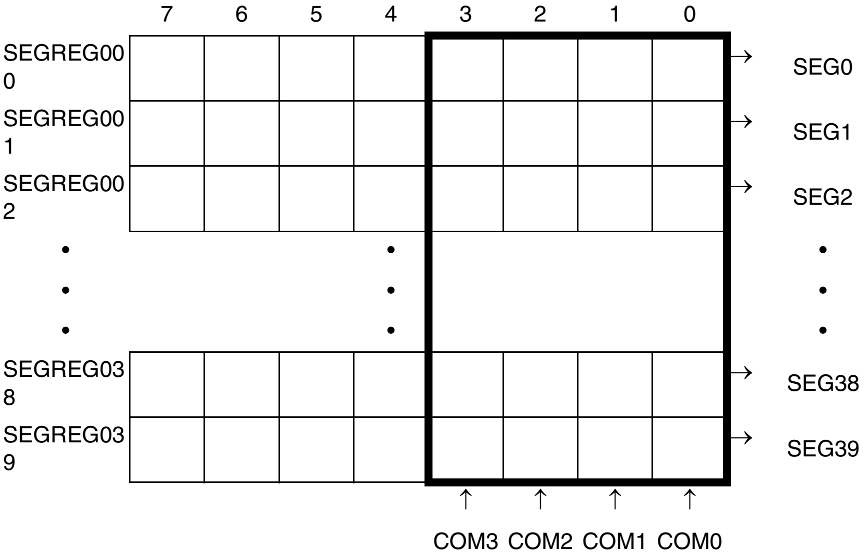


Figure 22-5 Selection level and non-selection level of segment signals

T = duty cycle time.

The table below shows the relation of the bits in registers SEGREG0k (k = 0 to 39) with common signals COM0 to COM3 and segment output signals SEG00 to SEG39.



Each of the bits 0 to 4 represents the status of one LCD segment. Setting the bit to 1 will make the LCD segment visible.

For example, setting bit SEGREG02[3] to 1 will make the LCD segment visible, that is controlled by the signal pair SEG2 and COM3.

22.3.2 Activation of LCD segments

An LCD segment becomes visible when the potential difference of the corresponding common signal and segment signal reaches or exceeds the LCD drive voltage V_{LCD}. This is achieved if common and segment signal are at their selection levels.

Within one frame cycle T_F, each LCD segment can be activated once. Activation lasts for one duty cycle T. LCD segments corresponding to common signals COM0 to COM3 are not activated simultaneously, but consecutively.

22.4 Display Example

As a display example, register contents and output signals for a 20-digit LCD display are presented in this section.

(1) LCD panel

The display pattern of a single digit is given below. Each digit is addressed by two segment signals and four common signals.

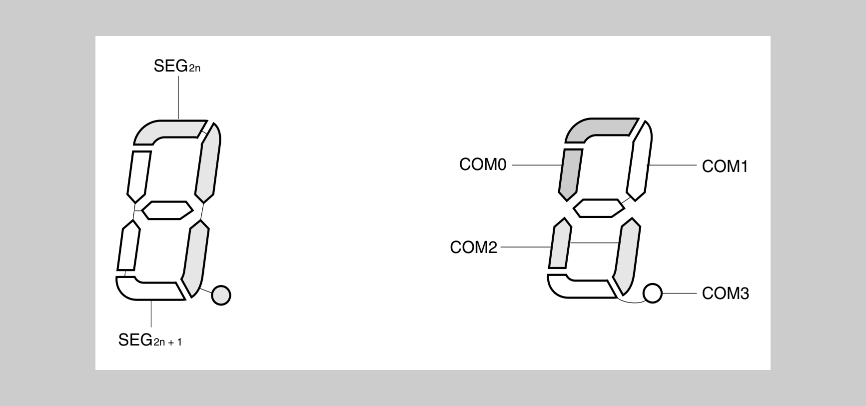


Figure 22-6 4-time-division LCD pattern and electrode connections

Figure 22-7 on page 820 shows the whole LCD panel and its connection to the segment signals and common signals. The display example is "123456.78901234567890," and the register contents of SEGREG0k (k = 0 to 39) correspond to this.

An explanation is given here taking the example of the 6th digit with point: "6.". The corresponding segment signals are output to pins SEG28 and SEG29 with the selection levels at the COM0 to COM3 common signal timings as shown in the table below:

Table 22-8 Selection and non-selection levels of example

Common signal	Segment signal SEG28	Segment signal SEG29
COM0	selected	selected
COM1	not selected	selected
COM2	selected	selected
COM3	selected	selected

From this, it can be seen that 1101_B must be prepared in the display control register SEGREG028 and 1111_B must be prepared in SEGREG029.

Examples of the LCD drive waveforms between SEG28 and the COM0 and COM1 signals are shown in *Figure 22-8 on page 821* (for the sake of simplicity, waveforms for COM2 and COM3 have been omitted).

When SEG28 is at the selection level at the COM0 selection timing, it can be seen that the $+V_{LCD}/–V_{LCD}$ AC square wave, which is the LCD illumination (ON) level, is generated.

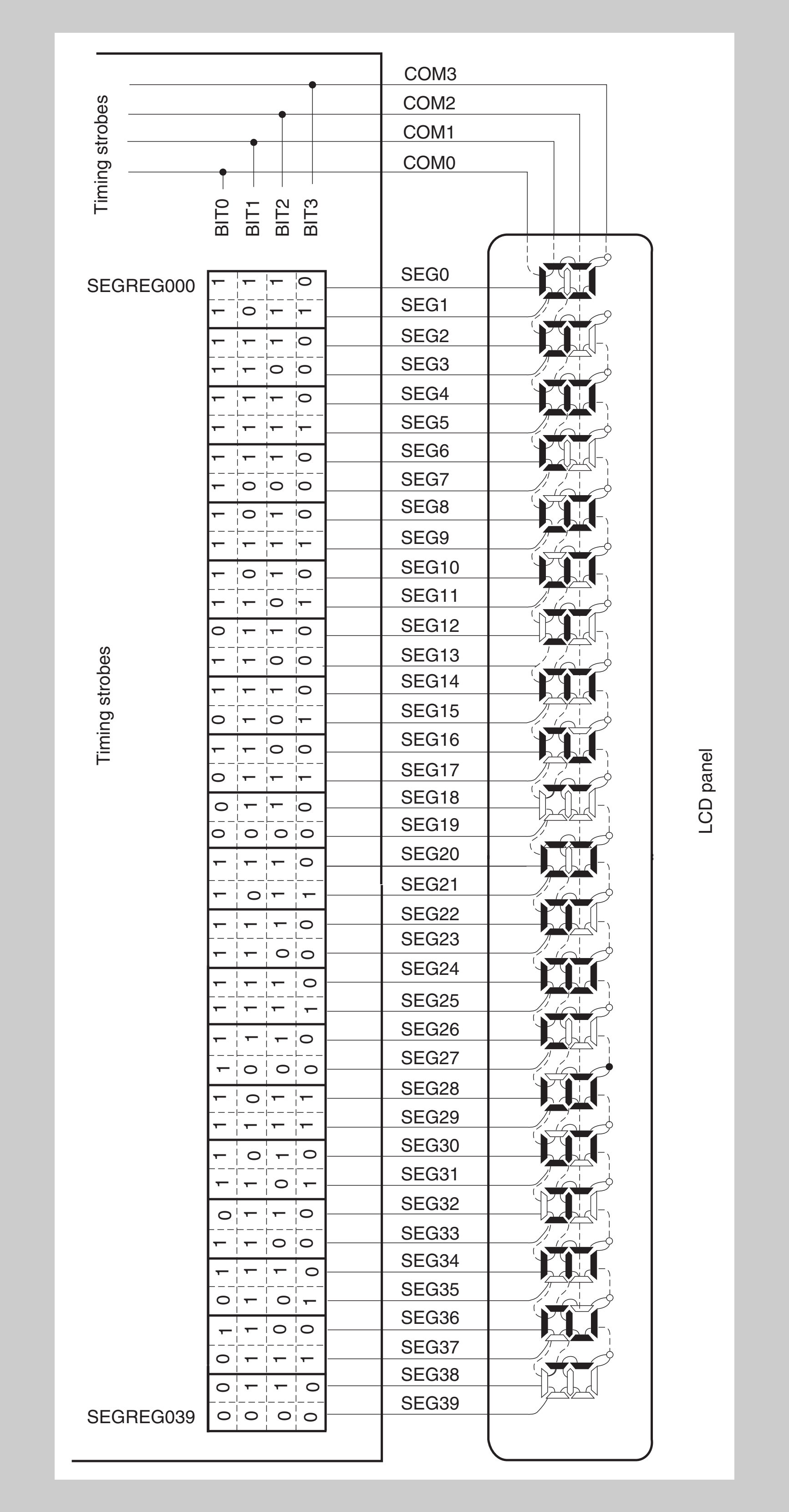


Figure 22-7 4-time-division LCD panel connection example

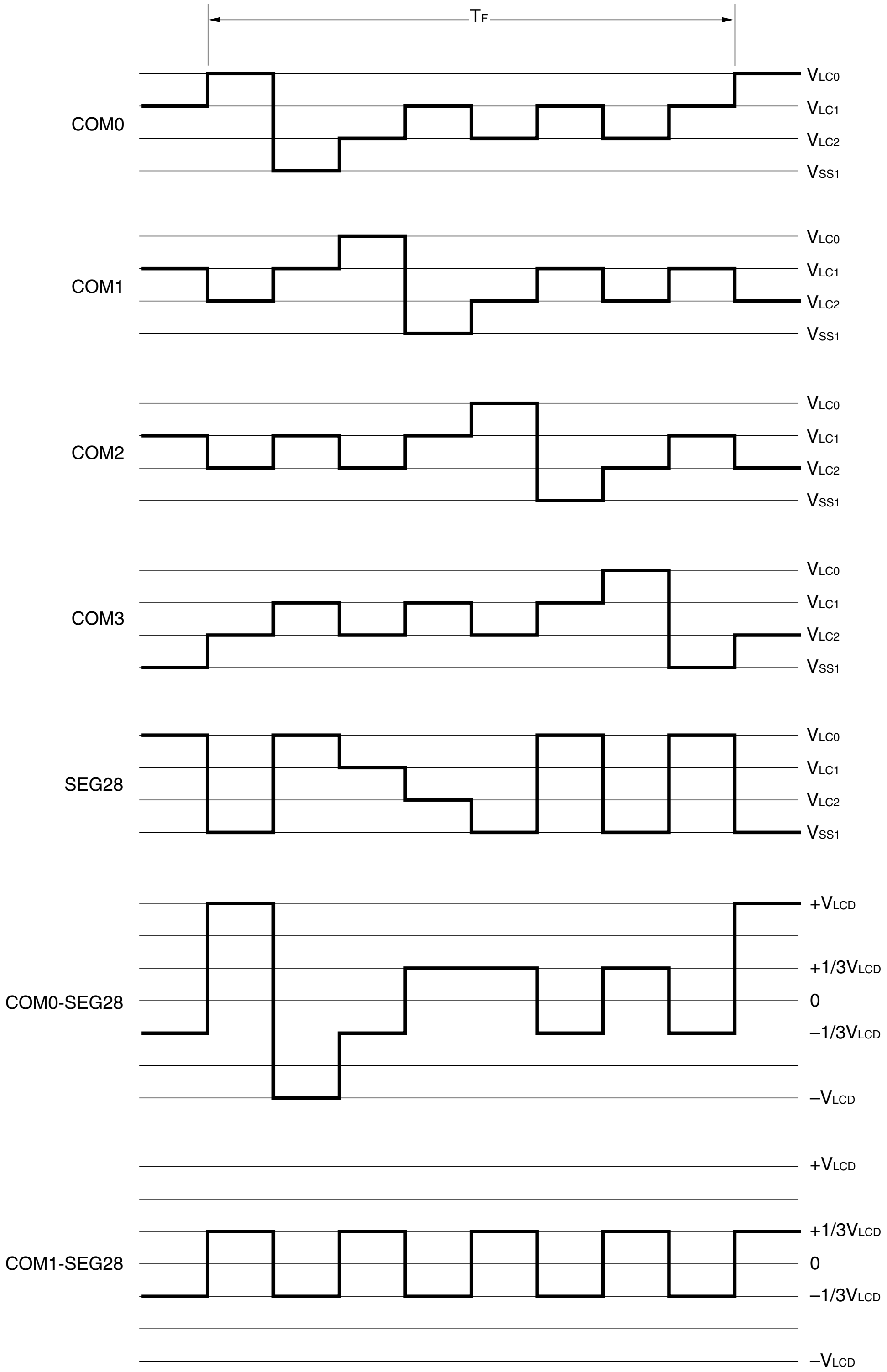


Figure 22-8 4-time-division LCD drive waveforms – examples

Chapter 23 LCD Bus Interface (LCD-I/F)

The LCD Bus Interface connects the internal peripheral bus to an external LCD controller. It provides an asynchronous 8-bit parallel data bus and two control lines.

The LCD Bus Interface supports bidirectional communication. You can send data to and query data from the LCD controller.

23.1 Overview

The LCD Bus Interface transmits or receives data bytes. Two control lines specify the read/write timing and the transfer direction.

The LCD Bus Interface suits LCD controllers that feature automatic generation of display memory addresses. The interface does not generate or interpret specific signals that may be required by the LCD controller, like address, chip select, hold, and so on. If necessary, such signals can be provided by general purpose I/Os.

Features summary The LCD Bus Interface provides:

- Support of two different control signals modes:
  - mod80 with separate read and write strobe
  - mod68 with read/write signal and data strobe “E” with selectable level
- DMA for read and write operations
- 8/16/32-bit write and read operations
- Programmable transfer speed (100 KHz … 3.2 MHz) through
  - selectable clock input
  - programmable transfer time
  - programmable wait states
- Interrupt generation selectable upon two events
  - internal data transfer allowed
  - external bus access completed
- Flags that indicate the status of the data register and the progress of data transfer to or from the LCD controller

Note
1. The programmer has to make sure that the timing requirements of the external LCD controller are met.
2. For electrical characteristics please refer to the Electrical Target Specification.
3. If the concerned pins are configured as LCD Bus Interface pins change between input and output is performed automatically by LCD Bus Interface read and write operations. Refer also to *"Port group 9" on page 85*.

23.1.1 Description

Data can be read from and written to the LCD Bus Interface by either involving the DMA Controller or by directly accessing the interface from the CPU. The timing of the external bus signals is determined by register settings (WST and CYC).

The LCD Bus Interface is 8 bits wide. In order to improve performance, the interface is equipped with a 32-bit register that allows the CPU or DMA to access the data register with 8-, 16-, or 32-bit data accesses. The interface automatically generates 1, 2, or 4 consecutive (8-bit) accesses on the external bus.

The LCD Bus Interface has an internal 32-bit write buffer that allows the next data to be written to the data register (LBDATA0) while a transfer on the external bus interface is in progress.

The following figure shows the main components of the LCD Bus Interface.

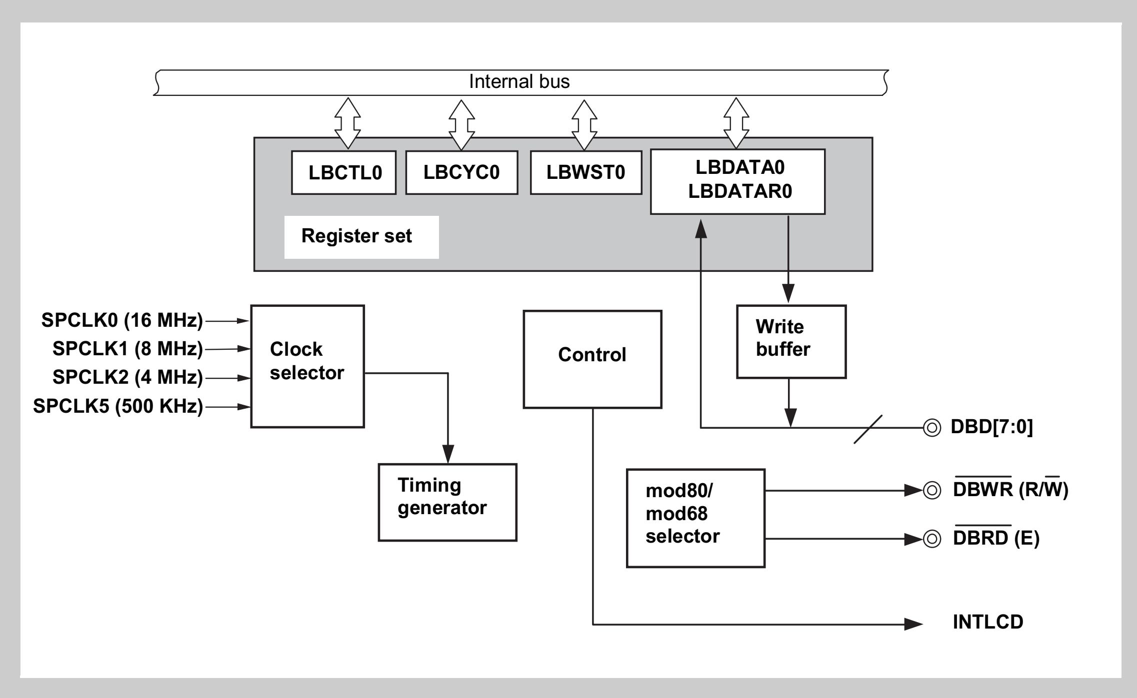


Figure 23-1 LCD Bus Interface block diagram

As shown in the figure, the result of a read operation is directly available in the LBDATA0 and LBDATAR0 registers. For data output, the contents of the LBDATA0 register is copied to the 32-bit write buffer.

The external signals are listed in the following table.

Table 23-1 LCD Bus Interface external connections

Signal name	I/O	Active level	Reset level	Function
$\overline{\text{DBWR}}$	O	L	H	mod80: Write strobe ($\overline{\text{WR}}$) mod68: Read/Write (R/$\overline{\text{W}}$)
$\overline{\text{DBRD}}$	O	L[a]	H	mod80: Read strobe ($\overline{\text{RD}}$) mod68: E strobe (E)
DBD[7:0]	I/O		L	LCD data bus

a) The active level of E in mod68 is controlled by the bit LBCTL0.EL0.

23.1.2 LCD Bus Interface access modes

The LCD Bus Interface can access the external LCD Controller/Driver in two different modes. The mode is selected by the bit LBCTL0.IMD.

- mod80
  The control signals $\overline{\text{WR}}$ ($\overline{\text{DBWR}}$) and $\overline{\text{RD}}$ ($\overline{\text{DBRD}}$) are used to control the external LCD Controller/Driver.
- mod68
  The control signals R/$\overline{\text{W}}$ ($\overline{\text{DBWR}}$) and E ($\overline{\text{DBRD}}$) are used to control the external component.
  The level of E depends on the setting of the bit LBCTL0.EL0:
  – EL0=0: E is active high; data is read/written on the falling edge.
  – EL0=1: E is active low; data is read/written on the rising edge.

23.1.3 Access types to the LBDATA0 register

Access to the LBDATA0 register can be performed as:

- Byte access (8-bit)
- Halfword access (16-bit)
- Word access (32-bit)

Note

1. Every access must address the base address of the LBDATA0 register. Access to the individual bytes within the register is prohibited.
2. Before writing to or reading from the LBDATA0 register or reading the LBDATAR0 register, always make sure that the busy flag LBCTL0.BYF is zero.

(1) Write operation

If there is no transfer in progress on the external bus interface, the new data is immediately transferred to the external LCD controller. If there is a transfer in progress, the new data is transferred after the current transfer has completed.

One, two or four bytes are transferred through the bus interface, depending on how LBDATA0 was accessed (byte, halfword, or word).

The write timing on the external bus interface is determined by the number of wait cycles (LBWST0.WST[4:0]), the cycle time (LBCYC0.CYC[5:0]) and the selected clock (LBCTL0.LBC00 and LBCTL0.LBC01).

(2) Read operation

When the CPU or the DMA reads the LBDATA0 register, the read operation on the LCD Bus Interface is started. If there is a write transfer in progress while the LBDATA0 register shall be read, the read transfer is stalled and started after the write transfer has completed.

The value read from the register is the data from the *previous* transfer. Therefore, an initial dummy read operation is required to update the register.

As soon as the data of the actual transfer is available in the LBDATA0 register, the busy flag LBCTL0.BYF0 is cleared and the data can be retrieved with the next read operation. Successive reads from the LBDATA0 register provide the desired data.

The read timing on the external bus interface is determined by the number of wait cycles (LBWST0.WST0[4:0]), the cycle time (LBCYC0.CYC0[5:0]) and the selected clock (LBCTL0.LBC0 and LBCTL00.LBC01).

(3) Read operation without initiating a bus transfer

Data can be read from the LBDATAR0 register without initiating a new read transfer via the LCD Bus Interface.

The read access to the LBDATAR0 register is useful when previous read accesses to the LBDATA0 register have been performed and only the last transferred data shall be read without starting a new LCD bus transfer.

23.1.4 Interrupt generation

An interrupt is generated on write and read accesses to the LCD Bus Interface. Depending on the setting of the bit LBCTL0.TCIS0, the interrupt is generated differently.

Table 23-2 Controlling interrupt generation of the LCD Bus Interface

Access	TCIS0 = 0	TCIS0 = 1
Write	An interrupt is generated as soon as data is transferred from LBDATA0 to the write buffer. Then LBDATA0 is ready to accept next data. The write buffer is filled whenever the external bus interface is idle (no transfer in progress) and data is available in LBDATA0.	An interrupt is generated as soon as the write transfer via the bus interface has completed. The transfer can consist of 1, 2, or 4 bytes dependent on the access to LBDATA0.
Read	An interrupt is generated as soon as the data is available in the LBDATA0 or LBDATAR0 register. Depending on the read access to LBDATA0 (byte, halfword or word) 1, 2, or 4 bytes are transferred and placed in the LBDATA0 and LBDATAR0 register. Finally, the interrupt is generated in order to indicate that new data is available.	An interrupt is generated as soon as the read transfer via the bus interface has completed. The transfer can consist of 1, 2, or 4 bytes depending on the access to LBDATA0 or LBDATAR0.

23.2 LCD Bus Interface Registers

The LCD Bus Interface is controlled and operated by means of the following registers:

Table 23-3 LCD Bus Interface registers overview

Register name	Shortcut	Address
LCD Bus Interface control register	LBCTL0	FFFF $FB60_H$
LCD Bus Interface cycle time register	LBCYC0	FFFF $FB61_H$
LCD Bus Interface wait states register	LBWST0	FFFF $FB62_H$
LCD Bus Interface data register	LBDATA0W	FFFF $FB70_H$
	LBDATA0	
	LBDATA0L	
LCD Bus Interface data (read) register	LBDATAR0W	FFFF $FB74_H$
	LBDATAR0	
	LBDATAR0L	

(1) LBCTL0 - LCD Bus Interface control register

The 8-bit LBCTL0 register controls the operation of the LCD Bus Interface.

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

Address FFFF $FB60_H$

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

7	6	5	4	3	2	1	0
EL0	IMD0	LBC01	LBC00	TCIS0	0	TPF0	BYF0
R/W	R/W	R/W	R/W	R/W	R	R	R

Table 23-4 LBCTL0 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>EL0</td><td>Level of signal “E” in mod68 mode
0: E is active high; data is read/written on the falling edge.
1: E is active low, data is read/written on the rising edge.</td></tr>
<tr><td>6</td><td>IMD0</td><td>Mode of external bus interface access
0: mod80 mode - control signals are $\overline{WR}$ and $\overline{RD}$
1: mod68 mode - control signals are E and R/$\overline{W}$</td></tr>
<tr><td>5 to 4</td><td>LBC0[1:0]</td><td>Selects the internal clock
<table>
<tr><th>LBC01</th><th>LBC00</th><th>Selected clock</th></tr>
<tr><td>0</td><td>0</td><td>SPCLK0</td></tr>
<tr><td>0</td><td>1</td><td>SPCLK1</td></tr>
<tr><td>1</td><td>0</td><td>SPCLK2</td></tr>
<tr><td>1</td><td>1</td><td>SPCLK5</td></tr>
</table></td></tr>
<tr><td>3</td><td>TCIS0</td><td>Select interrupt generation
0: During write access to the bus interface, an interrupt is generated as soon as data is transferred from LBDATA0 to the write buffer.
During read access from the bus interface, an interrupt is generated as soon as data is available in the LBDATA0 and LBDATAR0 registers.
1: An interrupt is generated as soon as the read or write transfer via the bus interface has completed.</td></tr>
<tr><td>1</td><td>TPF0</td><td>Transfer in progress on external bus interface
0: The external bus interface is idle
1: Data is transferred on the external bus interface</td></tr>
<tr><td>0</td><td>BYF0</td><td>Data register busy
0: Data can be read or written from/to LBDATA0
Data can be read from LBDATAR0
1: Register LBDATA0 (LBDATAR0) is busy</td></tr>
</table>

(2) LBCYC0 - LCD Bus Interface cycle time register

The 8-bit LBCYC0 register determines the cycle time of the LCD Bus Interface. The cycle time is the duration of one bus access for transferring one byte.

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

Address FFFF FB61$_H$

Initial Value 02_H. This register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	CYC05	CYC04	CYC03	CYC02	CYC01	CYC00
R	R	R/W	R/W	R/W	R/W	R/W	R/W

Table 23-5 LBCYC0 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>5 to 0</td><td>CYC0[5:0]</td><td>Sets the cycle time of the LCD Bus Interface.
<table>
<tr><th>CYC0[5:0]</th><th>Cycle time</th></tr>
<tr><td>000000_B</td><td>Setting prohibited</td></tr>
<tr><td>000001_B</td><td>Setting prohibited</td></tr>
<tr><td>000010_B</td><td>Cycle time is 2*T</td></tr>
<tr><td>000011_B</td><td>Cycle time is 3*T</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>111111_B</td><td>Cycle time is 63*T</td></tr>
</table>
</td></tr>
</table>

Note
1. T is the clock period of the selected SPCLK.
2. Always keep LBCYC0 $\geq$ 2.

(3) LBWST0 - LCD Bus Interface wait state register

The 8-bit LBWST0 register determines the number of wait states of the LCD Bus Interface. The number of wait states defines the duration of the $\overline{\text{DBWR}}$ and $\overline{\text{DBRD}}$ signals. This duration must remain below the cycle time.

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

Address FFFF FB62_H

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

7	6	5	4	3	2	1	0
0	0	0	WST04	WST03	WST02	WST01	WST00
R	R	R	R/W	R/W	R/W	R/W	R/W

Table 23-6 LBWST0 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>4 to 0</td><td>WST0[4:0]</td><td>Sets the number of wait states of the LCD Bus Interface.
<table>
<tr><th>WST0[4:0]</th><th>Wait states</th></tr>
<tr><td>00000_b</td><td>No wait state inserted</td></tr>
<tr><td>00001_b</td><td>1 wait state</td></tr>
<tr><td>00010_b</td><td>2 wait states</td></tr>
<tr><td>00011_b</td><td>3 wait states</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>11111_b</td><td>31 wait states</td></tr>
</table>
</td></tr>
</table>

Note Always keep LBWST0.WST0 $\leq$ LBCYC0.CYC0 – 2.

(4) LBDATA0 - LCD Bus Interface data register

The 32-bit LBDATA0 register contains the data that is transferred via the LCD Bus Interface.

Access This register can be read/written in 3 different units under following names:

- LBDATA0W: 32-bit access
- LBDATA0: 16-bit access
- LBDATA0L: 8-bit access

Address FFFF $FB70_H$

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

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

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

Table 23-7 LBDATA0W register contents

Bit Position	Bit Name	Function
31 to 0	DATA[31:0]	Data that is written to or read from the LCD Bus Interface

Table 23-8 LBDATA0 register contents

Bit Position	Bit Name	Function
15 to 0	DATA[15:0]	Data that is written to or read from the LCD Bus Interface

Table 23-9 LBDATA0L register contents

Bit Position	Bit Name	Function
7 to 0	DATA[7:0]	Data that is written to or read from the LCD Bus Interface

Access types Depending on the access to this register (byte, halfword or word), a defined number of transfers via the external bus interface are performed:

- Byte:
  The byte is transferred via the bus interface.
- Halfword:
  The halfword is split into 2 bytes that are transferred consecutively via the bus interface.
- Word:
  The word is split into 4 bytes that are transferred consecutively via the bus interface.

When the data is split into bytes and transferred consecutively, the byte order is as follows:

31 24	23 16	15 8	7 0
4th Byte	3rd Byte	2nd Byte	1st Byte

Write to this register A write operation to this register sets the busy flag LBCTL0.BYF0 immediately.

If there is no LCD bus transfer in progress (LBCTL0.TPF0 = 0), the data is copied to the write buffer and LBCTL0.BYF0 is cleared.

If there is a transfer going on (LBCTL0.TPF0 = 1), the data is not copied to the write buffer until the transfer has completed. As soon as the transfer is complete, the data is copied to the write buffer and LBCTL0.BYF0 is cleared.

A transfer via the LCD Bus Interface starts as soon as the LBDATA0 register is copied to the write buffer. This is indicated by the interrupt INTLCD that becomes active, provided that LBCTL0.TCIS0 = 0.

Read from this register A read operation from this register initiates a read transfer via the LCD Bus Interface. The data that is read from the register is always the data that was received during the previous transfer from the LCD Bus Interface.

Note

1. Every access must address the base address of the LBDATA0 register. Access to the individual bytes within the register is prohibited.
2. LBCTL0.BYF0 must be zero when accessing this register.

(5) LBDATAR0 - LCD Bus Interface data register

The LBDATAR0 register is read-only. It contains the data of the last previous read transfer via the LCD Bus Interface. Reading this register does not start a new read transfer on the LCD Bus Interface.

Access This register can be read/written in 3 different units under following names:

- LBDATAR0W: 32-bit access
- LBDATAR0: 16-bit access
- LBDATAR0L: 8-bit access

Address FFFF $FB74_H$

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

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

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

Table 23-10 LBDATAR0W register contents

Bit position	Bit name	Function
31 to 0	DATA[31:0]	Data that was previously read from the LCD Bus Interface

Table 23-11 LBDATAR0 register contents

Bit Position	Bit Name	Function
15 to 0	DATA[15:0]	Data that was previously read from the LCD Bus Interface

Table 23-12 LBDATAR0L register contents

Bit Position	Bit Name	Function
7 to 0	DATA[7:0]	Data that was previously read from the LCD Bus Interface

This register can be read to obtain data that was transferred during a previous read operation to the LBDATA0 register—without initiating a further LCD bus transfer.

Reading the LBDATAR0 register does not change the status of the LBCTL0.BYF0 and LBCTL0.TPF0 flags.

23.3 Timing

This section starts with the general timing and then presents examples of consecutive write and read operations.

23.3.1 Timing dependencies

The following figure shows the general timing when the mod80 mode is used.

It illustrates the effect of the LBCYC0 and LBWST0 register settings. It explains also the impact of LBCTL0.TCIS on the interrupt generation.

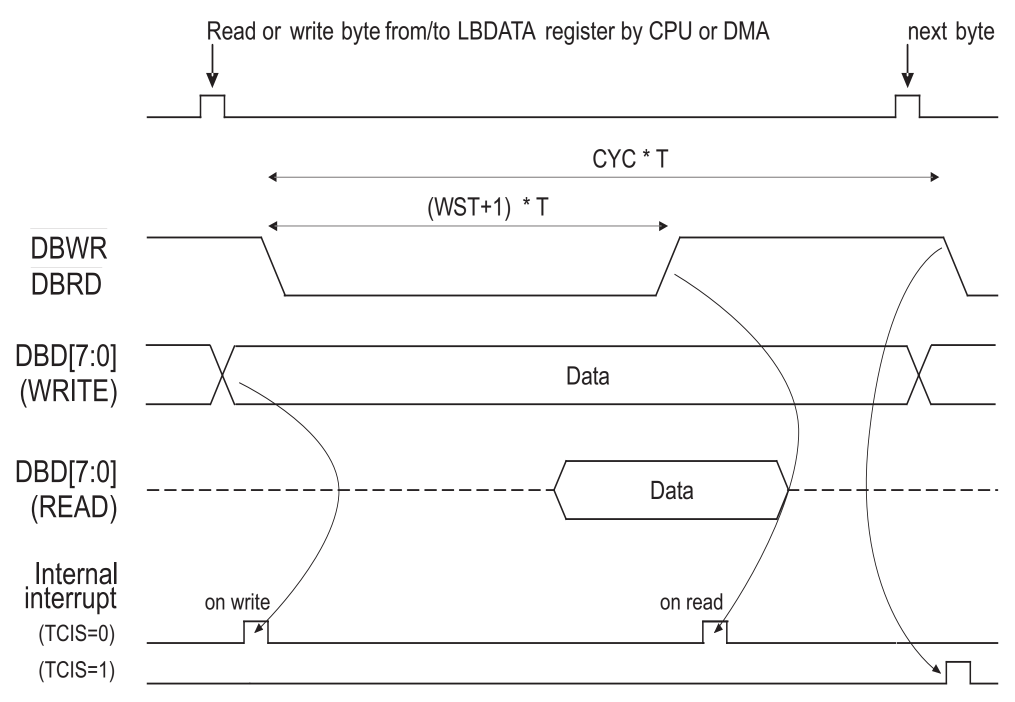


Figure 23-2 **LCD Bus Interface timing (mod80 mode)**

In mod80 mode, $\overline{\text{DBWR}}$ provides the write strobe $\overline{\text{WR}}$ and $\overline{\text{DBRD}}$ the read strobe $\overline{\text{RD}}$.

Note
1. T is the clock period of the selected SPCLK.
2. CYC is the chosen number of clock cycles (LBCYC0.CYC0). Always keep LBCYC0.CYC0 $\geq$ 2.
3. WST is the chosen number of wait states (LBWST0). Always keep LBWST0.SWST0 $\leq$ (LBCYC0.CYC0 – 2).

The only difference in mod68 mode is, that $\overline{\text{DBWR}}$ provides the read/write R/$\overline{\text{W}}$ strobe and $\overline{\text{DBRD}}$ the E strobe. The active edge of the E strobe is defined by LBCTL0.EL0.

23.3.2 LCD Bus I/F states during and after accesses

Changing between input and output mode of the LCD bus pins DB[7:0] is done automatically after they are configured as LCD Bus Interface pins via the port configuration registers.

After the pins are configured as DB[7:0] they are operating in input mode.

During and after a bus read access DB[7:0] are operating in input mode and retain this mode also after the read access is completed.

During and after a bus write access DB[7:0] are operating in output mode and retain this mode also after the write access is completed.

23.3.3 Writing to the LCD bus

This section shows typical sequences of writing words, halfwords and bytes to the LCD bus.

(1) Writing words

Writing a word transmits four bytes to the external LCD Controller/Driver.

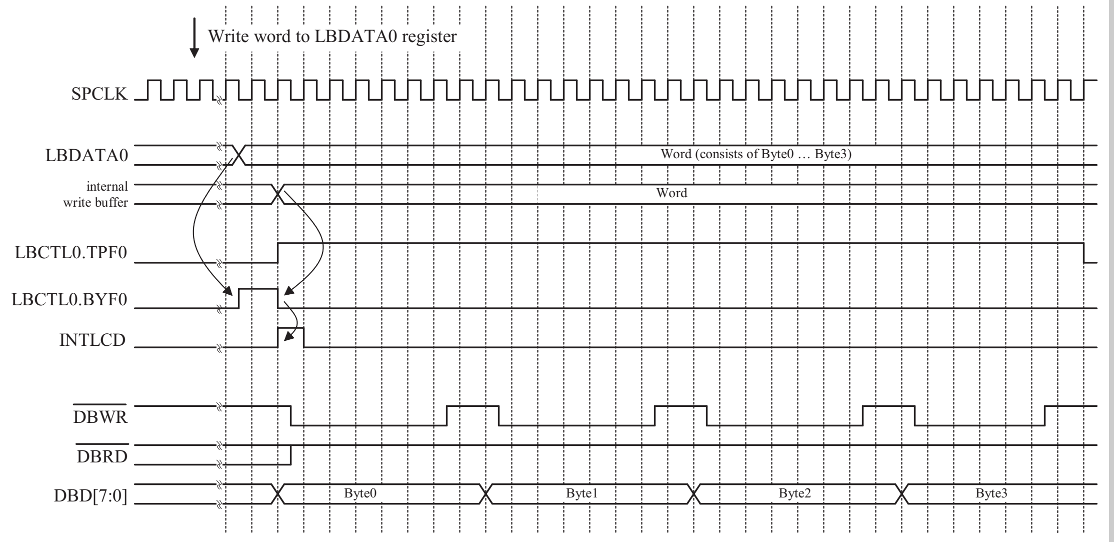


Figure 23-3 Timing (mod80: LBTCTL0.IMD0 = 0): write word, LBWST0.WST0 = 5, LBCYC0.CYC0 = 8, LBTCTL0.TCIS0 = 0

Note The timing diagrams are for functional explanation purposes only without any relevance to the real hardware implementation.

Sequence

1. A word of LCD data is written to the LBDATA0 register. The internal bus transfer takes some clocks until the register is updated. Then the busy flag LBCTL0.BYF0 is set until the data is copied to the write buffer.
2. The LBDATA0 register contents is copied to the write buffer. This clears LBCTL0.BYF0 and causes the interrupt output to become active for one clock cycle. Transfer on the external bus interface starts with byte 0. The “transfer in progress” flag LBCTL0.TPF0 is set to indicate that a transfer is in progress.

3. All four bytes of the word are transferred back-to-back via the external bus interface.
4. After the transfer on the external bus interface has been completed, the LBCTL0.TPF0 is cleared.

(2) Writing halfwords

Writing a halfword transmits two bytes to the external LCD Controller/Driver.

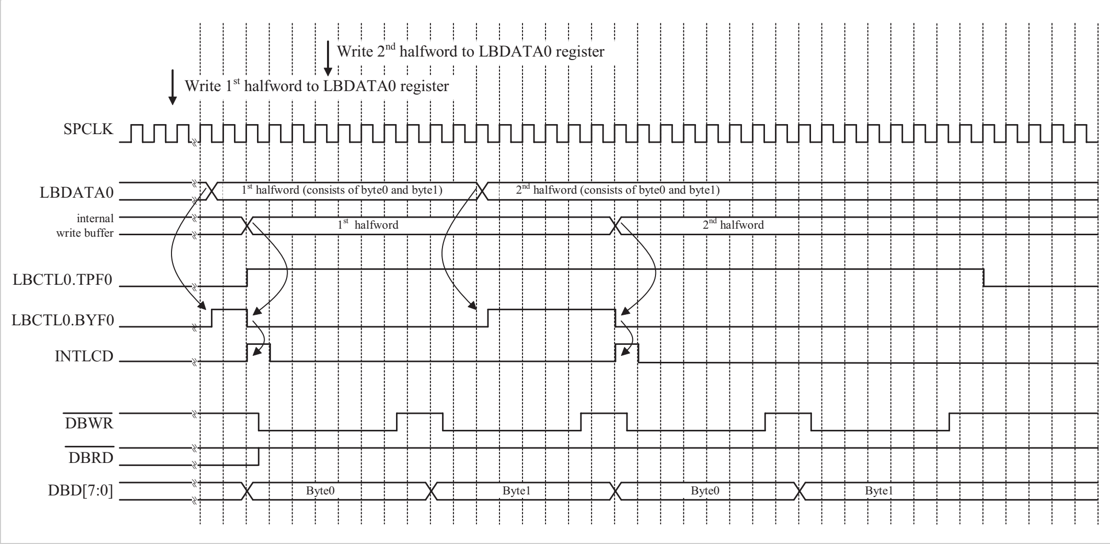


Figure 23-4 **Timing (mod80: LBTCTL0.IMD0 = 0): write consecutive halfwords, LBWST0.WST0 = 5, LBCYC0.CYC0 = 8, LBTCTL0.TCIS0 = 0**

Note The timing diagrams are for functional explanation purposes only without any relevance to the real hardware implementation.

Sequence

1. The first halfword of LCD data is written to the LBDATA0 register. The internal bus transfer takes some clocks until the interface register is written. Then the busy flag LBCTL0.BYF0 is set until the data is copied to the write buffer.
2. The LBDATA0 register contents is copied to the write buffer. This clears LBCTL0.BYF0 and causes the interrupt output to become active for one clock cycle. Transfer on the external bus interface starts with byte 0. The flag LBCTL0.TPF0 is set to indicate that a transfer is in progress.
3. Caused by the interrupt, the DMA writes a second halfword to LBDATA0. The CPU can write this halfword as well after it has checked the busy flag LBCTL0.BYF0. The internal bus transfer again takes some clock cycles until the LBDATA0 register is written and LBCTL0.BYF0 is set.
4. Because the transfer (two bytes) on the external bus interface is still going on and the LBDATA0 register contents can not be copied to the write buffer immediately, LBCTL0.BYF0 is set.
5. After the transfer over the external bus interface has been completed, the write buffer is filled with the contents of LBDATA0. The busy flag LBCTL0.BYF0 is cleared, and the interrupt output INTLCD becomes active for one clock cycle.

Filling the write buffer starts a new transfer to the external LCD controller.

(3) Writing bytes

Writing consecutive bytes transmits these bytes to the external LCD controller/ driver.

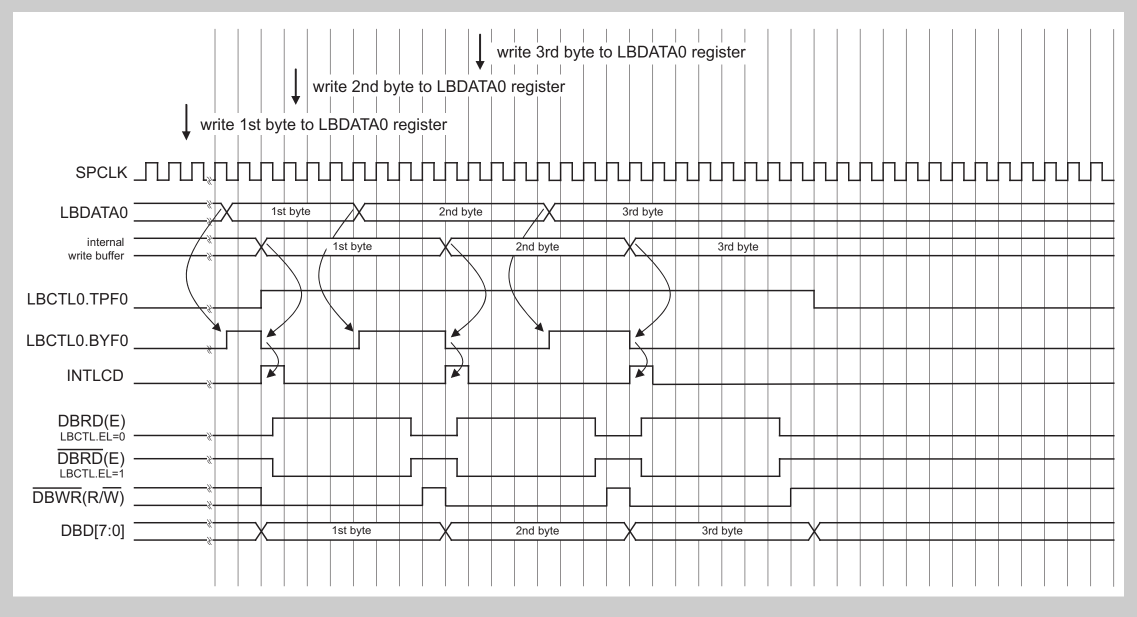


Figure 23-5 Timing (mod68 mode: LBTCTL0.IMD0 = 1): write consecutive bytes, LBWST0.WST0 = 5, LBCYC0.CYC0 = 8, , LBCTL0.TCIS0 = 0

Note The timing diagrams are for functional explanation purposes only without any relevance to the real hardware implementation.

Sequence

1. The first byte of LCD data is written to the LBDATA0 register. The internal bus transfer takes some clocks until the register of the interface is written. Then the busy flag LBCTL0.BYF0 is set until the data is copied to the write buffer.
2. The LBDATA0 register contents is copied to the write buffer. This clears LBCTL0.BYF0 and causes the interrupt output to become active for one clock cycle. Transfer on the external bus interface is started. The flag LBCTL0.TPF0 is set to indicate that a transfer is in progress.
3. Caused by the interrupt, the DMA writes a second byte to LBDATA0. The CPU can write this byte as well after it has checked the busy flag LBCTL0.BYF0. The internal bus transfer again takes some clock cycles until the LBDATA0 register is written and LBCTL0.BYF0 is set.
4. Since the transfer (one byte) on the external bus interface is still going on and the LBDATA0 register contents can not be copied to the write buffer immediately, the busy flag LBCTL0.BYF0 remains set.
5. After the transfer on the external bus interface has been completed, the write buffer is filled with the contents of LBDATA0. The busy flag LBCTL0.BYF0 is cleared and the interrupt output INTLCD becomes active for one clock cycle.

Filling the write buffer starts a new transfer to the external LCD controller.

23.3.4 Reading from the LCD bus

You can read from the LCD bus in word, halfword, or byte format. The following shows typical sequences of reading words and bytes.

(1) Reading words

Reading a word requires the transmission of four bytes.

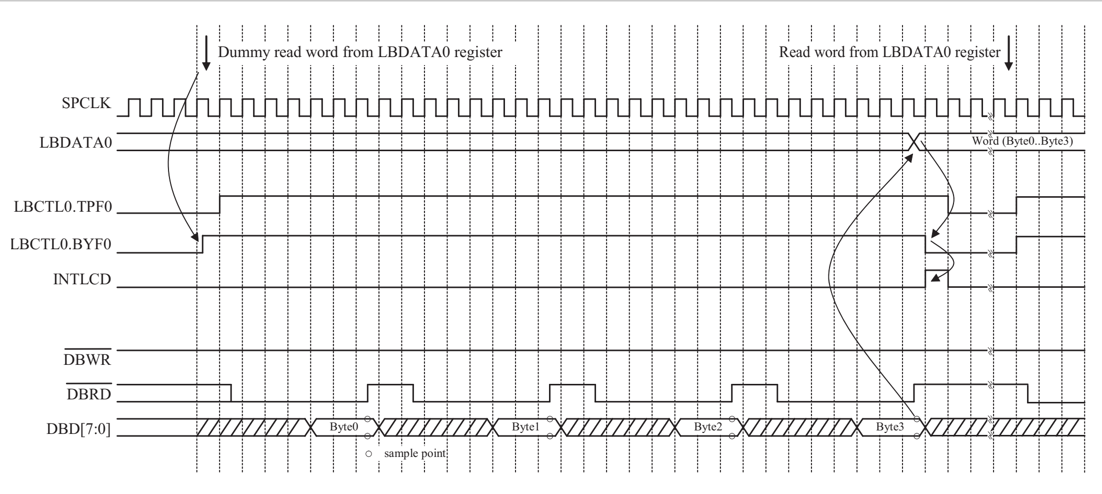


Figure 23-6 Timing (mod80: LBTCTL0.IMD0 = 0): read word, LBWST0.WST0 = 5, LBCYC0.CYC0 = 8, LBTCTL0.TCIS0 = 0

Note The timing diagrams are for functional explanation purposes only without any relevance to the real hardware implementation.

Sequence

1. A dummy read to the LBDATA0 register starts the transfer of four bytes from the external LCD controller. The busy flag LBCTL0.BYF0 is set immediately. The “transfer in progress” flag LBCTL0.TPF0 is set on the rising edge of the clock.
   The data that is read from LBDATA0 belongs to a previous transfer and may be ignored.
2. When the last of the four bytes is sampled and the complete word is available in the LBDATA0 register, the busy flag LBCTL0.BYF0 is cleared. The LBCTL0.TPF0 flag remains set until the cycle time of the last byte has elapsed.
3. A following read to the LBDATA0 register provides the LCD controller data and initiates a new transfer.

(2) Reading bytes

The following figure shows a byte read operation in mod68 mode.

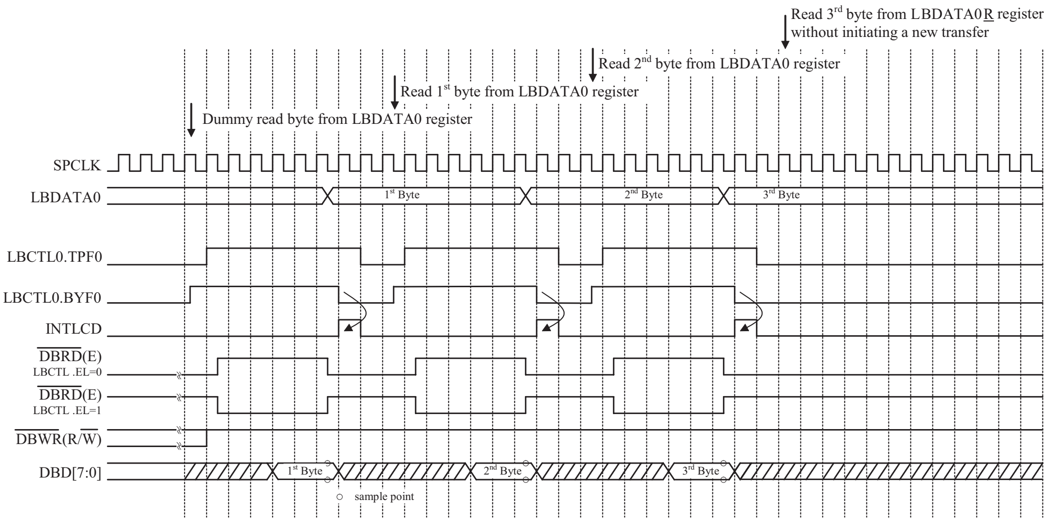


Figure 23-7 Timing (mod68: LBTCTL0.IMD0 = 1): read consecutive bytes, LBWST0.WST0 = 4, LBCYC0.CYC0 = 7, LBTCTL0.TCIS0 = 0

Note The timing diagrams are for functional explanation purposes only without any relevance to the real hardware implementation.

Sequence

1. A dummy read to the LBDATA0 register starts the transfer of one byte from the external LCD controller. The busy flag LBCTL0.BYF0 is set immediately. The “transfer in progress” flag LBCTL0.TPF0 is set on the rising edge of the clock.
   The data that is read from LBDATA0 belongs to a previous transfer and may be ignored.
2. When the data on the LCD Bus Interface is sampled, LBCTL0.BYF0 is cleared and the data is available in LBDATA0. The interrupt output INTLCD becomes active for one clock cycle.
3. A new read to LBDATA0 is performed while the previous transfer has not been finished (cycle time not elapsed). The busy flag LBCTL0.BYF0 is set immediately, but the new transfer is started after the previous one is complete. The “transfer in progress flag” LBCTL0.TPF0 remains set.
   The data that is read from LBDATA0 is the first LCD data byte.
4. Again, the data that has been sampled is available in LBDATA0 and the busy flag LBCTL0.BYF0 is cleared.
5. Steps 2 to 4 are repeated until the last byte to be read has been sampled.
6. The last byte is not read from the LBDATA0 register but from LBDATAR0 in order to avoid a further read transfer on the LCD bus.

23.3.5 Write-Read-Write sequence on the LCD bus

Figure 23-8 shows an example when a write access to the LCD bus is immediately followed by a read access and vice versa. The example is given in mod80 mode (LBCTL0.IMD0 = 0) with byte transfers.

In mode68 mode (LBCTL0.IMD0 = 1) the timing is equivalent, when the the $\overline{RD}$ strobe is considered as the low active E signal (LBCTL0.EL0 = 1)

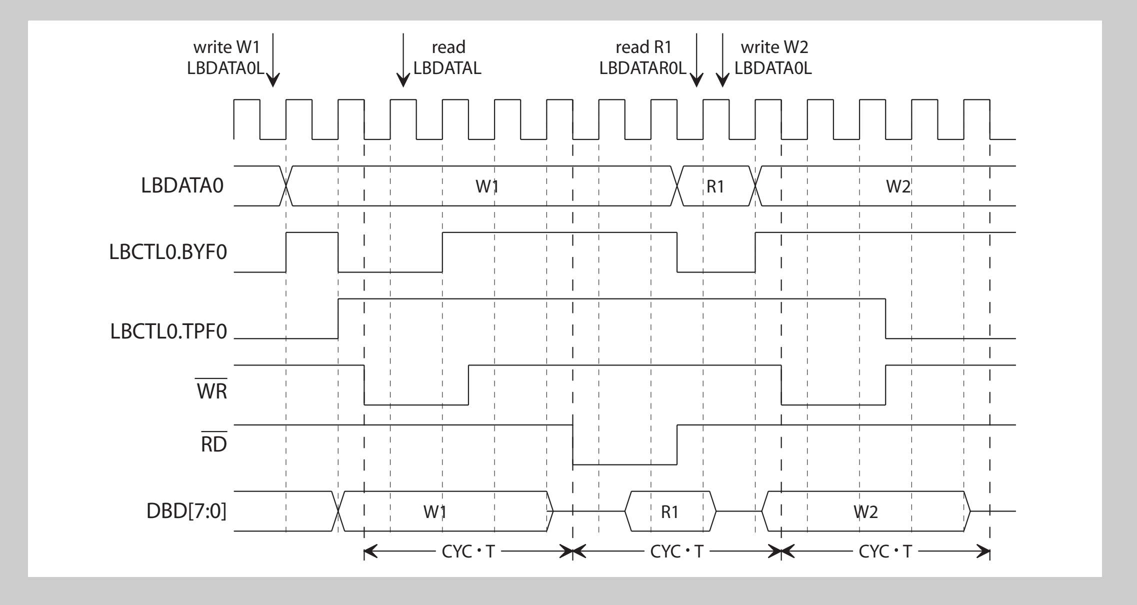


Figure 23-8 Timing (mod80: LBTCTL0.IMD0 = 0): byte write-read-write, LBWST0.WST = 4, LBCYC0.CYC = 7, LBTCTL0.TCIS = 0

Chapter 24 Sound Generator (SG)

The Sound Generator (SG0) generates an audio-frequency tone signal and a high-frequency pulse-width modulated (PWM) signal. The duty cycle of the PWM signal defines the volume.

By default, the two signal components are routed to separate pins. But both signals can also be combined to generate a composite signal that can be used to drive a loudspeaker circuit.

24.1 Overview

The Sound Generator consists of a programmable square wave tone generator and a programmable pulse-width modulator.

Features summary

Special features of the Sound Generator are:

- Programmable tone frequency (245 Hz to 6 KHz with a minimum step size of 20 Hz)
- Programmable volume level (9 bit resolution)
- Wide range of PWM signal frequency (32 KHz to 64 KHz)
- Sound can be stopped or retriggered
- Composite or separated frequency/volume output for external circuitry variation
- Hardware-optimized update of frequency and volume to avoid audible artifacts

24.1.1 Description

The following figure provides a functional block diagram of the Sound Generator.

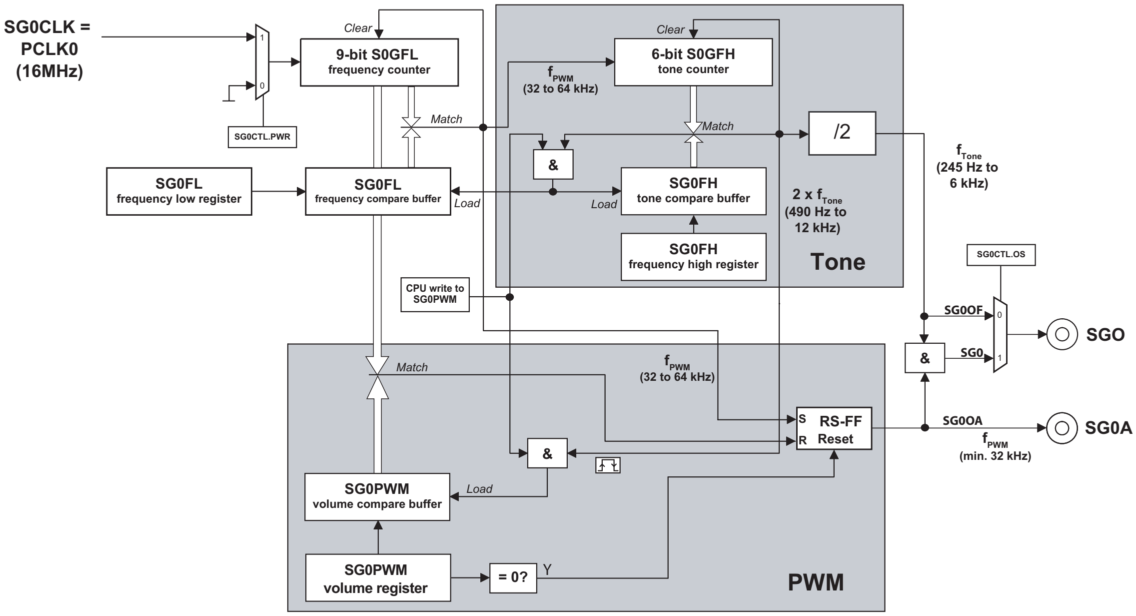


Figure 24-1 Sound Generator block diagram

The Sound Generator's input clock SG0CLK is the 16 MHz clock PCLK0.

Tone generator

The tone generator consists of two up-counters with compare registers. The values written to the frequency registers are automatically copied to compare buffers. The counters are reset to zero when their values match the contents of the associated compare buffers.

The 9-bit counter SG0FL generates a clock with a frequency between 32 KHz and 64 KHz. This clock constitutes the PWM frequency.

It is also the input of the second 6-bit counter SG0FH. The resulting tone signal behind the by-two-divider has a frequency between 245 Hz and 6 KHz and a 50 % duty cycle.

PWM

The PWM modulates the duty cycle according to the desired volume. It is controlled by the volume register SG0PWM. The value written to this register is automatically copied to the associated volume compare buffer.

The PWM continually compares the value of the counter SG0FL with the contents of its volume compare buffer.

The RS flipflop of the PWM is set by the pulses generated by the counter SG0FL. It is reset when the SG0FL counter value matches the contents of the volume buffer. Thus, the PWM output signal can have a duty cycle between 0 % (null volume) and 100 % (maximum volume).

The PWM frequency is above 32 KHz and hence outside the audible range.

Outputs

The Sound Generator is connected to the pins SGO and SGOA. By default, pin SGO provides the tone signal SG0OF and pin SGOA the PWM signal SG0OA that holds the volume (“amplitude”) information.

If bit SG0CTL.OS is set, pin SGO provides the composite signal SG0O that can directly control a speaker circuit.

24.1.2 Principle of operation

The software-controlled registers SG0FL, SG0FH, and SG0PWM are equipped with hardware buffers. The Sound Generator operates on these buffers.

This approach eliminates audible artifacts, because the buffers are only updated in synchronization with the generated tone waveform.

Note This section provides an overview. For details please refer to *"Sound Generator Operation" on page 849*.

(1) Generation of the tone frequency

The tone frequency is determined by two counters and their associated compare register values. Two counters are necessary to keep the tone pulse and the PWM signal synchronized.

The first counter (SG0FL) provides the input to the second (SG0FH) and also to the PWM. It is used to keep the PWM frequency outside the audio range (above 30 KHz) and within the signal bandwidth of the external sound system (usually below 64 KHz). Its match value defines also the 100 % volume level.

The second counter (SG0FH) generates the tone frequency (245 Hz to 6 KHz).

Note If the target values of the counters SG0FL/SG0FH are changed to generate a different tone frequency, the volume register SG0PWM has to be adjusted to keep the same volume.

(2) Generation of the volume information

The volume information (the "amplitude" of the audible signal) is provided as a high-frequency PWM signal. In composite mode, the PWM signal is ANDed with the tone signal, as illustrated in the following figure.

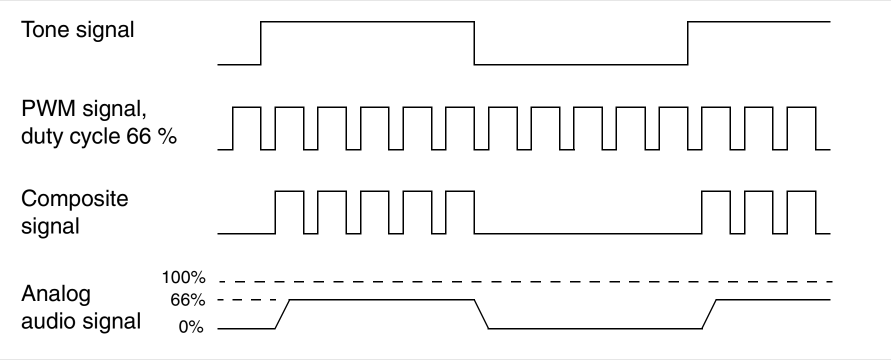


Figure 24-2 Generation of the composite output signal

After low-pass filtering, the analog signal amplitude corresponds to the duty cycle of the PWM signal. Low-pass filtering (averaging) is an inherent characteristic of a loudspeaker system.

The duty cycle can vary between 0 % and 100 %. Its generation is controlled by the counter register SG0FL and the volume register SG0PWM.

When the volume register SG0PWM is cleared, the sound stops immediately.

24.2 Sound Generator Registers

The Sound Generator is controlled by means of the following registers:

Table 24-1 **Sound Generator registers overview**

Register name	Shortcut	Address
SG0 frequency low register	SG0FL	<base>
SG0 frequency high register	SG0FH	<base> + 2_H
SG0 volume register	SG0PWM	<base> + 4_H
SG0 control register	SG0CTL	<base> + 7_H

Table 24-2 **Sound Generator register base address**

Module	Base address
SG0	FFFF $F5A0_H$

(1) SG0CTL - SG0 control register

The 8-bit SG0CTL register controls the operation of the Sound Generator.

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

Address <base> + 7_H

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

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

[a] The “0” value of this bit must not be changed!

Table 24-3 SG0CTL register contents

Bit position	Bit name	Function
4	PWR	Power save mode selection: 0: Clock input switched off (the Sound Generator is disabled and does not operate). 1: Clock input switched on (the Sound Generator is enabled and ready to use).
1	OS	SG0 output mode selection: 0: Selects SGOF and SGOA outputs (frequency and amplitude separated). 1: Selects SGO output (frequency and amplitude mixed).

Note Change the contents of this register only when the sound is stopped (register SG0PWM cleared).

(2) SG0FL - SG0 frequency low register

The 16-bit SG0FL register is used to specify the target value for the PWM frequency. It holds the target value for the 9-bit counter SG0FL.

Access This register is can be read/written in 16-bit units. It cannot be written if bit SG0CTL.PWR = 0.

The SG0FL register can also be read/written together with the SG0FH register by 32-bit access via the SG0F register.

Address <base>

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Counter SG0FL target value								
R	R	R	R	R	R	R	R/W								

For the calculation of the resulting PWM frequency refer to *“PWM calculations“ on page 852*.

The value written to SG0FL defines also the reference value for the maximum sound amplitude (100% PWM duty cycle). A 100 % duty cycle (continually high) will be generated if the SG0PWM value is higher than the SG0FL value. For details see *“PWM calculations“ on page 852*).

Note

1. The bits SG0FL[15:9] are not used.
2. The maximum value to be written is 510 ($01FE_H$). This yields a PWM frequency of 31.3 KHz. The minimum value to be written depends on the capability of the external circuit. A value of 255 ($00FF_H$) would yield a PWM frequency of 62.5 KHz.
3. The value read from this register does not necessarily reflect the current PWM frequency, because this frequency is determined by the frequency compare buffer value. The buffer might not be updated yet.
   For details see *“Updating the frequency buffer values“ on page 849*.

(3) SG0FH - SG0 frequency high register

The 16-bit SG0FH register is used to specify the final tone frequency. It holds the target value for the 6-bit counter SG0FH.

Access This register is can be read/written in 16-bit units. It cannot be written if bit SG0CTL.PWR = 0.

The SG0FH register can also be read/written together with the SG0FL register by 32-bit access to the SG0FL register via the SG0F register.

Address <base> + 2_H

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	Counter SG0FH target value					
R	R	R	R	R	R	R	R	R	R	R/W					

For the calculation of the resulting tone frequency refer to *“Tone frequency calculation“ on page 850*.

Note

1. The bits SG0FH[15:6] are not used.
2. Legal values depend on the contents of register SG0FL which defines the frequency of the input pulse. For example: If the counter SG0FL generates a frequency of 32.4 KHz, a value of 63 would generate a tone frequency of 253 Hz.
3. The value read from this register does not necessarily reflect the current tone frequency, because this frequency is determined by the frequency compare buffer value. The buffer might not be updated yet.
   For details see *“Updating the frequency buffer values“ on page 849*.

(4) SG0F - SG0 frequency register

The 32-bit register SG0F combines access to the 16-bit registers SG0FL and SG0H. This makes it possible to change the values for the PWM and tone frequency with one write access.

Access This register is can be read/written in 32-bit units. It cannot be written if bit SG0CTL.PWR = 0.

Address <base>

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

31	16	15	0
SG0FH		SG0FL	

(5) SG0PWM - SG0 volume register

The 16-bit register SG0PWM is used to specify the sound volume. It holds the target value for the sound amplitude that is given by the duty cycle of the PWM signal.

Access This register is can be read/written in 16-bit units. It cannot be written if bit SG0CTL.PWR = 0.

Address <base> + 4_H

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Sound volume target value								
R	R	R	R	R	R	R	R/W								

The value written to this register must be considered in conjunction with the contents of register SG0FL. The register SG0FL specifies the maximum value of the counter SG0FL.

For the calculation of the resulting duty cycle refer to *“PWM calculations“ on page 852*.

The setting takes effect after the SG0PWM buffer has been updated (see *“Updating the volume buffer value“ on page 851*).

Note

1. The bits 15:9 are not used.
2. The value read from this register does not necessarily reflect the current volume, because the value of counter SG0FL is compared with the contents of the volume buffer. The buffer might not be updated yet.
3. The sound stops immediately when this register is cleared.

24.3 Sound Generator Operation

This section explains the details of the Sound Generator.

24.3.1 Generating the tone

The tone signal is generated by the compare match signal of the SG0FH counter value with the value of the SG0FH buffer, followed by a by-two-divider. At each compare match, the counter is reset to zero.

Remember that the SG0FH counter is clocked by the output of the SG0FL counter.

(1) Updating the frequency buffer values

The values of the frequency buffers can be changed by writing to the associated frequency registers SG0FL and SG0FH. Both registers can be written together via SG0F.

Changing the value of the SG0FL (equivalent to SG0F[15:0]) register would also yield a change of the PWM frequency, i.e. the sound volume. Therefore it is obligatory to write the correct PWM value to SG0PWM before a new SG0F value is copied to the frequency buffers.

The SG0F register contents is copied to the buffers when the following sequence is detected:

1. CPU write access to SG0PWM register occurred.
2. SG0FH counter value and SG0FH buffer value have matched.
   This match is equivalent to the next edge (rising or falling) of the tone signal.

The following figure shows an example (not to scale).

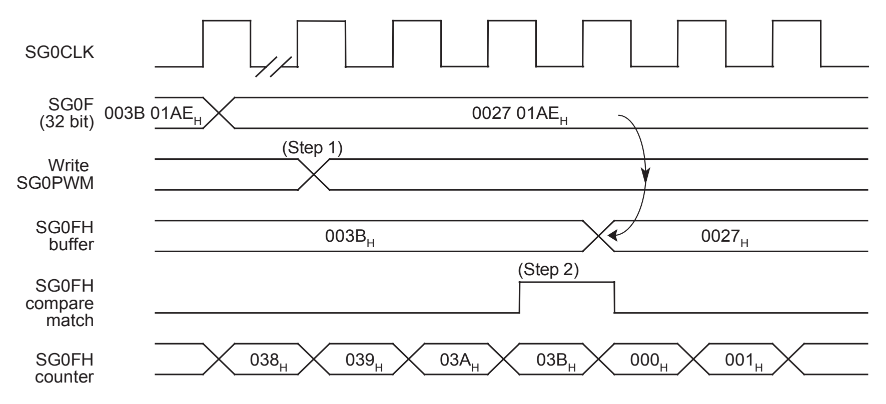


Figure 24-3 Update timing of the frequency buffers

Up to the next match, frequency registers and associated buffers can hold different values. If a 309 Hz tone is generated, as in the above example, the time span between writing to the SG0PWM register and updating the buffer can be up to 3.24 ms.

(2) Tone frequency calculation

The tone frequency can be calculated as:

$$f_{tone} = f_{SG0CLK} / (([SG0FL\ buffer] + 1) \times ([SG0FH\ buffer] + 1) \times 2)$$

where:

f_{SG0CLK} = frequency of the SG0 input clock

[SG0FL buffer] = contents of the SG0FL buffer

[SG0FH buffer] = contents of the SG0FH buffer

Example If:

- f_{SG0CLK} = 16 MHz
- [SG0FL buffer] = 255 ($00FF_H$) (this yields a PWM frequency of 62.5 KHz)
- [SG0FH buffer] = 32 (0020_H)

then:

- f_{tone} = 947 Hz

Note Note that the buffer contents can differ from the contents of the associated register until the next compare match.

24.3.2 Generating the volume information

The sound volume information is generated by comparing the SG0FL counter value with the contents of the SG0PWM volume buffer. An RS flipflop is set when the counter matches the SG0FL buffer and reset when the counter reaches the value of the volume buffer SG0PWM.

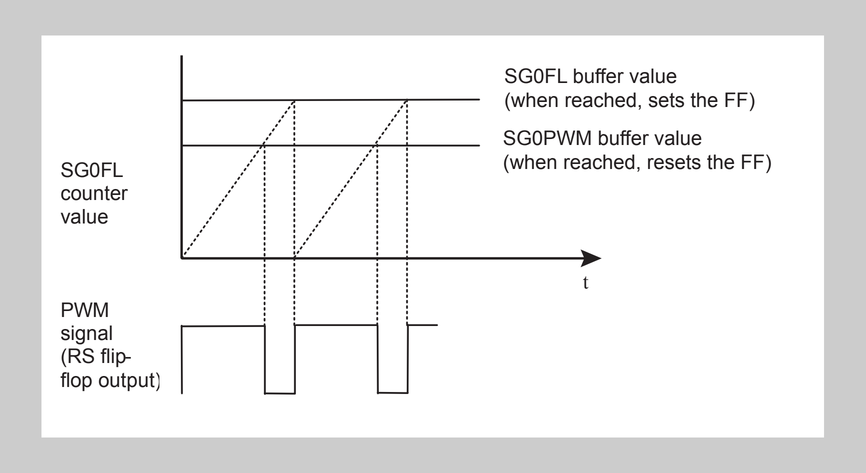


Figure 24-4 PWM signal generation

The duty cycle of the PWM signal is determined by the difference between the contents of the SG0FL counter buffer and the contents of the SG0PWM volume buffer. The larger the difference, the smaller the duty cycle.

The PWM signal is continually high when the value of the volume buffer is higher than the value of the frequency compare buffer.

Note To achieve 100 % duty cycle for all PWM frequencies, SGOFL must not be set to a value above $1FE_H$.

The PWM signal is continually low when the value of the volume buffer is zero—the sound has stopped.

(1) Updating the volume buffer value

The value of the volume compare buffer can be changed by writing to the volume register SG0PWM.

- If the register is cleared by writing 0000_H, the register value is copied to the volume compare buffer with the next rising edge of SG0CLK.
- As a result, the sound stops at the latest after one period of SG0CLK.
- If a non-zero value is written to the register, the buffer is updated with the next falling or rising edge of the tone frequency (match between SG0FH counter value and SG0FH buffer value).

(2) PWM calculations

PWM frequency The PWM frequency is generated by the counter SG0FL. It can be calculated as:

$f_{PWM} = f_{SG0CLK}$ / (([SG0FL buffer] + 1)

where:

f_{SG0CLK} = frequency of the SG0 input clock

[SG0FL buffer] = contents of the SG0FL buffer

Duty cycle The duty cycle of the PWM signal is calculated as follows:

- If [SG0PWM buffer] > [SG0FL buffer]:
  Duty cycle = 100 %
- If 0 ≤ [SG0PWM buffer] ≤ [SG0FL buffer]:
  Duty cycle = [SG0PWM buffer] / ([SG0FL buffer] + 1)

where:

[SG0PWM buffer] = contents of SG0PWM buffer

[SG0FL buffer] = contents of SG0FL buffer

Example If [SG0FL] is set to 240 ($00F0_H$), the following table applies:

Table 24-4 Duty cycle calculation example

[SG0PWM]	Calculation	Duty cycle [%]
$01FF_H$		100
...		100
$00F1_H$	241 / 241	100
$00F0_H$	240 / 241	99.6
$00EF_H$	239 / 241	99.2
...	...	...
0001_H	1 / 241	0.41
0000_H	0 / 241	0

The table shows, how the contents of register SG0FL affects the achievable volume resolution.

24.4 Sound Generator Application Hints

This section provides supplementary programming information.

24.4.1 Initialization

To enable the Sound Generator, set SG0CTL.PWR to 1. This connects the SG0 to the clock SG0CLK.

Check bit SG0CTL.OS.

When SG0CTL.OS is 0, the signal at pin SGO is a symmetrical square waveform with the frequency f_{tone}. When SG0CTL.OS is 1, the signal at pin SGO is composed of the tone signal and PWM pulses.

The frequency data registers SG0FL and SG0FH provide the buffer values for the counters. The combined value represents the frequency of the tone.

24.4.2 Start and stop sound

The sound is started by writing a non-zero value to the volume register SG0PWM.

Before starting the sound, all other register settings must be made.

The sound is stopped by writing 0000_H to the volume register SG0PWM. The sound is stopped regardless of the current value of amplitude output or frequency output. Thus, the sound can be stopped quickly, even if a very low sound frequency is chosen.

24.4.3 Change sound volume

The sound volume is changed by writing a new value to register SG0PWM.

The new volume takes effect with the next edge of the tone pulse (rising or falling).

24.4.4 Generate special sounds

To generate special sounds (like blinker clicks etc.), frequency and volume can be changed simultaneously.

To change the frequency of a sound that has already started:

1. Write to frequency register SG0FL in 32-bit mode (or to SG0FL and SG0FH separately in 16-bit mode).
2. Write to volume register SG0PWM.

Chapter 25 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.

25.1 Overview

The following table gives the naming convention of the pins:

Table 25-1 Naming convention of power supply pins

Dedicated function		V_{DD} or V_{SS}	5	n
<none>	CPU core, internal memory and peripherals	• VDD: Voltage Drain Drain • VSS: Voltage for Substrate and Source	level 5 V (nominative)	instance number
A	A/D Converter, Voltage Comparators			
B	Standard I/O buffer			
D	• µPD703420, µPD70(F)3421, µPD70(F)3422, µPD70F3423: LCD Controller/Driver driver I/O • µPD70F3424, µPD70F3425, µPD70F3426: LCD Bus interface I/O • µPD70F3427: D[31:16] ports, includes LCD Bus interface I/O			
SM	Stepper Motor Controller/Driver I/O			
M	• µPD70F3427: external memory I/F pins (except D[31:16] ports)			

The following pins belong to the Power Supply Scheme:

Table 25-2 Power supply pins

Pin	**Connected to**		
	µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423	**µPD70F3424, µPD70F3425, µPD70F3426**	**µPD70F3427**
VDD50 / VSS50	CPU core Pin pair is connected to voltage regulator 0.		
REGC0	Capacitor for voltage regulator 0 for pin pair VDD50 / VSS50.		
VDD51 / VSS51	CPU core Pin pair is connected to voltage regulator 0.		
REGC1	Capacitor for voltage regulator 1 for pin pair VDD51 / VSS51		
VDD52 / VSS52	Clock generation circuit and peripherals Power-on-clear circuit Pin pair is connected to a voltage regulator 1.		
REGC2	Capacitor for voltage regulator 2 for pin pair VDD52 / VSS52		
AVDD / AVSS	A/D Converter (power supply) Voltage Comparators		
AVREF	A/D Converter (reference input level)		
BVDD5n / BVSS5n	I/O buffer (n = 0, 1)		
DVDD50 / DVSS50	LCD Controller/Driver I/O	LCD Bus Interface I/O	D[31:16] ports, including LCD Bus Interface I/O
DVDD51 / DVSS51	–	–	
SMVDD5n / SMVSS5n	Stepper Motor Controller/Driver (n = 0, 1)		
MVDD5n / MVSS5n	–	–	External memory I/F, except D[31:16] (n = 0 to 4)

Note For electrical characteristics refer to the Electrical Target Specification.

25.2 Description

25.2.1 Devices µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423

Figure 25-1 gives an overview of the allocation of power supply pins of the µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423 devices. Their functional assignment is depicted in more detail in *Figure 25-2*.

Note The diagrams do not show the exact pin location.

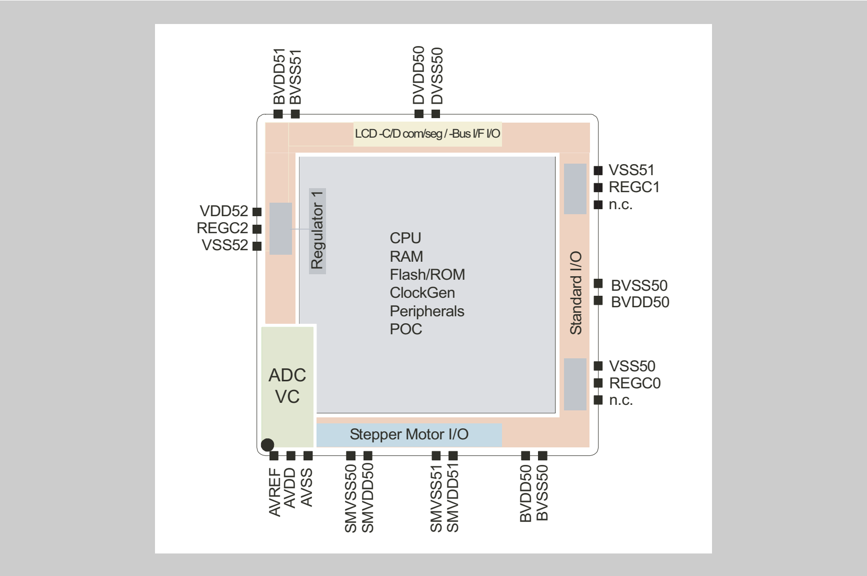


Figure 25-1 Power supply pins for µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423

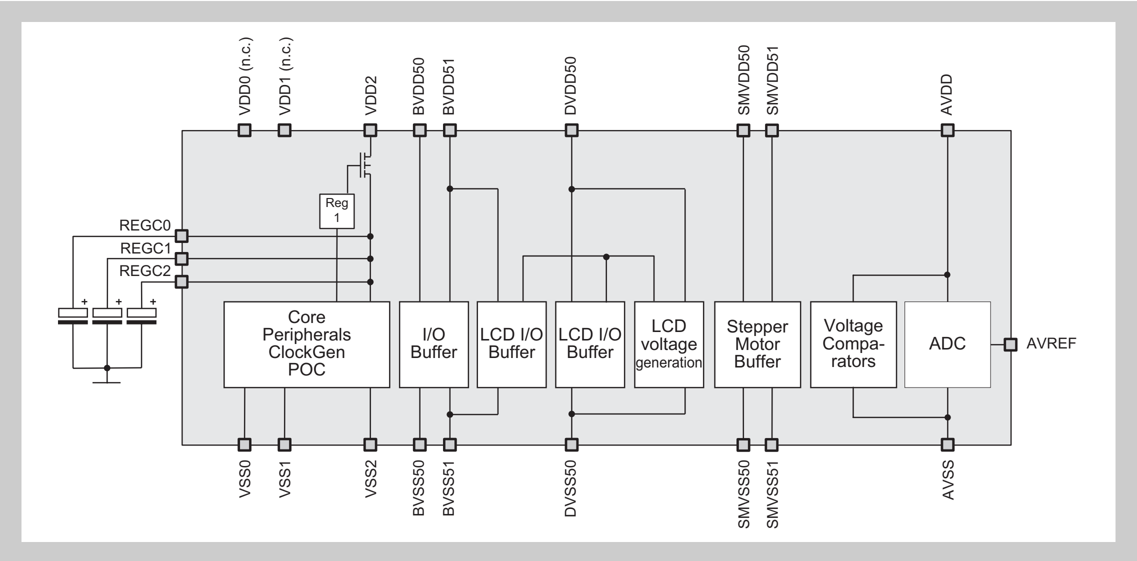


Figure 25-2 Functional assignment of power supply pins (µPD70(F)3420, µPD70(F)3421, µPD70(F)3422, µPD70F3423)

25.2.2 Devices µPD70F3424, µPD70F3425, µPD70F3426

Figure 25-3 gives an overview of the allocation of power supply pins of the µPD70F3424, µPD70F3425, µPD70F3426 devices. Their functional assignment is depicted in more detail in *Figure 25-4*.

Note The diagrams do not show the exact pin location.

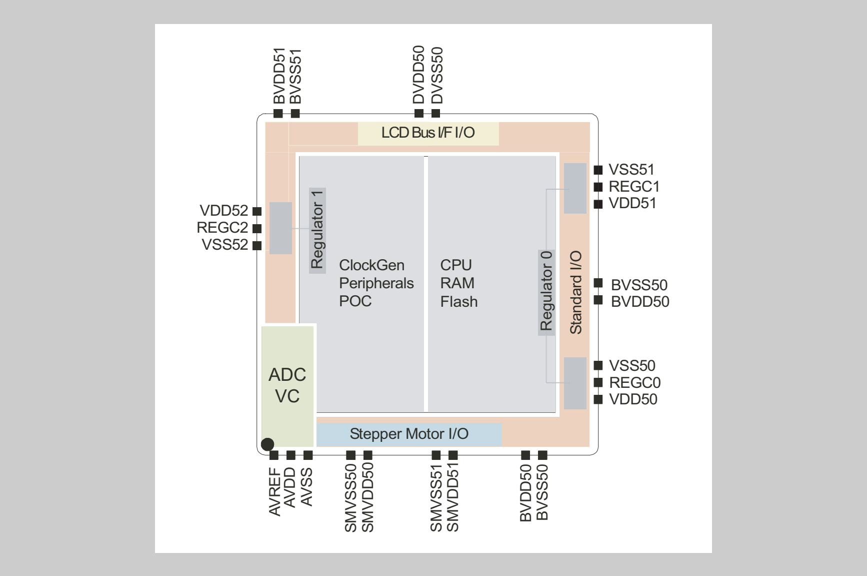


Figure 25-3 **Power supply pins for µPD70F3424, µPD70F3425, µPD70F3426**

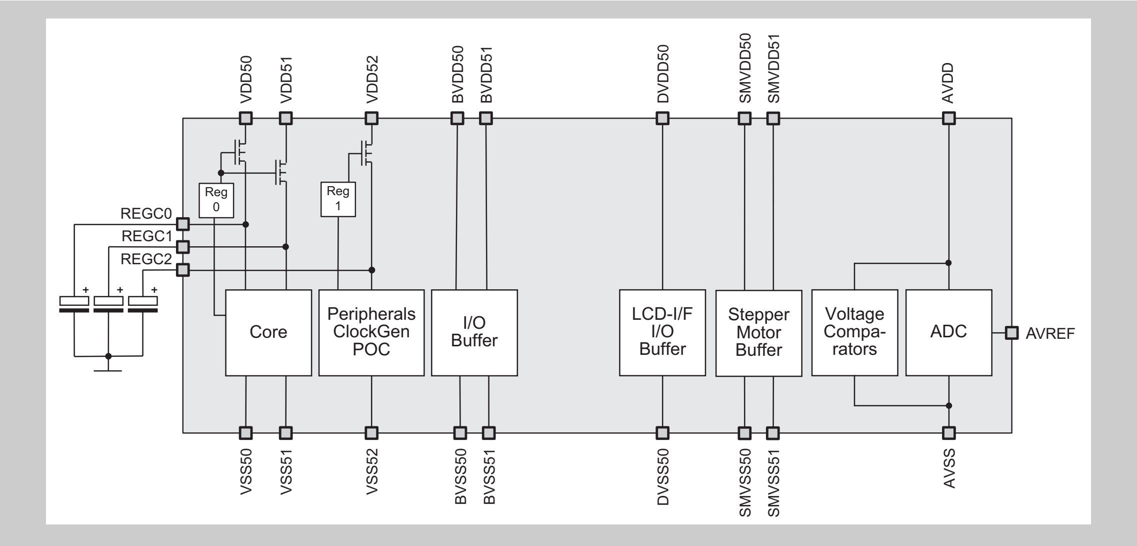


Figure 25-4 **Functional assignment of power supply pins (µPD70F3424, µPD70F3425, µPD70F3426)**

25.2.3 Device µPD70F3427

Figure 25-5 gives an overview of the allocation of power supply pins of the µPD70F3427 devices. Their functional assignment is depicted in more detail in *Figure 25-6*.

Note The diagrams do not show the exact pin location.

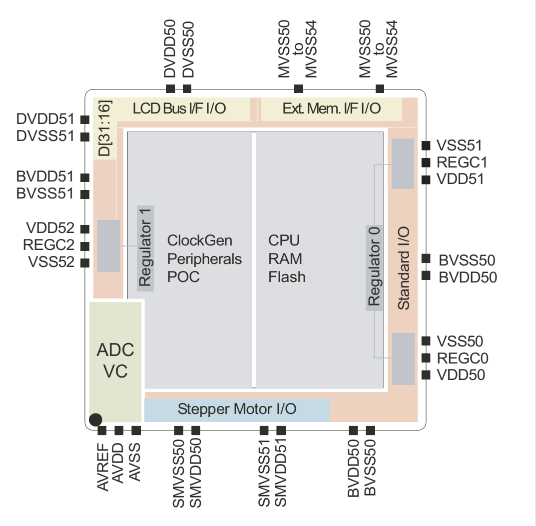


Figure 25-5 Power supply pins for µPD70F3427

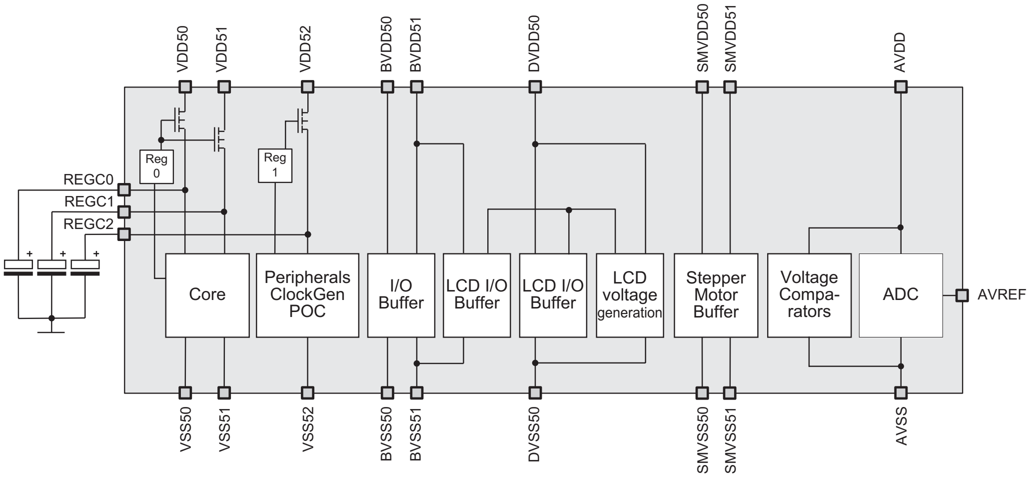


Figure 25-6 Functional assignment of power supply pins (µPD70F3427)

25.3 Voltage regulators

The on-chip voltage regulators generate the voltages for the internal circuitry (CPU core, clock generation circuit and peripherals), refer to *Figure 25-2*, *Figure 25-4* and *Figure 25-6*.

The regulators operate per default in all operation modes (normal operation, HALT, IDLE, STOP, WATCH, Sub-WATCH, and during RESET).

During power save modes the voltage regulators can be optionally disabled by setting the STBCTL register (refer to *“Control registers for power save modes” on page 157*).

Note To stabilize the output voltage of the regulator, connect a capacitor to the REGC pin. Refer to the Electrical Target Specification.

Chapter 26 Reset

Several system reset functions are provided in order to initialize hardware and registers.

26.1 Overview

Features summary

A reset can be caused by the following events:

- External reset signal $\overline{\text{RESET}}$
  Noise in the external reset signal is eliminated by an analog filter.
- Power-On-Clear (internal signal RESPOC)
- Overflow of the Watchdog Timer (internal signal RESWDT)
- Main or sub-oscillator fails (internal signals RESCMM, RESCMS)

As output, the reset function provides two internal reset signals:

- SYSRES (system reset)
- SYSRESWDT (Watchdog Timer reset)

26.1.1 General reset performance

The following figure shows the signals involved in the reset function:

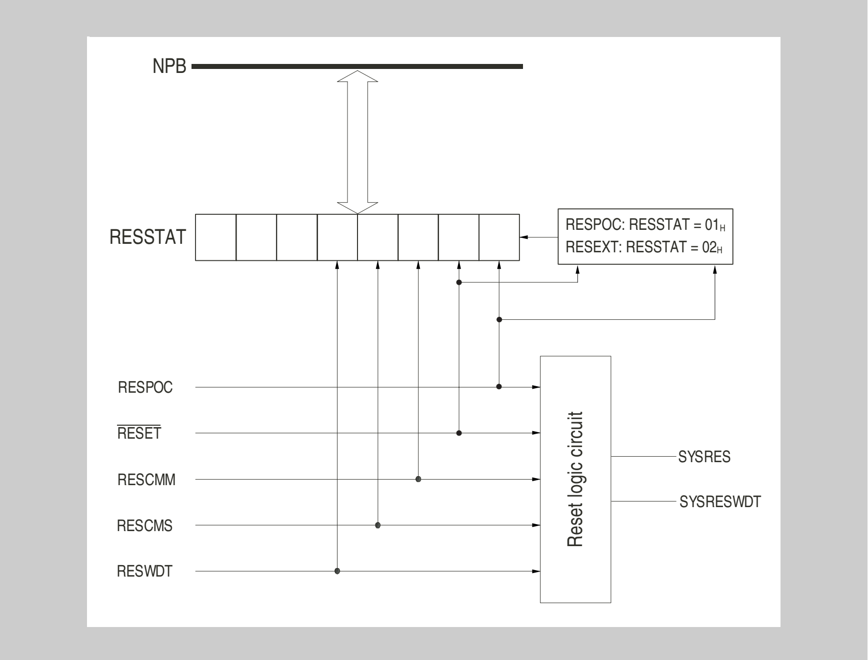


Figure 26-1 Reset function signal diagram

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.

The reset function provides two internal reset signals:

- System reset SYSRES
  SYSRES is activated by all reset sources.
- Watchdog reset SYSRESWDT
  SYSRESWDT is activated by Power-On-Clear and external $\overline{\text{RESET}}$ only.

Both resets provoke different reset behaviour of the Watchdog Timer. For details refer to the *“Watchdog Timer (WDT)“ on page 497*.

(1) Variable reset vector (flash memory devices only)

The flash memory devices allow to program the start address of the user’s program, instead of starting at address 0000 0000_H. The variable reset vector is stored in the extra area of the flash memory and can be written by an external flash programmer or in self-programming mode.

(2) Hardware status

With each reset function the hardware is initialized (including the watchdog). When the reset status is released, program execution is started.

The following table describes the status of the clocks during reset and after reset release. Note that the clock status "operates" does not inevitably mean that any function using this clock source operates as well. The function may additionally require to be enabled by other means.

Table 26-1 Hardware status during and after reset

Item	During reset	After reset
Main oscillator	Stops oscillation	Stopped[a]
Sub oscillator	Operates	Starts oscillation
Ring oscillator	Operates	Starts oscillation The ring oscillator clock is the default clock source after reset release.
SSCG clock	Stops operation	Stopped[a]
PLL clock	Stops operation	Stopped[a]
CPU system clock (VBCLK)	Stops operation	Starts oscillation based on the ring oscillator clock.
CPU	Initialized	Program execution starts after oscillation stabilization time.
Watchdog Timer (WDTCLK)	Stops operation	Starts operation based on ring oscillator clock
Watch Timer (WTCLK)	Stops operation	Starts operation based on ring oscillator clock
Peripheral clocks	Stop operation	• PCLK0–2: operating based on ring-osc • PCLK3-15: stopped • SPCLK0–2: operating based on ring-osc • SPCLK3-15: stopped
On-chip peripheral functions	Stop operation	Depends on availability of peripheral clock and default status of the peripheral function.
I/O pins (port/alternative function pins)	All pins are in input port mode[b]. See chapter *“Pin Functions“ on page 33* for a description.	

a) The main oscillator is started by the internal firmware. However the application software has to ensure stable main oscillation before utilizing this clock for any purpose. SSCG and PLL must be started by the application software. Assure also here that the stabilization time has passed. See chapter *“Clock Generator“ on page 129* for details.

b) 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 33* for details.

(3) Register status

With each reset function the registers of the CPU, internal RAM, and on-chip peripheral I/Os are initialized.

Since after reset the internal firmware is processed, some resources hold a different value as after reset, when the user's program is started. After a reset, make sure to set the registers to the values needed within your program.

Table 26-2 Initial values of CPU and internal RAM after reset

On-chip hardware		Register name	Initial value	
			After Reset	At start of user's program
CPU	Program registers	General-purpose register (r0)	0000 0000$_H$	0000 0000$_H$
		General-purpose registers (r1 to r31)	Undefined	Undefined
		Program counter (PC)	0000 0000$_H$	Variable reset vector programmed to flash extra area
	System registers	Status save registers during interrupt (EIPC, EIPSW)	Undefined	Undefined
		Status save registers during non-maskable interrupt (NMI) (FEPC, FEPSW)	Undefined	Undefined
		Interrupt cause register (ECR)	0000 0000$_H$	0000 0000$_H$
		Program status word (PSW)	0000 0020$_H$	• 0000 0020$_H$: if no security flags or variable reset vector are set • 0000 0021$_H$: else
		Status save registers during CALLT execution (CTPC, CTPSW)	Undefined	Undefined
		Status save registers during exception/debug trap (DBPC, DBPSW)	Undefined	Undefined
		CALLT base pointer (CTBP)	Undefined	Undefined
Internal RAM	After power-on	After Power-On-Clear reset the entire RAM contents is undefined.	Undefined	Undefined
	After $\overline{\text{RESET}}$	If a $\overline{\text{RESET}}$ occurs while writing to a RAM memory block, the contents of that RAM memory block may be corrupted. All other RAM memory blocks are not affected. Refer also to the note below the table.	All data in previous state	• 03FF 0000$_H$ - 03FF 07FF$_H$: undefined All other data in previous state or undefined (refer to note below).
	After any other reset	Any internal generated reset does not change the RAM contents.	All data in previous state	• 03FF 0000$_H$ - 03FF 07FF$_H$: undefined All other data in previous state.
Peripherals		Macro internal registers	The reset values of the various registers are given in the chapters of the peripheral functions	

Note In the table above, “Undefined” means either undefined at the time of a power-on reset, or undefined due to data destruction when the falling edge of the external $\overline{\text{RESET}}$ signal corrupts an ongoing RAM write access.
The internal RAM of the microcontroller comprises several separate RAM blocks. In case writing to one RAM block while a reset occurs the contents of only this RAM block may be corrupted. The other RAM blocks remain unchanged.

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 RESSTAT register is cleared and the RESSTAT.RESPOC bit is set (RESSTAT = 01_H, refer also to *“RESSTAT - Reset source flag register“ on page 868* for the interaction between Power-On-Clear and external $\overline{\text{RESET}}$). The system reset signals SYSRES and SYSRESWDT are generated.

Note

1. Depending on the supply voltage drop rate it may be required to apply an external $\overline{\text{RESET}}$ signal additionally in order to avoid microcontroller operation out of the specified operating conditions. For detailed electrical characteristics refer to the Electrical Target Specification.
2. POC shares the reference voltage supply with the power regulators.

The following figure shows the timing when a reset is performed at power-on.

The Power-On-Clear function holds the microcontroller in reset state as long as the power supply voltage does not exceed the threshold level V_{IP}.

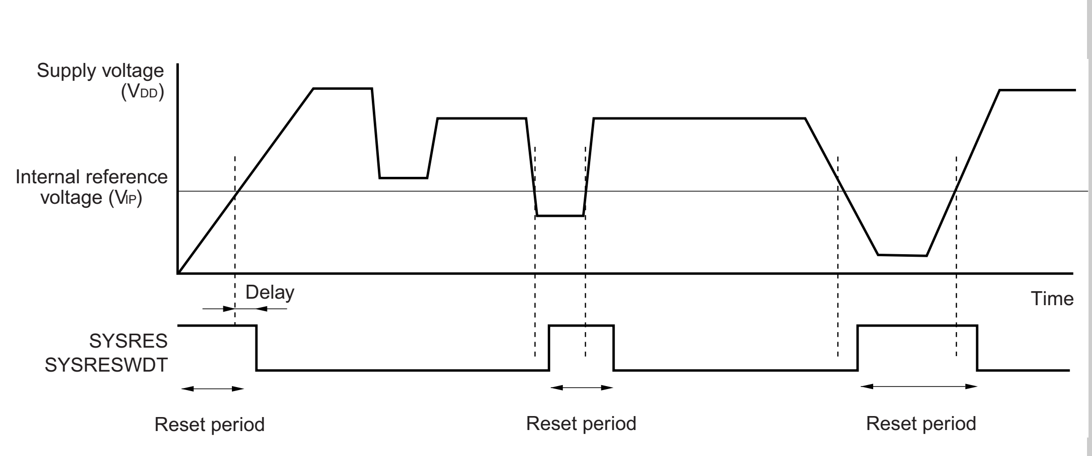


Figure 26-2 Timing of internal reset signal generation by Power-On-Clear circuit

26.1.3 External RESET

Reset is performed when a low level signal is applied to the RESET pin.

The reset status is released when the signal applied to the RESET pin changes from low to high.

After the external RESET is released, the RESSTAT register is cleared and the RESSTAT.RESEXT bit is set (RESSTAT = 02_H, refer also to *"RESSTAT - Reset source flag register" on page 868* for the interaction between Power-On-Clear and external RESET). The system reset signals SYSRES and SYSRESWDT are generated.

The RESET pin incorporates a noise eliminator, which is applied to the reset signal 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 RESET can keep the controller in reset state.

Note The internal system reset signals SYSRES and SYSRESWDT keep their active level for at least four system clock cycles after the RESET pin is released.

The following figure shows the timing when an external reset is performed. It explains the effect of the noise eliminator. The noise eliminator uses the analog delay to prevent the generation of an external reset due to noise.

The analog delay is caused by the analog input filter. The filter regards pulses up to a certain width as noise and suppresses them. For the minimum RESET pulse width refer to the Electrical Target Specification.

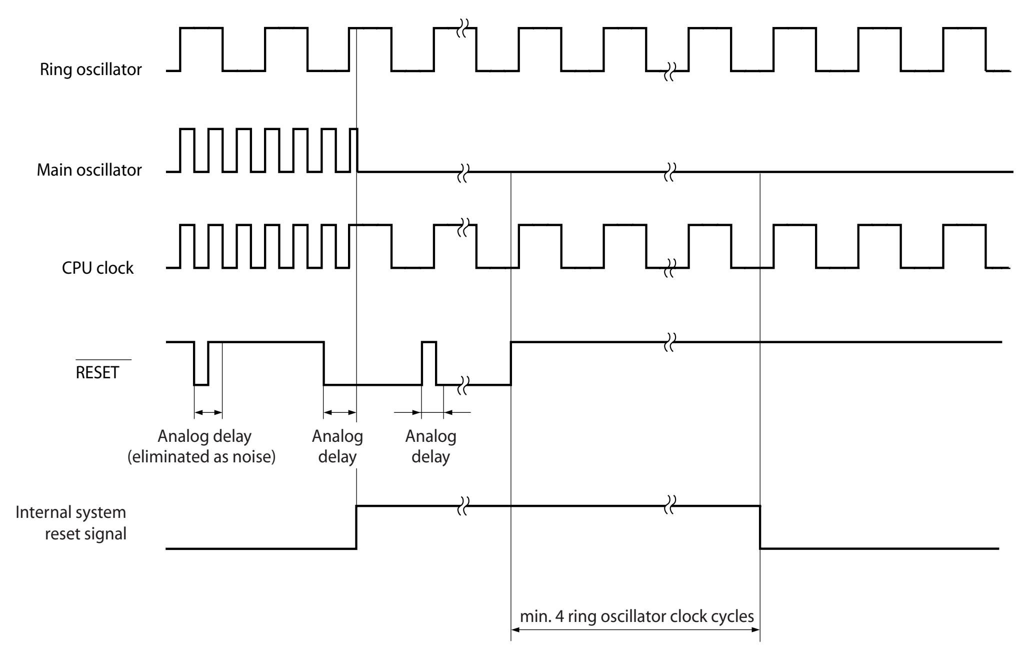


Figure 26-3 Timing for external RESET

26.1.4 Reset by Watchdog Timer

The Watchdog Timer can be configured to generate a reset if the watchdog time expires. After watchdog reset, the RESSTAT.RESWDT bit is set. The system reset signal SYSRES is generated.

After Watchdog Timer overflow, the reset status lasts for a specific time. Then the reset status is automatically released.

26.1.5 Reset by Clock Monitor

The two Clock Monitors generate a reset when either the main oscillator or the sub-oscillator fails. After a Clock Monitor reset, the corresponding bit (RESSTAT.RESCMM or RESSTAT.RESCMS) is set. The system reset signal SYSRES is generated.

After a Clock Monitor reset, the reset status lasts for a specific time. Then the reset status is automatically released.

26.2 Reset Registers

The reset functions are controlled and operated by means of the following registers:

Table 26-3 Reset function registers overview

Register name	Shortcut	Address
Reset source flag register	RESSTAT	FFFF $FF20_H$

(1) RESSTAT - Reset source flag register

The 8-bit RESSTAT register contains information about which type of resets occurred since the last Power-On-Clear or external $\overline{\text{RESET}}$ or after the last software clear of the register.

Each following reset condition sets the corresponding flag in the register. For example, if a Power-On-Clear reset is finished and then a Watchdog Timer reset occurs, the RESSTAT reads xxx1 0001_B.

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

Address FFFF $FF20_H$

Initial Value Power-On-Clear reset sets this register to 01_H.
External $\overline{\text{RESET}}$ sets this register to 02_H.

7	6	5	4	3	2	1	0
X	X	X	RESWDT	RESCM2	RESCM1	RESEXT	RESPOC
R	R	R[a]	R/W[a]	R/W[a]	R/W[a]	R/W[a]	R/W[a]

[a] Any write clears this register, independent of the data written.

Table 26-4 RESSTAT register contents

Bit position	Bit name	Function
4	RESWDT	Reset by Watchdog Timer 0: Not generated. 1: Generated.
3	RESCM2	Reset by Clock Monitor of sub oscillator 0: Not generated. 1: Generated.
2	RESCM1	Reset by Clock Monitor of main oscillator 0: Not generated. 1: Generated.
1	RESEXT	External $\overline{\text{RESET}}$ 0: Not generated. 1: Generated.
0	RESPOC	Reset at Power-On-Clear 0: Not generated. 1: Generated.

Note If clearing this register by writing and flag setting (occurrence of reset) conflict, flag setting takes precedence.

RESPOC and RESEXT Both Power-On-Clear and external $\overline{\text{RESET}}$ set RESSTAT to different initial states.

- Power-On-Clear reset sets RESSTAT = 01_H
- External $\overline{\text{RESET}}$ sets RESSTAT = 02_H

Special caution is required if both reset events are active concurrently:

- If the Power-On-Clear reset is longer active than the external $\overline{\text{RESET}}$: RESSTAT = 01_H. That means RESSTAT indicates only the occurrence of the Power-On-Clear reset.
- If the external $\overline{\text{RESET}}$ is longer active than the Power-On-Clear reset: RESSTAT = 02_H.That means RESSTAT indicates only the occurrence of the external $\overline{\text{RESET}}$.

- If the Power-On-Clear reset and external $\overline{\text{RESET}}$ has been released simultaneously: RESSTAT = 03_H. That means RESSTAT indicate the occurrence of both reset events.

All other reset events just set their respective bit in RESSTAT and do not change the others.

Chapter 27 Voltage Comparator

The microcontroller has two instances of a Voltage Comparator.

Note Throughout this chapter, the individual instances of the Voltage Comparator are identified by “n”, for example INTVCn for the generated interrupt signal.

27.1 Overview

The Voltage Comparator compares an external voltage V_{CMPn} at pin VCMPn and the internal reference voltage V_{LVI} and generates an interrupt if $V_{CMPn} < V_{LVI}$.

The comparison is mainly used to identify voltage drops of the external power supply. The CPU has then the possibility to reduce its own power consumption. By this it can avoid that its own power supply is dropping below the operating conditions.

Features summary The Voltage Comparator has the following special features:

- Comparison of an external voltage with the internal reference voltage
- Can be completely switched off (in order to achieve zero stand-by current)
- Shares the reference voltage supply with the power regulators
- Delivers status information to the CPU:
  - The compare result can be read by the CPU
  - An interrupt can be generated on falling edge, rising edge, or both edges of external supply voltage
  - Each Voltage Comparator generates a separate interrupt INTVCn
- Can operate in STOP mode.

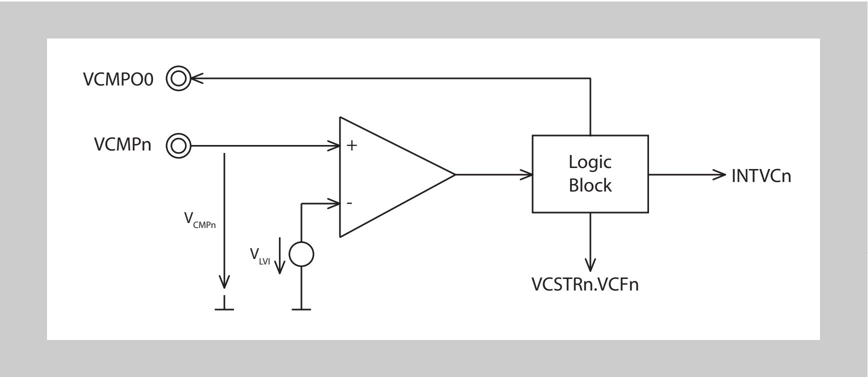
Note For details on the voltage levels refer to the Electrical Target Specification.

27.1.1 Description

Each Voltage Comparator consists of an operation amplifier and a logic block. The operation amplifier is connected to the external voltage (V_{CMPn}) with one input and to an internal reference voltage (V_{LVI}) with the other. It shares the reference voltage supply with the power regulators. The comparator output is fed into a logic block that generates the interrupt signal INTVCn and sets or clears the flag VCSTRn.VCFn. The comparison result of Voltage Comparator 0 is also output to the VCMPO0 pin.

The figure below shows a block diagram of the Voltage Comparator.

Figure 27-1 **Voltage**



mparator block diagram

27.1.2 Comparison results

Voltage comparison leads to the following results:

- Output signal VCMPO0 and flag VCSTRn.VCFn
  - $V_{CMPn} < V_{LVI}$:
    The output signal VCMPO0 of the Voltage Comparator is low (for n = 0) and the flag VCSTRn.VCFn is cleared.
  - $V_{CMPn} > V_{LVI}$:
    The output signal VCMPO0 of the Voltage Comparator is high (for n = 0) and the flag VCSTRn.VCFn is set.
- Interrupt signal INTVCn
  Depending on the settings of bits VCCTLn.ESn[1:0], the interrupt signal INTVCn is generated upon one or both of the above transitions of V_{CMPn}.

27.1.3 Stand-by mode

In order to reduce power consumption during STOP mode, the Voltage Comparator can be set into stand-by mode. This is done by setting VCCTLn.VCEn = 0.

If the Voltage Comparator is set in stand-by mode it assumes that $V_{CMPn} > V_{LVI}$ (VCSTRn.VCFn = 1 and VCMPO0 = high level).

27.2 Voltage Comparator Registers

The Voltage Comparator is controlled by means of the following registers:

Table 27-1 Voltage Comparator registers overview

Register name	Shortcut	Address
Voltage Comparator n control register	VCCTLn	<base>
Voltage Comparator n status register	VCSTRn	<base> + 2_H

Table 27-2 Base addresses of Voltage Comparator instances

Instance number	Base address
0	FFFF $FF10_H$
1	FFFF $FF14_H$

(1) VCCTLn - Voltage Comparator n control register

The 8-bit VCCTLn register controls whether the Voltage Comparator is operating or is in stand-by mode. Further it specifies whether an interrupt is generated when V_{CMPn} rises above or falls below V_{LVI} or any at of both transitions.

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

Address <base>

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

7	6	5	4	3	2	1	0
VCEn	0	0	0	0	0	ESn1	ESn0
R/W	R	R	R	R	R	R/W	R/W

Table 27-3 VCCTLn register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>VCEn</td><td>Enable Voltage Comparator
0: stand-by (VCSTRn.VCFn = 1, VCMPO0 = high level)
1: Operating</td></tr>
<tr><td>1 to 0</td><td>ESn[1:0]</td><td>VCMPO0 edge selection for interrupt
<table>
<tr><th>ESn1</th><th>ESn0</th><th>Operation</th></tr>
<tr><td>0</td><td>0</td><td>falling edge</td></tr>
<tr><td>0</td><td>1</td><td>rising edge</td></tr>
<tr><td>1</td><td>0</td><td>reserved</td></tr>
<tr><td>1</td><td>1</td><td>both edges</td></tr>
</table></td></tr>
</table>

Caution If the voltage comparator input level V_{CMPn} is below the reference voltage V_{LVI} an INTVCn interrupt is generated under both following conditions:

- The comparator is enabled (VCCTLn.VCEn = 0 → 1) and falling or both edges are specified (VCCTLn.VCEn = 00_B or 11_B).
- The comparator is disabled (VCCTLn.VCEn = 1 → 0) and rising or both edges are specified (VCCTLn.VCEn = 01_B or 11_B).

(2) VCSTRn - Voltage Comparator n status register

The 8-bit VCSTRn register reflects the result of the voltage comparison.

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

Address <base> + 2_H

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

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

Table 27-4 VCSTRn register contents

Bit position	Bit name	Function
0	VCFn	Voltage Comparator status flag 0: Input voltage is below reference voltage. 1: Input voltage is above reference voltage.

27.3 Timing

The following figure shows the timing of the Voltage Comparator 0. In this example, the interrupt INTVCn is generated at the falling edge (VCCTLn.ESTn[1:0] = 00_B) of the comparator's output signal.

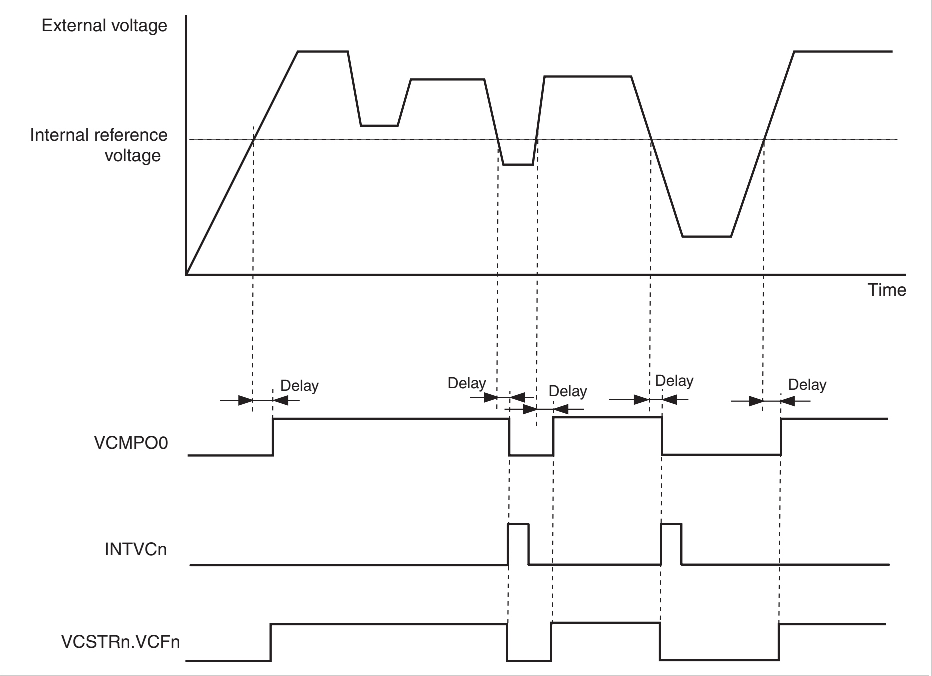


Figure 27-2 Voltage Comparator timing

Note For details on the delay time refer to the Electrical Target Specification.

Chapter 28 On-Chip Debug Unit

The microcontroller includes an on-chip debug unit. By connecting an N-Wire emulator, on-chip debugging can be executed.

28.1 Functional Outline

28.1.1 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 N-Wire 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 N-Wire emulator is used to connect the host PC to the on-chip debug unit.

(3) Forced reset function

The microcontroller can be forcibly reset.

(4) Break reset function

The CPU can be started in the debug mode immediately after reset of the CPU is released.

(5) Forced break function

Execution of the user program can be forcibly aborted.

(6) Hardware break function

Two breakpoints for instruction and data access can be used. The instruction breakpoint can abort program execution at any address. The access breakpoint can abort program execution by data access to any address.

(7) Software break function

Up to eight software breakpoints can be set in the internal flash memory area. The number of software breakpoints that can be set in the RAM area differs depending on the debugger to be used.

The software breakpoints utilize the “DBTRAP” ROM correction function. Thus following software breakpoints can be set:

- 8 breakpoints in the VFB flash/ROM address range
- 8 breakpoints in the VSB flash memory address range (µPD70F3426 only)

(8) Debug monitor function

A memory space for debugging that is different from the user memory space is used during debugging (background monitor mode). The user program can be executed starting from any address.

While execution of the user program is aborted, the user resources (such as memory and I/O) can be read and written, and the user program can be downloaded.

(9) Mask function

Each of the following signals can be masked. That means these signals will not be effective during debugging.

The correspondence with the mask functions of the debugger (ID850NWC) for the N-Wire emulator (IE-V850E1-CD-NW) of NEC Electronics is shown below.

– NMI0 mask function:	NMI pin
– NMIWDT mask function:	Watchdog Timer interrupt NMIWDT
– Reset mask function:	all reset sources

(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
- Functions that can operate or be stopped during break (however, each function cannot be selected individually)
  - A/D Converter
  - all timers P
  - all timers Z
  - Watch Timer
- 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 N-Wire interface is still accessible during power saving modes:

- N-Wire emulator can get status information from the on-chip debug unit.
- Stop mode can be released by the N-Wire emulator.

28.1.2 Security function

This microcontroller has a N-Wire security function, that demands the user to input an ID code upon start of the debugger. The ID code is compared to a predefined ID code, written in advance to the internal flash memory by an external flash programmer. This function prevents unauthorized persons to operate the microcontroller in N-Wire debug mode and to read the internal flash memory area.

The ID code in the internal flash memory can only be written by an external flash programmer. It can’t be changed in self-programming mode and therefore also not in N-Wire debugging mode.

ID code Be sure to write an ID code when writing a program to the internal flash memory.

The area of the ID code is 10 bytes wide and in the range of addresses 0000 0070_H to 0000 0079_H.

The ID code when the memory is erased is shown below.

Address	ID code
0000 0079_H	FF_H
0000 0078_H	FF_H
0000 0077_H	FF_H
0000 0076_H	FF_H
0000 0075_H	FF_H
0000 0074_H	FF_H
0000 0073_H	FF_H
0000 0072_H	FF_H
0000 0071_H	FF_H
0000 0070_H	FF_H

Security bit Bit 7 of address 0000 0079_H enables or disables use of the N-Wire emulator.

- Bit 7 of address 0000 0079_H

0: disabled	N-Wire emulator cannot connect to the on-chip debug unit.
1: enabled	N-Wire emulator can connect to the on-chip debug unit if the 10-byte ID code input matches the ID code stored in the flash memory

This security bit can only be modified by programming the flash memory via an external flash programmer. It is not possible to modify the security bit in self-programming mode, and therefore also not in N-Wire debugging mode.

After reset the entire ID code area is set to FF_H. This means that

- N-Wire debugging is generally enabled
- the ID code is FF_H for all ID code bytes

Consequently controller access is possible without any restriction.

Caution If access via the N-Wire interface should be disabled "block erase disabled" should be configured as well. Otherwise the flash memory blocks containing the ID code could be erased and N-Wire access could be enabled.

Security disable The entire ID code, i.e. also the security bit 7 of address 0000 0079_H, can be made temporarily ineffective by software. This is achieved by setting the control bit RSUDISC.DIS = 1. Setting RSUDISC.DIS = 1 does not change the security bit. Thus after a Power-On-Clear reset the N-Wire security is effective again.

The N-Wire security function can not be suspended when the microcontroller is operating in N-Wire debug mode.

(1) RSUDISC- N-Wire security disable control register

The 8-bit RSUDISC register is used to temporarily disable the N-Wire security function.

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 *“RSUDISCP - RSUDISC write protection register” on page 881* for details.

Address FFFF $F9E0_H$.

Initial Value 00_H. This register is cleared by Power-On-Clear reset.

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

Table 28-1 RSUDISC register contents

Bit position	Bit name	Function
0	DIS	N-Wire security function disable: 0: N-Wire security function enabled. 1: N-Wire security function disabled.

RSUDISC.DIS can not be changed, while the microcontroller is operating in N-Wire debug mode, i.e. while the concerned ports are operating as N-Wire debug pins (OCDM.OCDM0 = 1).

Thus proceed as follows to

- enable N-Wire debugging (from status OCDM.OCDM0 = 0):
  - set RSUDISC.DIS = 1 (disable N-Wire security)
  - set OCDM.OCDM0 = 1 (ports are N-Wire pins)
- disable N-Wire debugging (from status OCDM.OCDM0 = 1):
  - set OCDM.OCDM0 = 0 (ports are not N-Wire pins)
  - set RSUDISC.DIS = 0 (ensable N-Wire security)

(2) RSUDISCP - RSUDISC write protection register

The 8-bit RSUDISCP register protects the register RSUDISC from inadvertent write access.

After data has been written to the RSUDISCP register, the first write access to register RSUDISC is valid. All subsequent write accesses are ignored. Thus, the value of RSUDISC can only be rewritten in a specified sequence, and illegal write access is inhibited.

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

Address FFFF $FCA4_H$

Initial Value The contents of this register is undefined.

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

After writing to the RSUDISCP register, you are permitted to write once to RSUDISC. The write access to RSUDISC must happen with the immediately following instruction.

28.2 Controlling the N-Wire Interface

The N-Wire interface pins $\overline{\text{DRST}}$, DDI, DDO, DCK, DMS are shared with port functions, see Table 28-2. 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 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 28-2 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 OCDM0 control bit in the OCDM register determines the function of these device pins.

The register can be read or written in 8-bit and 1-bit units.

Address FFFF F9FC_H

	7	6	5	4	3	2	1	0
Bit name	0	0	0	0	0	0	0	OCDM0
Reset value	0	0	0	0	0	0	0	0/1[a]

[a] Reset value depends on reset source (see below)

OCDM0	Usage of N-Wire pins
0	Pins used as port/alternative function pins
1	Pins used as N-Wire interface pins

The reset value of OCDM.OCDM0 depends on the reset source.

(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 the port configuration register bit PFC0.PDC05.

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.
- In N-Wire debug mode the output current limiting function of the DDO pin is disabled. By this means the port pin provides maximum driver capability in order to maximize the transmission data rate to the N-Wire debugger. Note that the settings of the port registers are not affected.

Note This chapter describes the N-Wire interface control only. An additional security function decides, if the debugger access to the microcontroller is granted or not. Please refer to *"Code Protection and Security" on page 339*.

28.3 N-Wire Enabling Methods

28.3.1 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.

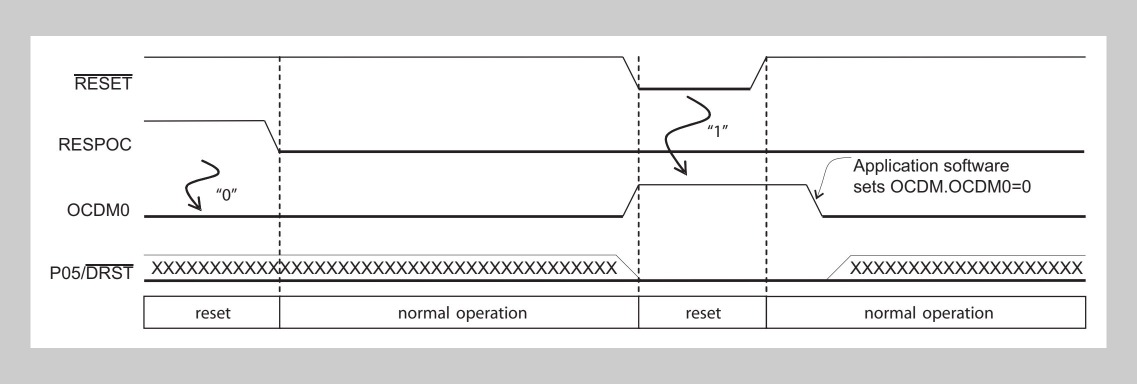


Figure 28-1 **Start without N-Wire activation**

28.3.2 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 $0000\ 0000_H$ with ROM mask devices respectively to the variable reset vector for flash memory devices.

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 28-2*. 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.

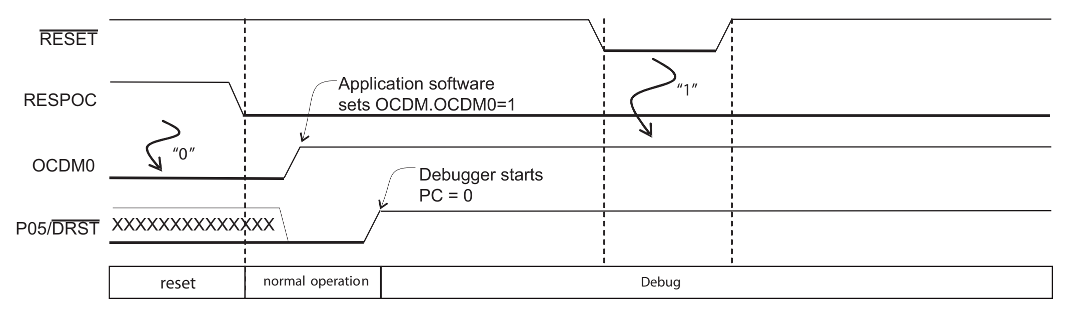


Figure 28-2 Start with N-Wire activation

28.3.3 N-Wire activation by RESET pin

The N-Wire interface can also be activated after power up by keeping RESET active for at least 2 sec after RESPOC release. 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.

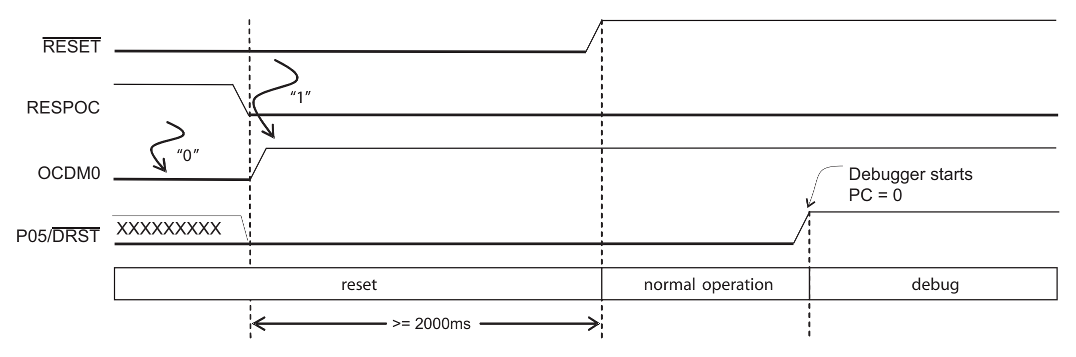


Figure 28-3 N-Wire activation by RESET pin

28.4 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.

28.4.1 KEL connector

KEL connector product names:

- 8830E-026-170S (KEL): straight type
- 8830E-026-170L (KEL): right-angle type

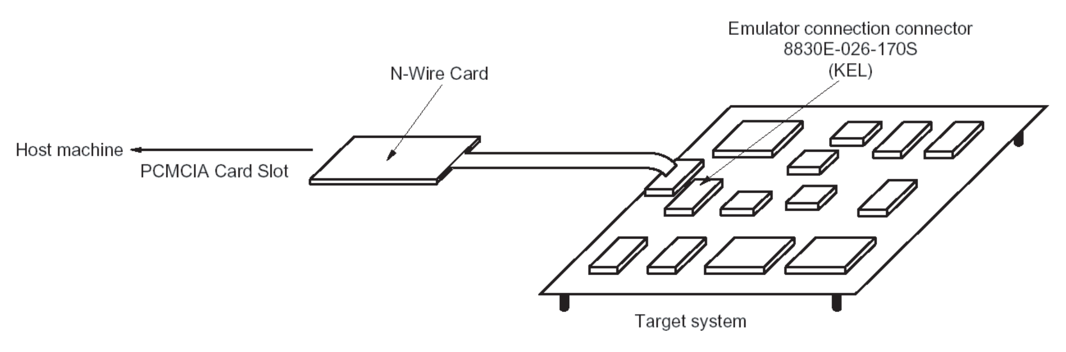


Figure 28-4 Connection to N-Wire emulator (NEC Electronics IE-V850E1-CD-NW: N-Wire Card)

(1) Pin configuration

Figure 28-5 shows the pin configuration of the connector for emulator connection (target system side), and *Table 28-3 on page 888* shows the pin functions.

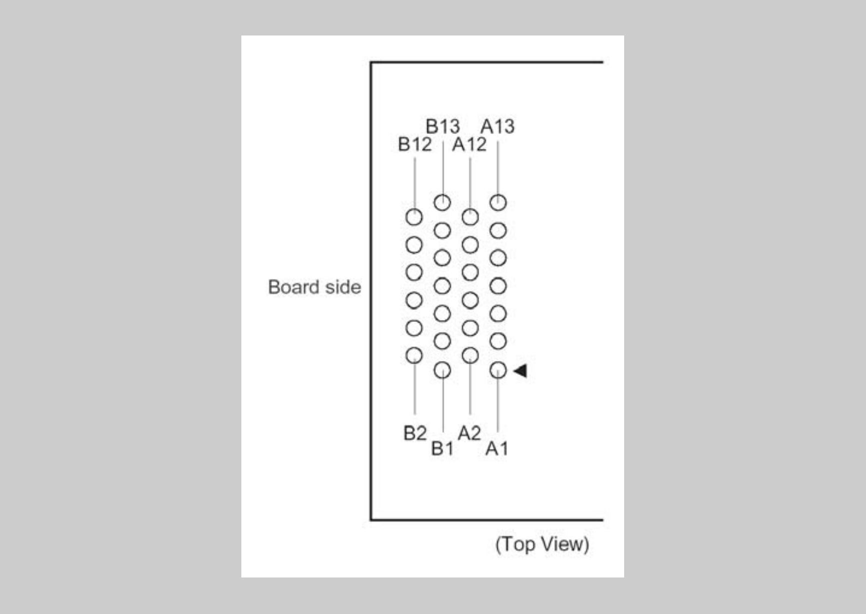


Figure 28-5 Pin configuration of connector for emulator connection (target system side)

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 28-3 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 28-6 on page 889* to set the OCDM0 bit to 1.)
A13	FLMD0[a]	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)

a) The FLMD0 signal is not required, if the N-Wire debugger serves the FLMD0 signal internally by using the SELFEN register to enable flash self-programming (refer to *“Flash Self-Programming” on page 234*). However the FLMD0 signal may be connected.

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.

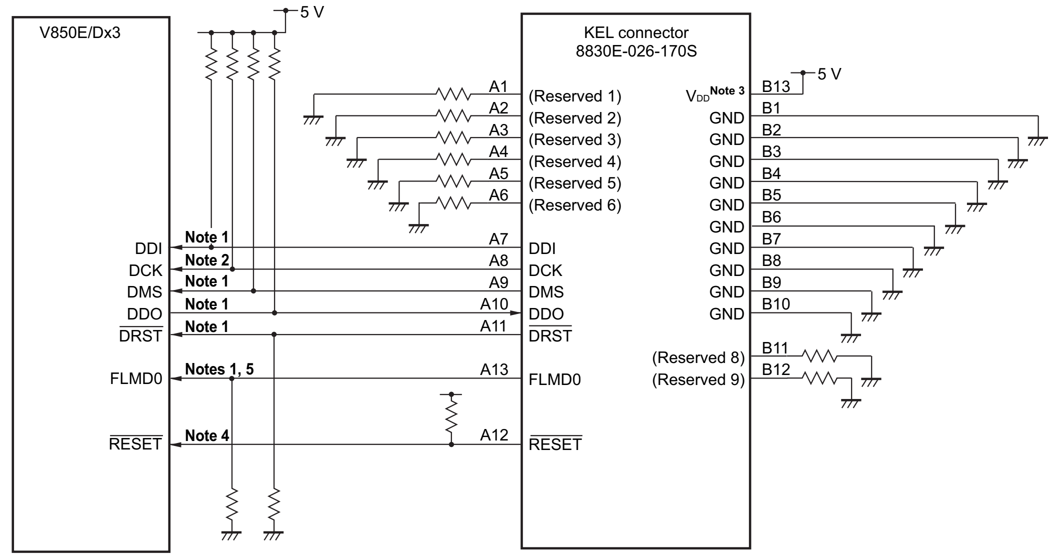


Figure 28-6 Example of recommended emulator connection circuit

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 28-6* to set the OCDM.OCDM0 bit to 1.
5. The FLMD0 signal is not required, but may be connected.

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.

28.5 Restrictions and Cautions on On-Chip Debug Function

- Do not mount a device that was used for debugging on a mass-produced product (this is because the flash memory was rewritten during debugging and the number of rewrites of the flash memory cannot be guaranteed).
- If a reset signal (reset input from the target system or reset by an internal reset source) is input during RUN (program execution), the break function may malfunction.
- Even if reset is masked by using a mask function, the I/O buffer (port pin, etc.) is reset when a pin reset signal is input.
- With a debugger that can set software breakpoints in the internal flash memory, the breakpoints temporarily become invalid when pin reset or internal reset is effected. The breakpoints become valid again if a break such as a hardware break or forced break is executed. Until then, no software break occurs.
- The $\overline{\text{RESET}}$ signal input is masked during a break.
- The 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 187*). Thus the debugger takes over control when one of the following exceptions occur:
  - debug trap (DBTRAP)
  - illegal op-code detection (ILGOP)
  - ROM Correction

  The debugger executes its own exception handler. Therefore, the user's exception handler at address 60_H will not be executed.

Appendix A Special Function Registers

The following tables list all registers that are accessed via the NPB (NEC peripheral bus). The registers are called “special function registers” (SFR).

Table A-1 lists all CAN special function registers. The addresses are given as offsets to programmable peripheral base address (refer to *“CAN module register and message buffer addresses“ on page 666*.

The tables list all registers and do not distinguish between the different derivatives.

A.1 CAN Registers

The CAN registers are accessible via the programmable peripheral area.

Table A-1 CAN special function registers (1/3)

Address offset	Register name	Shortcut	1	8	16	32
0x000	CAN0 Global Macro Control register	C0GMCTRL	-	-	R/W	-
0x000	CAN0 Global Macro Control register low byte	C0GMCTRLL	R/W	R/W	-	-
0x001	CAN0 Global Macro Control register high byte	C0GMCTRLH	R/W	R/W	-	-
0x002	CAN0 Global Macro Clock Selection register	C0GMCS	R/W	R/W	-	-
0x006	CAN0 Global Macro Automatic Block Transmission register	C0GMABT	-	-	R/W	-
0x006	CAN0 Global Macro Automatic Block Transmission register low byte	C0GMABTL	R/W	R/W	-	-
0x007	CAN0 Global Macro Automatic Block Transmission register high byte	C0GMABTH	R/W	R/W	-	-
0x008	CAN0 Global Macro Automatic Block Transmission Delay register	C0GMABTD	R/W	R/W	-	-
0x040	CAN0 Module Mask 1 register lower half word	C0MASK1L	-	-	R/W	-
0x042	CAN0 Module Mask 1 register upper half word	C0MASK1H	-	-	R/W	-
0x044	CAN0 Module Mask 2 register lower half word	C0MASK2L	-	-	R/W	-
0x046	CAN0 Module Mask 2 register upper half word	C0MASK2H	-	-	R/W	-
0x048	CAN0 Module Mask 3 register lower half word	C0MASK3L	-	-	R/W	-
0x04A	CAN0 Module Mask 3 register upper half word	C0MASK3H	-	-	R/W	-
0x04C	CAN0 Module Mask 4 register lower half word	C0MASK4L	-	-	R/W	-
0x04E	CAN0 Module Mask 4 register upper half word	C0MASK4H	-	-	R/W	-
0x050	CAN0 Module Control register	C0CTRL	-	-	R/W	-
0x052	CAN0 Module Last Error Code register	C0LEC	R/W	R/W	-	-
0x053	CAN0 Module Information register	C0INFO	R	R	-	-
0x054	CAN0 Module Error Counter	C0ERC	-	-	R/W	-
0x056	CAN0 Module Interrupt Enable register	C0IE	-	-	R/W	-
0x056	CAN0 Module Interrupt Enable register low byte	C0IEL	R/W	R/W	-	-

Table A-1 CAN special function registers (2/3)

Address offset	Register name	Shortcut	1	8	16	32
0x057	CAN0 Module Interrupt Enable register high byte	C0IEH	R/W	R/W	-	-
0x058	CAN0 Module Interrupt Status register	C0INTS	-	-	R/W	-
0x058	CAN0 Module Interrupt Status register low byte	C0INTSL	R/W	R/W	-	-
0x05A	CAN0 Module Bit-Rate Prescaler register	C0BRP	R/W	R/W	-	-
0x05C	CAN0 Bit Rate register	C0BTR	-	-	R/W	-
0x05E	CAN0 Module Last In-Pointer register	C0LIPT	-	R/W	-	-
0x060	CAN0 Module Receive History List Get Pointer register	C0RGPT	-	-	R/W	-
0x060	CAN0 Module Receive History List Get Pointer register low byte	C0RGPTL	R/W	R/W	-	-
0x062	CAN0 Module Last Out-Pointer register	C0LOPT	-	R	-	-
0x064	CAN0 Module Transmit History List Get Pointer register	C0TGPT	-	-	R/W	-
0x064	CAN0 Module Transmit History List Get Pointer register low byte	C0TGPTL	R/W	R/W	-	-
0x066	CAN0 Module Time Stamp register	C0TS	-	-	R/W	-
0x066	CAN0 Module Time Stamp register low byte	C0TSL	R/W	R/W	-	-
0x067	CAN0 Module Time Stamp register high byte	C0TSH	R/W	R/W	-	-
0x100 to 0x4EF	CAN0 Message Buffer registers, see *Table 19-20 on page 669*.					
0x600	CAN1 Global Macro Control register	C1GMCTRL	-	-	R/W	-
0x600	CAN1 Global Macro Control register low byte	C1GMCTRLL	R/W	R/W	-	-
0x601	CAN1 Global Macro Control register high byte	C1GMCTRLH	R/W	R/W	-	-
0x602	CAN1 Global Macro Clock Selection register	C1GMCS	R/W	R/W	-	-
0x606	CAN1 Global Macro Automatic Block Transmission register	C1GMABT	-	-	R/W	-
0x606	CAN1 Global Macro Automatic Block Transmission register low byte	C1GMABTL	R/W	R/W	-	-
0x607	CAN1 Global Macro Automatic Block Transmission register high byte	C1GMABTH	R/W	R/W	-	-
0x608	CAN1 Global Macro Automatic Block Transmission Delay register	C1GMABTD	R/W	R/W	-	-
0x640	CAN1 Module Mask 1 register lower half word	C1MASK1L	-	-	R/W	-
0x642	CAN1 Module Mask 1 register upper half word	C1MASK1H	-	-	R/W	-
0x644	CAN1 Module Mask 2 register lower half word	C1MASK2L	-	-	R/W	-
0x646	CAN1 Module Mask 2 register upper half word	C1MASK2H	-	-	R/W	-
0x648	CAN1 Module Mask 3 register lower half word	C1MASK3L	-	-	R/W	-
0x64A	CAN1 Module Mask 3 register upper half word	C1MASK3H	-	-	R/W	-
0x64C	CAN1 Module Mask 4 register lower half word	C1MASK4L	-	-	R/W	-
0x64E	CAN1 Module Mask 4 register upper half word	C1MASK4H	-	-	R/W	-
0x650	CAN1 Module Control register	C1CTRL	-	-	R/W	-
0x652	CAN1 Module Last Error Code register	C1LEC	R/W	R/W	-	-
0x653	CAN1 Module Information register	C1INFO	R	R	-	-
0x654	CAN1 Module Error Counter	C1ERC	-	-	R/W	-
0x656	CAN1 Module Interrupt Enable register	C1IE	-	-	R/W	-

Table A-1 CAN special function registers (3/3)

Address offset	Register name	Shortcut	1	8	16	32
0x656	CAN1 Module Interrupt Enable register low byte	C1IEL	R/W	R/W	-	-
0x657	CAN1 Module Interrupt Enable register high byte	C1IEH	R/W	R/W	-	-
0x658	CAN1 Module Interrupt Status register	C1INTS	-	-	R/W	-
0x658	CAN1 Module Interrupt Status register low byte	C1INTSL	R/W	R/W	-	-
0x65A	CAN1 Module Bit-Rate Prescaler register	C1BRP	R/W	R/W	-	-
0x65C	CAN1 Bit Rate register	C1BTR	-	-	R/W	-
0x65E	CAN1 Module Last In-Pointer register	C1LIPT	-	R/W	-	-
0x660	CAN1 Module Receive History List Get Pointer register	C1RGPT	-	-	R/W	-
0x660	CAN1 Module Receive History List Get Pointer register low byte	C1RGPTL	R/W	R/W	-	-
0x662	CAN1 Module Last Out-Pointer register	C1LOPT	-	R	-	-
0x664	CAN1 Module Transmit History List Get Pointer register	C1TGPT	-	-	R/W	-
0x664	CAN1 Module Transmit History List Get Pointer register low byte	C1TGPTL	R/W	R/W	-	-
0x666	CAN1 Module Time Stamp register	C1TS	-	-	R/W	-
0x666	CAN1 Module Time Stamp register low byte	C1TSL	R/W	R/W	-	-
0x667	CAN1 Module Time Stamp register high byte	C1TSH	R/W	R/W	-	-
0x700 to 0xAEF	CAN1 Message Buffer registers, see *Table 19-22 on page 671*.					

A.2 Other Special Function Registers

Table A-2 Other special function registers (1/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF060	CPU: Chip Area Select Control register 0	CSC0	-	-	R/W	-
0xFFFFF062	CPU: Chip Area Select Control register 1	CSC1	-	-	R/W	-
0xFFFFF064	CPU: Peripheral Area Select Control register	BPC	-	-	R/W	-
0xFFFFF066	CPU: Bus Size Configuration register	BSC	-	-	R/W	-
0xFFFFF068	CPU: Endian Configuration register	BEC	-	-	R/W	-
0xFFFFF06E	CPU: VPB Strobe Wait Control register	VSWC	R/W	R/W	-	-
0xFFFFF080	DMA source address register 0L	DSAL0	-	-	R/W	-
0xFFFFF082	DMA source address register 0H	DSAH0	-	-	R/W	-
0xFFFFF084	DMA destination address register 0L	DDAL0	-	-	R/W	-
0xFFFFF086	DMA destination address register 0H	DDAH0	-	-	R/W	-
0xFFFFF088	DMA source address register 1L	DSAL1	-	-	R/W	-
0xFFFFF08A	DMA source address register 1H	DSAH1	-	-	R/W	-
0xFFFFF08C	DMA destination address register 1L	DDAL1	-	-	R/W	-
0xFFFFF08E	DMA destination address register 1H	DDAH1	-	-	R/W	-
0xFFFFF090	DMA source address register 2L	DSAL2	-	-	R/W	-
0xFFFFF092	DMA source address register 2H	DSAH2	-	-	R/W	-
0xFFFFF094	DMA destination address register 2L	DDAL2	-	-	R/W	-
0xFFFFF096	DMA destination address register 2H	DDAH2	-	-	R/W	-
0xFFFFF098	DMA source address register 3L	DSAL3	-	-	R/W	-
0xFFFFF09A	DMA source address register 3H	DSAH3	-	-	R/W	-
0xFFFFF09C	DMA destination address register 3L	DDAL3	-	-	R/W	-
0xFFFFF09E	DMA destination address register 3H	DDAH3	-	-	R/W	-
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	-	-
0xFFFFF0F2	DMA restart register	DRST	R/W	R/W	-	-
0xFFFFF100	Interrupt Mask register 0	IMR0	-	-	R/W	-
0xFFFFF100	Interrupt Mask register 0L	IMR0L	R/W	R/W	-	-
0xFFFFF101	Interrupt Mask register 0H	IMR0H	R/W	R/W	-	-
0xFFFFF102	Interrupt Mask register 1	IMR1	-	-	R/W	-

Table A-2 Other special function registers (2/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF102	Interrupt Mask register 1L	IMR1L	R/W	R/W	-	-
0xFFFFF103	Interrupt Mask register 1H	IMR1H	R/W	R/W	-	-
0xFFFFF104	Interrupt Mask register 2	IMR2	-	-	R/W	-
0xFFFFF104	Interrupt Mask register 2L	IMR2L	R/W	R/W	-	-
0xFFFFF105	Interrupt Mask register 2H	IMR2H	R/W	R/W	-	-
0xFFFFF106	Interrupt Mask register 3	IMR3	-	-	R/W	-
0xFFFFF106	Interrupt Mask register 3L	IMR3L	R/W	R/W	-	-
0xFFFFF107	Interrupt Mask register 3H	IMR3H	R/W	R/W	-	-
0xFFFFF108	Interrupt Mask register 4	IMR4	-	-	R/W	-
0xFFFFF108	Interrupt Mask register 4L	IMR4L	R/W	R/W	-	-
0xFFFFF109	Interrupt Mask register 4H	IMR4H	R/W	R/W	-	-
0xFFFFF10A	Interrupt Mask register 5	IMR5	-	-	R/W	-
0xFFFFF10A	Interrupt Mask register 5L	IMR5L	R/W	R/W	-	-
0xFFFFF10B	Interrupt Mask register 5H	IMR5H	R/W	R/W	-	-
0xFFFFF110	Interrupt control register of INTVC0	VC0IC	R/W	R/W	-	-
0xFFFFF112	Interrupt control register of INTVC1	VC1IC	R/W	R/W	-	-
0xFFFFF114	Interrupt control register of INTWT0UV	WT0UVIC	R/W	R/W	-	-
0xFFFFF116	Interrupt control register of INTWT1UV	WT1UVIC	R/W	R/W	-	-
0xFFFFF118	Interrupt control register of INTTM00	TM00IC	R/W	R/W	-	-
0xFFFFF11A	Interrupt control register of INTTM01	TM01IC	R/W	R/W	-	-
0xFFFFF11C	Interrupt control register of INTP0	P0IC	R/W	R/W	-	-
0xFFFFF11E	Interrupt control register of INTP1	P1IC	R/W	R/W	-	-
0xFFFFF120	Interrupt control register of INTP2	P2IC	R/W	R/W	-	-
0xFFFFF122	Interrupt control register of INTP3	P3IC	R/W	R/W	-	-
0xFFFFF124	Interrupt control register of INTP4	P4IC	R/W	R/W	-	-
0xFFFFF126	Interrupt control register of INTP5	P5IC	R/W	R/W	-	-
0xFFFFF128	Interrupt control register of INTP6	P6IC	R/W	R/W	-	-
0xFFFFF12A	Interrupt control register of INTTZ0UV	TZ0UVIC	R/W	R/W	-	-
0xFFFFF12C	Interrupt control register of INTTZ1UV	TZ1UVIC	R/W	R/W	-	-
0xFFFFF12E	Interrupt control register of INTTZ2UV	TZ2UVIC	R/W	R/W	-	-
0xFFFFF130	Interrupt control register of INTTZ3UV	TZ3UVIC	R/W	R/W	-	-
0xFFFFF132	Interrupt control register of INTTZ4UV	TZ4UVIC	R/W	R/W	-	-
0xFFFFF134	Interrupt control register of INTTZ5UV	TZ5UVIC	R/W	R/W	-	-
0xFFFFF136	Interrupt control register of INTTP0OV	TP0OVIC	R/W	R/W	-	-
0xFFFFF138	Interrupt control register of INTTP0CC0	TP0CC0IC	R/W	R/W	-	-
0xFFFFF13A	Interrupt control register of INTTP0CC1	TP0CC1IC	R/W	R/W	-	-
0xFFFFF13C	Interrupt control register of INTTP1OV	TP1OVIC	R/W	R/W	-	-
0xFFFFF13E	Interrupt control register of INTTP1CC0	TP1CC0IC	R/W	R/W	-	-
0xFFFFF140	Interrupt control register of INTTP1CC1	TP1CC1IC	R/W	R/W	-	-
0xFFFFF142	Interrupt control register of INTTP2OV	TP2OVIC	R/W	R/W	-	-
0xFFFFF144	Interrupt control register of INTTP2CC0	TP2CC0IC	R/W	R/W	-	-

Table A-2 Other special function registers (3/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF146	Interrupt control register of INTTP2CC1	TP2CC1IC	R/W	R/W	-	-
0xFFFFF148	Interrupt control register of INTTP3OV	TP3OVIC	R/W	R/W	-	-
0xFFFFF14A	Interrupt control register of INTTP3CC0	TP3CC0IC	R/W	R/W	-	-
0xFFFFF14C	Interrupt control register of INTTP3CC1	TP3CC1IC	R/W	R/W	-	-
0xFFFFF14E	Interrupt control register of INTTG0OV0	TG0OV0IC	R/W	R/W	-	-
0xFFFFF150	Interrupt control register of INTTG0OV1	TG0OV1IC	R/W	R/W	-	-
0xFFFFF152	Interrupt control register of INTTG0CC0	TG0CC0IC	R/W	R/W	-	-
0xFFFFF154	Interrupt control register of INTTG0CC1	TG0CC1IC	R/W	R/W	-	-
0xFFFFF156	Interrupt control register of INTTG0CC2	TG0CC2IC	R/W	R/W	-	-
0xFFFFF158	Interrupt control register of INTTG0CC3	TG0CC3IC	R/W	R/W	-	-
0xFFFFF15A	Interrupt control register of INTTG0CC4	TG0CC4IC	R/W	R/W	-	-
0xFFFFF15C	Interrupt control register of INTTG0CC5	TG0CC5IC	R/W	R/W	-	-
0xFFFFF15E	Interrupt control register of INTTG1OV0	TG1OV0IC	R/W	R/W	-	-
0xFFFFF160	Interrupt control register of INTTG1OV1	TG1OV1IC	R/W	R/W	-	-
0xFFFFF162	Interrupt control register of INTTG1CC0	TG1CC0IC	R/W	R/W	-	-
0xFFFFF164	Interrupt control register of INTTG1CC1	TG1CC1IC	R/W	R/W	-	-
0xFFFFF166	Interrupt control register of INTTG1CC2	TG1CC2IC	R/W	R/W	-	-
0xFFFFF168	Interrupt control register of INTTG1CC3	TG1CC3IC	R/W	R/W	-	-
0xFFFFF16A	Interrupt control register of INTTG1CC4	TG1CC4IC	R/W	R/W	-	-
0xFFFFF16C	Interrupt control register of INTTG1CC5	TG1CC5IC	R/W	R/W	-	-
0xFFFFF172	Interrupt control register of INTAD	ADIC	R/W	R/W	-	-
0xFFFFF174	Interrupt control register of INTC0ERR	C0ERRIC	R/W	R/W	-	-
0xFFFFF176	Interrupt control register of INTC0WUP	C0WUPIC	R/W	R/W	-	-
0xFFFFF178	Interrupt control register of INTC0REC	C0RECIC	R/W	R/W	-	-
0xFFFFF17A	Interrupt control register of INTC0TRX	C0TRXIC	R/W	R/W	-	-
0xFFFFF17C	Interrupt control register of INTCB0RE	CB0REIC	R/W	R/W	-	-
0xFFFFF17E	Interrupt control register of INTCB0R	CB0RIC	R/W	R/W	-	-
0xFFFFF180	Interrupt control register of INTCB0T	CB0TIC	R/W	R/W	-	-
0xFFFFF182	Interrupt control register of INTUA0RE	UA0REIC	R/W	R/W	-	-
0xFFFFF184	Interrupt control register of INTUA0R	UA0RIC	R/W	R/W	-	-
0xFFFFF186	Interrupt control register of INTUA0T	UA0TIC	R/W	R/W	-	-
0xFFFFF188	Interrupt control register of INTUA1RE	UA1REIC	R/W	R/W	-	-
0xFFFFF18A	Interrupt control register of INTUA1R	UA1RIC	R/W	R/W	-	-
0xFFFFF18C	Interrupt control register of INTUA1T	UA1TIC	R/W	R/W	-	-
0xFFFFF18E	Interrupt control register of INTIIC0	IIC0IC	R/W	R/W	-	-
0xFFFFF190	Interrupt control register of INTIIC1	IIC1IC	R/W	R/W	-	-
0xFFFFF194	Interrupt control register of INTDMA0	DMA0IC	R/W	R/W	-	-
0xFFFFF196	Interrupt control register of INTDMA1	DMA1IC	R/W	R/W	-	-
0xFFFFF198	Interrupt control register of INTDMA2	DMA2IC	R/W	R/W	-	-
0xFFFFF19A	Interrupt control register of INTDMA3	DMA3IC	R/W	R/W	-	-
0xFFFFF19C	Interrupt control register of INTSW0	SW0IC	R/W	R/W	-	-

Table A-2 Other special function registers (4/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF19E	Interrupt control register of INTSW1	SW11IC	R/W	R/W	-	-
0xFFFFF1A0	Interrupt control register of INTP7	P7IC	R/W	R/W	-	-
0xFFFFF1A2	Interrupt control register of INTC1ERR	C1ERRIC	R/W	R/W	-	-
0xFFFFF1A4	Interrupt control register of INTC1WUP	C1WUPIC	R/W	R/W	-	-
0xFFFFF1A6	Interrupt control register of INTC1REC	C1RECIC	R/W	R/W	-	-
0xFFFFF1A8	Interrupt control register of INTC1TRX	C1TRXIC	R/W	R/W	-	-
0xFFFFF1AA	Interrupt control register of INTTZ6UV	TZ6UVIC	R/W	R/W	-	-
0xFFFFF1AC	Interrupt control register of INTTZ7UV	TZ7UVIC	R/W	R/W	-	-
0xFFFFF1AE	Interrupt control register of INTTZ8UV	TZ8UVIC	R/W	R/W	-	-
0xFFFFF1B0	Interrupt control register of INTTZ9UV	TZ9UVIC	R/W	R/W	-	-
0xFFFFF1B2	Interrupt control register of INTTG2OV0	TG2OV0IC	R/W	R/W	-	-
0xFFFFF1B4	Interrupt control register of INTTG2OV1	TG2OV1IC	R/W	R/W	-	-
0xFFFFF1B6	Interrupt control register of INTTG2CC0	TG2CC0IC	R/W	R/W	-	-
0xFFFFF1B8	Interrupt control register of INTTG2CC1	TG2CC1IC	R/W	R/W	-	-
0xFFFFF1BA	Interrupt control register of INTTG2CC2	TG2CC2IC	R/W	R/W	-	-
0xFFFFF1BC	Interrupt control register of INTTG2CC3	TG2CC3IC	R/W	R/W	-	-
0xFFFFF1BE	Interrupt control register of INTTG2CC4	TG2CC4IC	R/W	R/W	-	-
0xFFFFF1C0	Interrupt control register of INTTG2CC5	TG2CC5IC	R/W	R/W	-	-
0xFFFFF1C2	Interrupt control register of INTCB1RE	CB1REIC	R/W	R/W	-	-
0xFFFFF1C4	Interrupt control register of INTCB1R	CB1RIC	R/W	R/W	-	-
0xFFFFF1C6	Interrupt control register of INTCB1T	CB1TIC	R/W	R/W	-	-
0xFFFFF1C8	Interrupt control register of INTCB2RE	CB2REIC	R/W	R/W	-	-
0xFFFFF1CA	Interrupt control register of INTCB2R	CB2RIC	R/W	R/W	-	-
0xFFFFF1CC	Interrupt control register of INTCB2T	CB2TIC	R/W	R/W	-	-
0xFFFFF1CE	Interrupt control register of INTLCD	LCDIC	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	ADC mode register 0	ADA0M0	R/W	R/W	-	-
0xFFFFF201	ADC mode register 1	ADA0M1	R/W	R/W	-	-
0xFFFFF202	ADC channel select register	ADA0S	R/W	R/W	-	-
0xFFFFF203	ADC mode register 2	ADA0M2	R/W	R/W	-	-
0xFFFFF204	ADC power fail comparison mode register	ADA0PFM	R/W	R/W	-	-
0xFFFFF205	ADC power fail threshold register	ADA0PFT	R/W	R/W	-	-
0xFFFFF210	ADC result register channel 0	ADCR00	-	-	R	-
0xFFFFF211	ADC result register high byte channel 0	ADCR0H0	R	R	-	-
0xFFFFF212	ADC result register channel 1	ADCR01	-	-	R	-
0xFFFFF213	ADC result register high byte channel 1	ADCR0H1	R	R	-	-
0xFFFFF214	ADC result register channel 2	ADCR02	-	-	R	-
0xFFFFF215	ADC result register high byte channel 2	ADCR0H2	R	R	-	-
0xFFFFF216	ADC result register channel 3	ADCR03	-	-	R	-

Table A-2 Other special function registers (5/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF217	ADC result register high byte channel 3	ADCR0H3	R	R	-	-
0xFFFFF218	ADC result register channel 4	ADCR04	-	-	R	-
0xFFFFF219	ADC result register high byte channel 4	ADCR0H4	R	R	-	-
0xFFFFF21A	ADC result register channel 5	ADCR05	-	-	R	-
0xFFFFF21B	ADC result register high byte channel 5	ADCR0H5	R	R	-	-
0xFFFFF21C	ADC result register channel 6	ADCR06	-	-	R	-
0xFFFFF21D	ADC result register high byte channel 6	ADCR0H6	R	R	-	-
0xFFFFF21E	ADC result register channel 7	ADCR07	-	-	R	-
0xFFFFF21F	ADC result register high byte channel 7	ADCR0H7	R	R	-	-
0xFFFFF220	ADC result register channel 8	ADCR08	-	-	R	-
0xFFFFF221	ADC result register high byte channel 8	ADCRH08	R	R	-	-
0xFFFFF222	ADC result register channel 9	ADCR09	-	-	R	-
0xFFFFF223	ADC result register high byte channel 9	ADCR0H9	R	R	-	-
0xFFFFF224	ADC result register channel 10	ADCR010	-	-	R	-
0xFFFFF225	ADC result register high byte channel 10	ADCR0H10	R	R	-	-
0xFFFFF226	ADC result register channel 11	ADCR011	-	-	R	-
0xFFFFF227	ADC result register high byte channel 11	ADCR0H11	R	R	-	-
0xFFFFF228	ADC result register channel 12	ADCR012	-	-	R	-
0xFFFFF229	ADC result register high byte channel 12	ADCR0H12	R	R	-	-
0xFFFFF22A	ADC result register channel 13	ADCR013	-	-	R	-
0xFFFFF22B	ADC result register high byte channel 13	ADCR0H13	R	R	-	-
0xFFFFF22C	ADC result register channel 14	ADCR014	-	-	R	-
0xFFFFF22D	ADC result register high byte channel 14	ADCR0H14	R	R	-	-
0xFFFFF22E	ADC result register channel 15	ADCR015	-	-	R	-
0xFFFFF22F	ADC result register high byte channel 15	ADCR0H15	R	R	-	-
0xFFFFF300	Port Drive strength control register P0	PDSC0	R/W	R/W	-	-
0xFFFFF302	Port Drive strength control register P1	PDSC1	R/W	R/W	-	-
0xFFFFF304	Port Drive strength control register P2	PDSC2	R/W	R/W	-	-
0xFFFFF306	Port Drive strength control register P3	PDSC3	R/W	R/W	-	-
0xFFFFF308	Port Drive strength control register P4	PDSC4	R/W	R/W	-	-
0xFFFFF30A	Port Drive strength control register P5	PDSC5	R/W	R/W	-	-
0xFFFFF30C	Port Drive strength control register P6	PDSC6	R/W	R/W	-	-
0xFFFFF310	Port Drive strength control register P8	PDSC8	R/W	R/W	-	-
0xFFFFF312	Port Drive strength control register P9	PDSC9	R/W	R/W	-	-
0xFFFFF314	Port Drive strength control register P10	PDSC10	R/W	R/W	-	-
0xFFFFF344	Port LCD control register P2	PLCDC2	R/W	R/W	-	-
0xFFFFF346	Port LCD control register P3	PLCDC3	R/W	R/W	-	-
0xFFFFF348	Port LCD control register P4	PLCDC4	R/W	R/W	-	-
0xFFFFF34C	Port LCD control register port 6	PLCDC6	R/W	R/W	-	-
0xFFFFF350	Port LCD control register port 8	PLCDC8	R/W	R/W	-	-
0xFFFFF352	Port LCD control register port 9	PLCDC9	R/W	R/W	-	-

Table A-2 Other special function registers (6/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF354	Port LCD control register port 10	PLCDC10	R/W	R/W	-	-
0xFFFFF360	Port open drain control register P0	PODC0	R/W	R/W	-	-
0xFFFFF362	Port open drain control register P1	PODC1	R/W	R/W	-	-
0xFFFFF364	Port open drain control register P2	PODC2	R/W	R/W	-	-
0xFFFFF366	Port open drain control register P3	PODC3	R/W	R/W	-	-
0xFFFFF368	Port open drain control register P4	PODC4	R/W	R/W	-	-
0xFFFFF36A	Port open drain control register P5	PODC5	R/W	R/W	-	-
0xFFFFF36C	Port open drain control register P6	PODC6	R/W	R/W	-	-
0xFFFFF370	Port open drain control register P8	PODC8	R/W	R/W	-	-
0xFFFFF372	Port open drain control register P9	PODC9	R/W	R/W	-	-
0xFFFFF374	Port open drain control register P10	PODC10	R/W	R/W	-	-
0xFFFFF376	Port open drain control register P11	PODC11	R/W	R/W	-	-
0xFFFFF378	Port open drain control register P12	PODC12	R/W	R/W	-	-
0xFFFFF37A	Port open drain control register P13	PODC13	R/W	R/W	-	-
0xFFFFF380	Port input characteristic control register P0	PICC0	R/W	R/W	-	-
0xFFFFF382	Port input characteristic control register P1	PICC1	R/W	R/W	-	-
0xFFFFF384	Port input characteristic control register P2	PICC2	R/W	R/W	-	-
0xFFFFF386	Port input characteristic control register P3	PICC3	R/W	R/W	-	-
0xFFFFF388	Port input characteristic control register P4	PICC4	R/W	R/W	-	-
0xFFFFF38A	Port input characteristic control register P5	PICC5	R/W	R/W	-	-
0xFFFFF38C	Port input characteristic control register P6	PICC6	R/W	R/W	-	-
0xFFFFF390	Port input characteristic control register P8	PICC8	R/W	R/W	-	-
0xFFFFF392	Port input characteristic control register P9	PICC9	R/W	R/W	-	-
0xFFFFF394	Port input characteristic control register P10	PICC10	R/W	R/W	-	-
0xFFFFF396	Port input characteristic control register P11	PICC11	R/W	R/W	-	-
0xFFFFF398	Port input characteristic control register P12	PICC12	R/W	R/W	-	-
0xFFFFF39A	Port input characteristic control register P13	PICC13	R/W	R/W	-	-
0xFFFFF3B0	Port input level control register P8	PILC8	R/W	R/W	-	-
0xFFFFF3C0	Port pin read register P0	PPR0	R	R	-	-
0xFFFFF3C2	Port pin read register P1	PPR1	R	R	-	-
0xFFFFF3C4	Port pin read register P2	PPR2	R	R	-	-
0xFFFFF3C6	Port pin read register P3	PPR3	R	R	-	-
0xFFFFF3C8	Port pin read register P4	PPR4	R	R	-	-
0xFFFFF3CA	Port pin read register P5	PPR5	R	R	-	-
0xFFFFF3CC	Port pin read register P6	PPR6	R	R	-	-
0xFFFFF3D0	Port pin read register P8	PPR8	R	R	-	-
0xFFFFF3D2	Port pin read register P9	PPR9	R	R	-	-
0xFFFFF3D4	Port pin read register P10	PPR10	R	R	-	-
0xFFFFF3D6	Port pin read register P11	PPR11	R	R	-	-
0xFFFFF3D8	Port pin read register P12	PPR12	R	R	-	-
0xFFFFF3DA	Port pin read register P13	PPR13	R	R	-	-

Table A-2 Other special function registers (7/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF400	Port register port 0	P0	R/W	R/W	-	-
0xFFFFF402	Port register port 1	P1	R/W	R/W	-	-
0xFFFFF404	Port register port 2	P2	R/W	R/W	-	-
0xFFFFF406	Port register port 3	P3	R/W	R/W	-	-
0xFFFFF408	Port register port 4	P4	R/W	R/W	-	-
0xFFFFF40A	Port register port 5	P5	R/W	R/W	-	-
0xFFFFF40C	Port register port 6	P6	R/W	R/W	-	-
0xFFFFF40E	ADC port register ANI0 to ANI15	P7	-	-	R/W	-
0xFFFFF40E	ADC port register ANI0 to ANI7	P7L	R/W	R/W	-	-
0xFFFFF40F	ADC port register ANI8 to ANI15	P7H	R/W	R/W	-	-
0xFFFFF410	Port register port 8	P8	R/W	R/W	-	-
0xFFFFF412	Port register port 9	P9	R/W	R/W	-	-
0xFFFFF414	Port register port 10	P10	R/W	R/W	-	-
0xFFFFF416	Port register port 11	P11	R/W	R/W	-	-
0xFFFFF418	Port register port 12	P12	R/W	R/W	-	-
0xFFFFF41A	Port register port 13	P13	R/W	R/W	-	-
0xFFFFF420	Port mode register port 0	PM0	R/W	R/W	-	-
0xFFFFF422	Port mode register port 1	PM1	R/W	R/W	-	-
0xFFFFF424	Port mode register port 2	PM2	R/W	R/W	-	-
0xFFFFF426	Port mode register port 3	PM3	R/W	R/W	-	-
0xFFFFF428	Port mode register port 4	PM4	R/W	R/W	-	-
0xFFFFF42A	Port mode register port 5	PM5	R/W	R/W	-	-
0xFFFFF42C	Port mode register port 6	PM6	R/W	R/W	-	-
0xFFFFF430	Port mode register port 8	PM8	R/W	R/W	-	-
0xFFFFF432	Port mode register port 9	PM9	R/W	R/W	-	-
0xFFFFF434	Port mode register port 10	PM10	R/W	R/W	-	-
0xFFFFF436	Port mode register port 11	PM11	R/W	R/W	-	-
0xFFFFF438	Port mode register port 12	PM12	R/W	R/W	-	-
0xFFFFF43A	Port mode register port 13	PM13	R/W	R/W	-	-
0xFFFFF440	Port mode control register port 0	PMC0	R/W	R/W	-	-
0xFFFFF442	Port mode control register port 1	PMC1	R/W	R/W	-	-
0xFFFFF444	Port mode control register port 2	PMC2	R/W	R/W	-	-
0xFFFFF446	Port mode control register port 3	PMC3	R/W	R/W	-	-
0xFFFFF448	Port mode control register port 4	PMC4	R/W	R/W	-	-
0xFFFFF44A	Port mode control register port 5	PMC5	R/W	R/W	-	-
0xFFFFF44C	Port mode control register port 6	PMC6	R/W	R/W	-	-
0xFFFFF44E	Port mode control register port 7	PMC7	R/W	R/W	-	-
0xFFFFF450	Port mode control register port 8	PMC8	R/W	R/W	-	-
0xFFFFF452	Port mode control register port 9	PMC9	R/W	R/W	-	-
0xFFFFF454	Port mode control register port 10	PMC10	R/W	R/W	-	-
0xFFFFF456	Port mode control register port 11	PMC11	R/W	R/W	-	-

Table A-2 Other special function registers (8/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF458	Port mode control register port 12	PMC12	R/W	R/W	-	-
0xFFFFF45A	Port mode control register port 13	PMC13	R/W	R/W	-	-
0xFFFFF466	Port function control register port 3	PFC3	R/W	R/W	-	-
0xFFFFF46A	Port function control register port 5	PFC5	R/W	R/W	-	-
0xFFFFF46C	Port function control register port 6	PFC6	R/W	R/W	-	-
0xFFFFF480	Bus cycle type configuration register 0	BCT0	-	-	R/W	-
0xFFFFF482	Bus cycle type configuration register 1	BCT1	-	-	R/W	-
0xFFFFF484	Data wait control register 0	DWC0	-	-	R/W	-
0xFFFFF486	Data wait control register 1	DWC1	-	-	R/W	-
0xFFFFF488	Bus cycle control register	BCC	-	-	R/W	-
0xFFFFF48A	Address setting wait control register	ASC	-	-	R/W	-
0xFFFFF48E	Local bus size control register	LBS	-	-	R/W	-
0xFFFFF49A	Page ROM control register	PRC	-	-	R/W	-
0xFFFFF560	Synchronized counter read register WT0	WT0CNT0	-	-	R	-
0xFFFFF562	Non-synchronized counter read register WT0	WT0CNT1	-	-	R	-
0xFFFFF564	Counter reload register WT0	WT0R	-	-	R/W	-
0xFFFFF566	Control register WT0	WT0CTL	R/W	R/W	-	-
0xFFFFF570	Synchronized counter read register WT1	WT1CNT0	-	-	R	-
0xFFFFF572	Non-synchronized counter read register WT1	WT1CNT1	-	-	R	-
0xFFFFF574	Counter reload register WT1	WT1R	-	-	R/W	-
0xFFFFF576	Control register WT1	WT1CTL	R/W	R/W	-	-
0xFFFFF590	Watchdog timer Frequency select register	WDCS	R/W	R/W	-	-
0xFFFFF592	Watchdog timer security register	WCMD	R/W	R/W	-	-
0xFFFFF594	Watchdog timer mode register	WDTM	R/W	R/W	-	-
0xFFFFF596	Watchdog timer error register	WPHS	R/W	R/W	-	-
0xFFFFF5A0	SG0 Frequency register	SG0F	-	-	-	R/W
0xFFFFF5A0	SG0 Frequency register low	SG0FL	-	-	R/W	-
0xFFFFF5A2	SG0 Frequency register high	SG0FH	-	-	R/W	-
0xFFFFF5A4	SG0 Amplitude register	SG0PWM	-	-	R/W	-
0xFFFFF5A7	SG0 Control register	SG0CTL	R/W	R/W	-	-
0xFFFFF5C0	Timer Mode Control register 0	MCNTC00	R/W	R/W	-	-
0xFFFFF5C2	Compare register 1HW	MCMP01HW	-	-	R/W	-
0xFFFFF5C2	Compare register 10	MCMP010	-	R/W	-	-
0xFFFFF5C3	Compare register 11	MCMP011	-	R/W	-	-
0xFFFFF5C4	Compare register 2HW	MCMP02HW	-	-	R/W	-
0xFFFFF5C4	Compare register 20	MCMP020	-	R/W	-	-
0xFFFFF5C5	Compare register 21	MCMP021	-	R/W	-	-
0xFFFFF5C6	Compare register 3HW	MCMP03HW	-	-	R/W	-
0xFFFFF5C6	Compare register 30	MCMP030	-	R/W	-	-
0xFFFFF5C7	Compare register 31	MCMP031	-	R/W	-	-
0xFFFFF5C8	Compare register 4HW	MCMP04HW	-	-	R/W	-

Table A-2 Other special function registers (9/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF5C8	Compare register 40	MCMP040	-	R/W	-	-
0xFFFFF5C9	Compare register 41	MCMP041	-	R/W	-	-
0xFFFFF5CA	Compare Control register 1	MCMPC01	R/W	R/W	-	-
0xFFFFF5CC	Compare Control register 2	MCMPC02	R/W	R/W	-	-
0xFFFFF5CE	Compare Control register 3	MCMPC03	R/W	R/W	-	-
0xFFFFF5D0	Compare Control register 4	MCMPC04	R/W	R/W	-	-
0xFFFFF5D4	Timer Mode Control register 1	MCNTC01	R/W	R/W	-	-
0xFFFFF5D6	Compare register 5HW	MCMP05HW	-	-	R/W	-
0xFFFFF5D6	Compare register 50	MCMP050	-	R/W	-	-
0xFFFFF5D7	Compare register 51	MCMP051	-	R/W	-	-
0xFFFFF5D8	Compare register 6HW	MCMP06HW	-	-	R/W	-
0xFFFFF5D8	Compare register 60	MCMP060	-	R/W	-	-
0xFFFFF5D9	Compare register 61	MCMP061	-	R/W	-	-
0xFFFFF5DA	Compare Control register 5	MCMPC05	R/W	R/W	-	-
0xFFFFF5DC	Compare Control register 6	MCMPC06	R/W	R/W	-	-
0xFFFFF5E0	TM00 16-bit timer/counter register	TM00	-	-	R	-
0xFFFFF5E2	TM00 16-bit capture/compare register 0	CR000	-	-	R/W	-
0xFFFFF5E6	TM00 Control register	TMC00	R/W	R/W	-	-
0xFFFFF5E7	TM00 Prescaler mode register	PRM00	R/W	R/W	-	-
0xFFFFF5E8	TM00 Capture/Compare Control register	CRC00	R/W	R/W	-	-
0xFFFFF5E9	TM00 Timer Output Control register	TOC00	R/W	R/W	-	-
0xFFFFF600	TMZ0 Synchronized counter read register	TZ0CNT0	-	-	R	-
0xFFFFF602	TMZ0 non-synchronized counter read register	TZ0CNT1	-	-	R	-
0xFFFFF604	TMZ0 counter reload register	TZ0R	-	-	R/W	-
0xFFFFF606	TMZ0 control register	TZ0CTL	R/W	R/W	-	-
0xFFFFF608	TMZ1 Synchronized counter read register	TZ1CNT0	-	-	R	-
0xFFFFF60A	TMZ1 non-synchronized counter read register	TZ1CNT1	-	-	R	-
0xFFFFF60C	TMZ1 counter reload register	TZ1R	-	-	R/W	-
0xFFFFF60E	TMZ1 control register	TZ1CTL	R/W	R/W	-	-
0xFFFFF610	TMZ2 Synchronized counter read register	TZ2CNT0	-	-	R	-
0xFFFFF612	TMZ2 non-synchronized counter read register	TZ2CNT1	-	-	R	-
0xFFFFF614	TMZ2 counter reload register	TZ2R	-	-	R/W	-
0xFFFFF616	TMZ2 control register	TZ2CTL	R/W	R/W	-	-
0xFFFFF618	TMZ3 Synchronized counter read register	TZ3CNT0	-	-	R	-
0xFFFFF61A	TMZ3 non-synchronized counter read register	TZ3CNT1	-	-	R	-
0xFFFFF61C	TMZ3 counter reload register	TZ3R	-	-	R/W	-
0xFFFFF61E	TMZ3 control register	TZ3CTL	R/W	R/W	-	-
0xFFFFF620	TMZ4 Synchronized counter read register	TZ4CNT0	-	-	R	-
0xFFFFF622	TMZ4 non-synchronized counter read register	TZ4CNT1	-	-	R	-
0xFFFFF624	TMZ4 counter reload register	TZ4R	-	-	R/W	-
0xFFFFF626	TMZ4 control register	TZ4CTL	R/W	R/W	-	-

Table A-2 Other special function registers (10/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF628	TMZ5 Synchronized counter read register	TZ5CNT0	-	-	R	-
0xFFFFF62A	TMZ5 non-synchronized counter read register	TZ5CNT1	-	-	R	-
0xFFFFF62C	TMZ5 counter reload register	TZ5R	-	-	R/W	-
0xFFFFF62E	TMZ5 control register	TZ5CTL	R/W	R/W	-	-
0xFFFFF630	TMZ6 Synchronized counter read register	TZ6CNT0	-	-	R	-
0xFFFFF632	TMZ6 non-synchronized counter read register	TZ6CNT1	-	-	R	-
0xFFFFF634	TMZ6 counter reload register	TZ6R	-	-	R/W	-
0xFFFFF636	TMZ6 control register	TZ6CTL	R/W	R/W	-	-
0xFFFFF638	TMZ7 Synchronized counter read register	TZ7CNT0	-	-	R	-
0xFFFFF63A	TMZ7 non-synchronized counter read register	TZ7CNT1	-	-	R	-
0xFFFFF63C	TMZ7 counter reload register	TZ7R	-	-	R/W	-
0xFFFFF63E	TMZ7 control register	TZ7CTL	R/W	R/W	-	-
0xFFFFF640	TMZ8 Synchronized counter read register	TZ8CNT0	-	-	R	-
0xFFFFF642	TMZ8 non-synchronized counter read register	TZ8CNT1	-	-	R	-
0xFFFFF644	TMZ8 counter reload register	TZ8R	-	-	R/W	-
0xFFFFF646	TMZ8 control register	TZ8CTL	R/W	R/W	-	-
0xFFFFF648	TMZ9 Synchronized counter read register	TZ9CNT0	-	-	R	-
0xFFFFF64A	TMZ9 non-synchronized counter read register	TZ9CNT1	-	-	R	-
0xFFFFF64C	TMZ9 counter reload register	TZ9R	-	-	R/W	-
0xFFFFF64E	TMZ9 control register	TZ9CTL	R/W	R/W	-	-
0xFFFFF660	TMP0 timer control register 0	TP0CTL0	R/W	R/W	-	-
0xFFFFF661	TMP0 timer control register 1	TP0CTL1	R/W	R/W	-	-
0xFFFFF662	TMP0 timer-specific I/O control register 0	TP0IOC0	R/W	R/W	-	-
0xFFFFF663	TMP0 timer-specific I/O control register 1	TP0IOC1	R/W	R/W	-	-
0xFFFFF664	TMP0 timer-specific I/O control register 2	TP0IOC2	R/W	R/W	-	-
0xFFFFF665	TMP0 option register	TP0OPT0	R/W	R/W	-	-
0xFFFFF666	TMP0 capture/compare register 0	TP0CCR0	-	-	R/W	-
0xFFFFF668	TMP0 capture/compare register 1	TP0CCR1	-	-	R/W	-
0xFFFFF66A	TMP0 count register	TP0CNT	-	-	R	-
0xFFFFF670	TMP1 timer control register 0	TP1CTL0	R/W	R/W	-	-
0xFFFFF671	TMP1 timer control register 1	TP1CTL1	R/W	R/W	-	-
0xFFFFF672	TMP1 timer-specific I/O control register 0	TP1IOC0	R/W	R/W	-	-
0xFFFFF673	TMP1 timer-specific I/O control register 1	TP1IOC1	R/W	R/W	-	-
0xFFFFF674	TMP1 timer-specific I/O control register 2	TP1IOC2	R/W	R/W	-	-
0xFFFFF675	TMP1 option register	TP1OPT0	R/W	R/W	-	-
0xFFFFF676	TMP1 capture/compare register 0	TP1CCR0	-	-	R/W	-
0xFFFFF678	TMP1 capture/compare register 1	TP1CCR1	-	-	R/W	-
0xFFFFF67A	TMP1 count register	TP1CNT	-	-	R	-
0xFFFFF680	TMP2 timer control register 0	TP2CTL0	R/W	R/W	-	-
0xFFFFF681	TMP2 timer control register 1	TP2CTL1	R/W	R/W	-	-
0xFFFFF682	TMP2 timer-specific I/O control register 0	TP2IOC0	R/W	R/W	-	-

Table A-2 Other special function registers (11/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF683	TMP2 timer-specific I/O control register 1	TP2IOC1	R/W	R/W	-	-
0xFFFFF684	TMP2 timer-specific I/O control register 2	TP2IOC2	R/W	R/W	-	-
0xFFFFF685	TMP2 option register	TP2OPT0	R/W	R/W	-	-
0xFFFFF686	TMP2 capture/compare register 0	TP2CCR0	-	-	R/W	-
0xFFFFF688	TMP2 capture/compare register 1	TP2CCR1	-	-	R/W	-
0xFFFFF68A	TMP2 count register	TP2CNT	-	-	R	-
0xFFFFF690	TMP3 timer control register 0	TP3CTL0	R/W	R/W	-	-
0xFFFFF691	TMP3 timer control register 1	TP3CTL1	R/W	R/W	-	-
0xFFFFF692	TMP3 timer-specific I/O control register 0	TP3IOC0	R/W	R/W	-	-
0xFFFFF693	TMP3 timer-specific I/O control register 1	TP3IOC1	R/W	R/W	-	-
0xFFFFF694	TMP3 timer-specific I/O control register 2	TP3IOC2	R/W	R/W	-	-
0xFFFFF695	TMP3 option register	TP3OPT0	R/W	R/W	-	-
0xFFFFF696	TMP3 capture/compare register 0	TP3CCR0	-	-	R/W	-
0xFFFFF698	TMP3 capture/compare register 1	TP3CCR1	-	-	R/W	-
0xFFFFF69A	TMP3 count register	TP3CNT	-	-	R	-
0xFFFFF6A0	Timer mode register TMG 0	TMGM0	-	-	R/W	-
0xFFFFF6A0	Timer mode register TMG 0 low byte	TMGM0L	R/W	R/W	-	-
0xFFFFF6A1	Timer mode register TMG 0 high byte	TMGM0H	R/W	R/W	-	-
0xFFFFF6A2	Channel mode register TMG 0	TMGCM0	-	-	R/W	-
0xFFFFF6A2	Channel mode register TMG 0 low byte	TMGCM0L	R/W	R/W	-	-
0xFFFFF6A3	Channel mode register TMG 0 high byte	TMGCM0H	R/W	R/W	-	-
0xFFFFF6A4	Output control register TMG 0	OCTLG0	-	-	R/W	-
0xFFFFF6A4	Output control register TMG 0 low byte	OCTLG0L	R/W	R/W	-	-
0xFFFFF6A5	Output control register TMG 0 high byte	OCTLG0H	R/W	R/W	-	-
0xFFFFF6A6	Time base status register TMG 0	TMGST0	R	R	-	-
0xFFFFF6A8	Timer count register 0 TMG 0	TMG00	-	-	R	-
0xFFFFF6AA	Timer count register 1 TMG 0	TMG01	-	-	R	-
0xFFFFF6AC	Capture / Compare register 0 TMG 0	GCC00	-	-	R/W	-
0xFFFFF6AE	Capture / Compare register 1 TMG 0	GCC01	-	-	R/W	-
0xFFFFF6B0	Capture / Compare register 2 TMG 0	GCC02	-	-	R/W	-
0xFFFFF6B2	Capture / Compare register 3 TMG 0	GCC03	-	-	R/W	-
0xFFFFF6B4	Capture / Compare register 4 TMG 0	GCC04	-	-	R/W	-
0xFFFFF6B6	Capture / Compare register 5 TMG 0	GCC05	-	-	R/W	-
0xFFFFF6C0	Timer mode register TMG 1	TMGM1	-	-	R/W	-
0xFFFFF6C0	Timer mode register TMG 1 low byte	TMGM1L	R/W	R/W	-	-
0xFFFFF6C1	Timer mode register TMG 1 high byte	TMGM1H	R/W	R/W	-	-
0xFFFFF6C2	Channel mode register TMG 1	TMGCM1	-	-	R/W	-
0xFFFFF6C2	Channel mode register TMG 1 low byte	TMGCM1L	R/W	R/W	-	-
0xFFFFF6C3	Channel mode register TMG 1 high byte	TMGCM1H	R/W	R/W	-	-
0xFFFFF6C4	Output control register TMG 1	OCTLG1	-	-	R/W	-
0xFFFFF6C4	Output control register TMG 1 low byte	OCTLG1L	R/W	R/W	-	-

Table A-2 Other special function registers (12/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF6C5	Output control register TMG 1 high byte	OCTLG1H	R/W	R/W	-	-
0xFFFFF6C6	Time base status TMG 1	TMGST1	R	R	-	-
0xFFFFF6C8	Timer count register 0 TMG 1	TMG10	-	-	R	-
0xFFFFF6CA	Timer count register 1 TMG 1	TMG11	-	-	R	-
0xFFFFF6CC	Capture / Compare register 0 TMG 1	GCC10	-	-	R/W	-
0xFFFFF6CE	Capture / Compare register 1 TMG 1	GCC11	-	-	R/W	-
0xFFFFF6D0	Capture / Compare register 2 TMG 1	GCC12	-	-	R/W	-
0xFFFFF6D2	Capture / Compare register 3 TMG 1	GCC13	-	-	R/W	-
0xFFFFF6D4	Capture / Compare register 4 TMG 1	GCC14	-	-	R/W	-
0xFFFFF6D6	Capture / Compare register 5 TMG 1	GCC15	-	-	R/W	-
0xFFFFF6E0	Timer mode register TMG 2	TMGM2	-	-	R/W	-
0xFFFFF6E0	Timer mode register TMG 2 low byte	TMGM2L	R/W	R/W	-	-
0xFFFFF6E1	Timer mode register TMG 2 high byte	TMGM2H	R/W	R/W	-	-
0xFFFFF6E2	Channel mode register TMG 2	TMGCM2	-	-	R/W	-
0xFFFFF6E2	Channel mode register TMG 2 low byte	TMGCM2L	R/W	R/W	-	-
0xFFFFF6E3	Channel mode register TMG 2 high byte	TMGCM2H	R/W	R/W	-	-
0xFFFFF6E4	Output control register TMG 2	OCTLG2	-	-	R/W	-
0xFFFFF6E4	Output control register TMG 2 low byte	OCTLG2L	R/W	R/W	-	-
0xFFFFF6E5	Output control register TMG 2 high byte	OCTLG2H	R/W	R/W	-	-
0xFFFFF6E6	Time base status TMG 2	TMGST2	R	R	-	-
0xFFFFF6E8	Timer count register 0 TMG 2	TMG20	-	-	R	-
0xFFFFF6EA	Timer count register 1 TMG 2	TMG21	-	-	R	-
0xFFFFF6EC	Capture / Compare register 0 TMG 2	GCC20	-	-	R/W	-
0xFFFFF6EE	Capture / Compare register 1 TMG 2	GCC21	-	-	R/W	-
0xFFFFF6F0	Capture / Compare register 2 TMG 2	GCC22	-	-	R/W	-
0xFFFFF6F2	Capture / Compare register 3 TMG 2	GCC23	-	-	R/W	-
0xFFFFF6F4	Capture / Compare register 4 TMG 2	GCC24	-	-	R/W	-
0xFFFFF6F6	Capture / Compare register 5 TMG 2	GCC25	-	-	R/W	-
0xFFFFF700	Interrupt mode register 0	INTM0	R/W	R/W	-	-
0xFFFFF702	Interrupt mode register 1	INTM1	R/W	R/W	-	-
0xFFFFF704	Interrupt mode register 2	INTM2	R/W	R/W	-	-
0xFFFFF706	Interrupt mode register 3	INTM3	R/W	R/W	-	-
0xFFFFF710	Digital filter enable register 0	DFEN0	-	-	R/W	-
0xFFFFF710	Digital filter enable register 0 low byte	DFEN0L	R/W	R/W	-	-
0xFFFFF711	Digital filter enable register 0 high byte	DFEN0H	R/W	R/W	-	-
0xFFFFF712	Digital filter enable register 1	DFEN1	-	-	R/W	-
0xFFFFF712	Digital filter enable register 1 low byte	DFEN1L	R/W	R/W	-	-
0xFFFFF713	Digital filter enable register 1 high byte	DFEN1H	R/W	R/W	-	-
0xFFFFF71A	Clock monitor control register	CLMCS	R/W	R/W	-	-
0xFFFFF720	Peripheral Function Select register 0	PFSR0	R/W	R/W	-	-
0xFFFFF726	Peripheral Function Select register 3	PFSR3	R/W	R/W	-	-

Table A-2 Other special function registers (13/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF800	Protection register	PHCMD	-	R/W	-	-
0xFFFFF802	Peripheral status	PHS	R/W	R/W	-	-
0xFFFFF820	Power Save Mode	PSM	R/W	R/W	-	-
0xFFFFF822	Clock Control	CKC	-	R/W	-	-
0xFFFFF824	Clock Generator Status	CGSTAT	-	R	-	-
0xFFFFF826	Watch Dog Clock Control	WCC	-	R/W	-	-
0xFFFFF828	Processor Clock Control	PCC	-	R/W	-	-
0xFFFFF82A	Frequency Modulation Control	SCFMC	R/W	R/W	-	-
0xFFFFF82C	Frequency Control 0	SCFC0	R/W	R/W	-	-
0xFFFFF82E	Frequency Control 1	SCFC1	R/W	R/W	-	-
0xFFFFF830	SSCG Postscaler Control	SCPS	R/W	R/W	-	-
0xFFFFF832	SPCLK Control	SCC	R/W	R/W	-	-
0xFFFFF834	FOUTCLK Control	FCC	R/W	R/W	-	-
0xFFFFF836	Watch Timer Clock Control	TCC	R/W	R/W	-	-
0xFFFFF838	IIC Clock Control	ICC	R/W	R/W	-	-
0xFFFFF840	VFB flash/ROM correction address register 0	CORAD0	-	-	-	R/W
0xFFFFF840	VFB flash/ROM correction address register 0L	CORAD0L	-	-	R/W	-
0xFFFFF842	VFB flash/ROM correction address register 0H	CORAD0H	-	-	R/W	-
0xFFFFF844	VFB flash/ROM correction address register 1	CORAD1	-	-	-	R/W
0xFFFFF844	VFB flash/ROM correction address register 1L	CORAD1L	-	-	R/W	-
0xFFFFF846	VFB flash/ROM correction address register 1H	CORAD1H	-	-	R/W	-
0xFFFFF848	VFB flash/ROM correction address register 2	CORAD2	-	-	-	R/W
0xFFFFF848	VFB flash/ROM correction address register 2L	CORAD2L	-	-	R/W	-
0xFFFFF84A	VFB flash/ROM correction address register 2H	CORAD2H	-	-	R/W	-
0xFFFFF84C	VFB flash/ROM correction address register 3	CORAD3	-	-	-	R/W
0xFFFFF84C	VFB flash/ROM correction address register 3L	CORAD3L	-	-	R/W	-
0xFFFFF84E	VFB flash/ROM correction address register 3H	CORAD3H	-	-	R/W	-
0xFFFFF850	VFB flash/ROM correction address register 4	CORAD4	-	-	-	R/W
0xFFFFF850	VFB flash/ROM correction address register 4L	CORAD4L	-	-	R/W	-
0xFFFFF852	VFB flash/ROM correction address register 4H	CORAD4H	-	-	R/W	-
0xFFFFF854	VFB flash/ROM correction address register 5	CORAD5	-	-	-	R/W
0xFFFFF854	VFB flash/ROM correction address register 5L	CORAD5L	-	-	R/W	-
0xFFFFF856	VFB flash/ROM correction address register 5H	CORAD5H	-	-	R/W	-
0xFFFFF858	VFB flash/ROM correction address register 6	CORAD6	-	-	-	R/W
0xFFFFF858	VFB flash/ROM correction address register 6L	CORAD6L	-	-	R/W	-
0xFFFFF85A	VFB flash/ROM correction address register 6H	CORAD6H	-	-	R/W	-
0xFFFFF85C	VFB flash/ROM correction address register 7	CORAD7	-	-	-	R/W
0xFFFFF85C	VFB flash/ROM correction address register 7L	CORAD7L	-	-	R/W	-
0xFFFFF85E	VFB flash/ROM correction address register 7L	CORAD7H	-	-	R/W	-
0xFFFFF870	Main oscillator clock monitor mode register	CLMM	R/W	R/W	-	-
0xFFFFF878	Sub oscillator clock monitor mode register	CLMS	R/W	R/W	-	-

Table A-2 Other special function registers (14/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF880	VFB flash/ROM correction control register	CORCN	-	R/W	-	-
0xFFFFF8A0	VSB flash correction address register 0	COR2AD0	-	-	-	R/W
0xFFFFF8A0	VSB flash correction address register 0L	COR2AD0L	-	-	R/W	-
0xFFFFF8A2	VSB flash correction address register 0H	COR2AD0H	-	-	R/W	-
0xFFFFF8A4	VSB flash correction address register 1	COR2AD1	-	-	-	R/W
0xFFFFF8A4	VSB flash correction address register 1L	COR2AD1L	-	-	R/W	-
0xFFFFF8A6	VSB flash correction address register 1H	COR2AD1H	-	-	R/W	-
0xFFFFF8A8	VSB flash correction address register 2	COR2AD2	-	-	-	R/W
0xFFFFF8A8	VSB flash correction address register 2L	COR2AD2L	-	-	R/W	-
0xFFFFF8AA	VSB flash correction address register 2H	COR2AD2H	-	-	R/W	-
0xFFFFF8AC	VSB flash correction address register 3	COR2AD3	-	-	-	R/W
0xFFFFF8AC	VSB flash correction address register 3L	COR2AD3L	-	-	R/W	-
0xFFFFF8AE	VSB flash correction address register 3H	COR2AD3H	-	-	R/W	-
0xFFFFF8B0	VSB flash correction address register 4	COR2AD4	-	-	-	R/W
0xFFFFF8B0	VSB flash correction address register 4L	COR2AD4L	-	-	R/W	-
0xFFFFF8B2	VSB flash correction address register 4H	COR2AD4H	-	-	R/W	-
0xFFFFF8B4	VSB flash correction address register 5	COR2AD5	-	-	-	R/W
0xFFFFF8B4	VSB flash correction address register 5L	COR2AD5L	-	-	R/W	-
0xFFFFF8B6	VSB flash correction address register 5H	COR2AD5H	-	-	R/W	-
0xFFFFF8B8	VSB flash correction address register 6	COR2AD6	-	-	-	R/W
0xFFFFF8B8	VSB flash correction address register 6L	COR2AD6L	-	-	R/W	-
0xFFFFF8BA	VSB flash correction address register 6H	COR2AD6H	-	-	R/W	-
0xFFFFF8BC	VSB flash correction address register 7	COR2AD7	-	-	-	R/W
0xFFFFF8BC	VSB flash correction address register 7L	COR2AD7L	-	-	R/W	-
0xFFFFF8BE	VSB flash correction address register 7L	COR2AD7H	-	-	R/W	-
0xFFFFF9FC	On Chip Debug Mode register	OCDM	R/W	R/W	-	-
0xFFFFFA00	UARTA0 Control register 0	UA0CTL0	R/W	R/W	-	-
0xFFFFFA01	UARTA0 Control register 1	UA0CTL1	R/W	R/W	-	-
0xFFFFFA02	UARTA0 Control register 2	UA0CTL2	R/W	R/W	-	-
0xFFFFFA03	UARTA0 Option register	UA0OPT0	R/W	R/W	-	-
0xFFFFFA04	UARTA0 Status register	UA0STR	R/W	R/W	-	-
0xFFFFFA06	UARTA0 Reception data register	UA0RX	-	R	-	-
0xFFFFFA07	UARTA0 Transfer data register	UA0TX	R/W	R/W	-	-
0xFFFFFA10	UARTA1 Control register 0	UA1CTL0	R/W	R/W	-	-
0xFFFFFA11	UARTA1 Control register 1	UA1CTL1	R/W	R/W	-	-
0xFFFFFA12	UARTA1 Control register 2	UA1CTL2	R/W	R/W	-	-
0xFFFFFA13	UARTA1 Option register	UA1OPT0	R/W	R/W	-	-
0xFFFFFA14	UARTA1 Status register	UA1STR	R/W	R/W	-	-
0xFFFFFA16	UARTA1 Reception data register	UA1RX	-	R	-	-
0xFFFFFA17	UARTA1 Transfer data register	UA1TX	R/W	R/W	-	-
0xFFFFFB00	LCD clock control	LCDC0	R/W	R/W	-	-

Table A-2 Other special function registers (15/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFFB01	LCD display mode control	LCDM0	R/W	R/W	-	-
0xFFFFFB20	LCD RAM data	SEGREG000	R/W	R/W	-	-
0xFFFFFB20	LCD RAM data	SEGREG020	R/W	R/W	-	-
0xFFFFFB21	LCD RAM data	SEGREG001	R/W	R/W	-	-
0xFFFFFB21	LCD RAM data	SEGREG021	R/W	R/W	-	-
0xFFFFFB22	LCD RAM data	SEGREG002	R/W	R/W	-	-
0xFFFFFB22	LCD RAM data	SEGREG022	R/W	R/W	-	-
0xFFFFFB23	LCD RAM data	SEGREG003	R/W	R/W	-	-
0xFFFFFB23	LCD RAM data	SEGREG023	R/W	R/W	-	-
0xFFFFFB24	LCD RAM data	SEGREG004	R/W	R/W	-	-
0xFFFFFB24	LCD RAM data	SEGREG024	R/W	R/W	-	-
0xFFFFFB25	LCD RAM data	SEGREG005	R/W	R/W	-	-
0xFFFFFB25	LCD RAM data	SEGREG025	R/W	R/W	-	-
0xFFFFFB26	LCD RAM data	SEGREG006	R/W	R/W	-	-
0xFFFFFB26	LCD RAM data	SEGREG026	R/W	R/W	-	-
0xFFFFFB27	LCD RAM data	SEGREG007	R/W	R/W	-	-
0xFFFFFB27	LCD RAM data	SEGREG027	R/W	R/W	-	-
0xFFFFFB28	LCD RAM data	SEGREG008	R/W	R/W	-	-
0xFFFFFB28	LCD RAM data	SEGREG028	R/W	R/W	-	-
0xFFFFFB29	LCD RAM data	SEGREG009	R/W	R/W	-	-
0xFFFFFB29	LCD RAM data	SEGREG029	R/W	R/W	-	-
0xFFFFFB30	LCD RAM data	SEGREG010	R/W	R/W	-	-
0xFFFFFB30	LCD RAM data	SEGREG030	R/W	R/W	-	-
0xFFFFFB31	LCD RAM data	SEGREG011	R/W	R/W	-	-
0xFFFFFB31	LCD RAM data	SEGREG031	R/W	R/W	-	-
0xFFFFFB32	LCD RAM data	SEGREG012	R/W	R/W	-	-
0xFFFFFB33	LCD RAM data	SEGREG013	R/W	R/W	-	-
0xFFFFFB34	LCD RAM data	SEGREG014	R/W	R/W	-	-
0xFFFFFB35	LCD RAM data	SEGREG015	R/W	R/W	-	-
0xFFFFFB36	LCD RAM data	SEGREG016	R/W	R/W	-	-
0xFFFFFB37	LCD RAM data	SEGREG017	R/W	R/W	-	-
0xFFFFFB38	LCD RAM data	SEGREG018	R/W	R/W	-	-
0xFFFFFB39	LCD RAM data	SEGREG019	R/W	R/W	-	-
0xFFFFFB40	LCD RAM data	SEGREG032	R/W	R/W	-	-
0xFFFFFB41	LCD RAM data	SEGREG033	R/W	R/W	-	-
0xFFFFFB42	LCD RAM data	SEGREG034	R/W	R/W	-	-
0xFFFFFB43	LCD RAM data	SEGREG035	R/W	R/W	-	-
0xFFFFFB44	LCD RAM data	SEGREG036	R/W	R/W	-	-
0xFFFFFB45	LCD RAM data	SEGREG037	R/W	R/W	-	-
0xFFFFFB46	LCD RAM data	SEGREG038	R/W	R/W	-	-
0xFFFFFB47	LCD RAM data	SEGREG039	R/W	R/W	-	-

Table A-2 Other special function registers (16/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFFB60	LCD Bus Interface Control	LBCTL0	R/W	R/W	-	-
0xFFFFFB61	LCD Bus Interface Cycle Time	LBCYC0	R/W	R/W	-	-
0xFFFFFB62	LCD Bus Interface Wait States	LBWST0	R/W	R/W	-	-
0xFFFFFB70	LCD Bus Interface Data	LBDATA0W	-	-	-	R/W
0xFFFFFB70	LCD Bus Interface Data	LBDATA0	-	-	R/W	-
0xFFFFFB70	LCD Bus Interface Data	LBDATA0L	-	R/W	-	-
0xFFFFFB74	LCD Bus Interface Data	LBDATAR0L	-	R	-	-
0xFFFFFB74	LCD Bus Interface Data	LBDATAR0	-	-	R	-
0xFFFFFB74	LCD Bus Interface Data	LBDATAR0W	-	-	-	R
0xFFFFFCA0	Self-programming enable control register	SELFEN	R/W	R/W	-	-
0xFFFFFCA2	Stand-by control register	STBCTL	R/W	R/W		
0xFFFFFCA8	Self-programming enable protection register	SELFENP	-	W	-	-
0xFFFFFCAA	Stand-by control protection register	STBCTLP	-	W		
0xFFFFFCB0	CLMM write protection register	PRCMDCMM	-	W	-	-
0xFFFFFCB2	CLMS write protection register	PRCMDCMS	-	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 received data register low byte	CB0RX0L	-	R	-	-
0xFFFFFD04	CSIB0 received data register	CB0RX0	-	-	R	-
0xFFFFFD06	CSIB0 send data register	CB0TX0	-	-	R/W	-
0xFFFFFD06	CSIB0 send data register low byte	CB0TX0L	-	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 received data register low byte	CB1RX0L	-	R	-	-
0xFFFFFD14	CSIB1 received data register	CB1RX0	-	-	R	-
0xFFFFFD16	CSIB1 send data register	CB1TX0	-	-	R/W	-
0xFFFFFD16	CSIB1 send data register low byte	CB1TX0L	-	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 received data register low byte	CB2RX0L	-	R	-	-
0xFFFFFD24	CSIB2 received data register	CB2RX0	-	-	R	-
0xFFFFFD26	CSIB2 send data register	CB2TX0	-	-	R/W	-
0xFFFFFD26	CSIB2 send data register low byte	CB2TX0L	-	R/W	-	-
0xFFFFFD80	IIC0 shift register	IIC0	-	R/W	-	-
0xFFFFFD82	IIC0 control register	IICC0	R/W	R/W	-	-

Table A-2 Other special function registers (17/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFFD83	IIC0 Slave address register	SVA0	-	R/W	-	-
0xFFFFFD84	IIC0 combined IICCL0 and IICX0 register	IICCL0IICX0	-	-	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	-	-
0xFFFFFD87	IIC0 state register (for emulation only)	IICSE0	R	R	-	-
0xFFFFFD8A	IIC0 flag register	IICF0	R/W	R/W	-	-
0xFFFFFD90	IIC1 shift register	IIC1	-	R/W	-	-
0xFFFFFD92	IIC1 control register	IICC1	R/W	R/W	-	-
0xFFFFFD93	IIC1 Slave address register	SVA1	-	R/W	-	-
0xFFFFFD94	IIC1 combined IICCL0 and IICX0 register	IICCL1IICX1	-	-	R/W	-
0xFFFFFD94	IIC1 clock selection register	IICCL1	R/W	R/W	-	-
0xFFFFFD95	IIC1 function expansion register	IICX1	R/W	R/W	-	-
0xFFFFFD96	IIC1 state register	IICS1	R	R	-	-
0xFFFFFD97	IIC1 state register (for emulation only)	IICSE1	R	R	-	-
0xFFFFFD9A	IIC1 flag register	IICF1	R/W	R/W	-	-
0xFFFFFDA0	Clock selection register odd prescaler 0	OCKS0	R/W	R/W	-	-
0xFFFFFDB0	Clock selection register odd prescaler 1	OCKS1	R/W	R/W	-	-
0xFFFFFDC0	Pre-scalar mode register	PRSM0	R/W	R/W	-	-
0xFFFFFDC1	Pre-scalar compare register	PRSCM0	R/W	R/W	-	-
0xFFFFFDE0	Pre-scalar mode register	PRSM1	R/W	R/W	-	-
0xFFFFFDE1	Pre-scalar compare register	PRSCM1	R/W	R/W	-	-
0xFFFFFDF0	Pre-scalar mode register	PRSM2	R/W	R/W	-	-
0xFFFFFDF1	Pre-scalar compare register	PRSCM2	R/W	R/W	-	-
0xFFFFFE00	DMA trigger source select register 0	DTFR0	R/W	R/W	-	-
0xFFFFFE02	DMA trigger source select register 1	DTFR1	R/W	R/W	-	-
0xFFFFFE04	DMA trigger source select register 2	DTFR2	R/W	R/W	-	-
0xFFFFFE06	DMA trigger source select register 3	DTFR3	R/W	R/W	-	-
0xFFFFFF10	Voltage Comparator 0 Control	VCCTL0	R/W	R/W	-	-
0xFFFFFF00	Read delay control register	RDDLY	R/W	R/W	-	-
0xFFFFFF12	Voltage Comparator 0 Status	VCSTR0	R/W	R/W	-	-
0xFFFFFF14	Voltage Comparator 1 Control	VCCTL1	R/W	R/W	-	-
0xFFFFFF16	Voltage Comparator 1 Status	VCSTR1	R/W	R/W	-	-
0xFFFFFF20	Reset Source Flag register	RESSTAT	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.

In 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 - Peripheral function wait control register“ on page 234* 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 P

Register **TPnCCR0, TPnCCR1**

Access R

Formula

$$T_a = \left\{ SUWL + VSWL + 3 + ru\left[\frac{f_{VBCLK}}{(2 + VSWL) \cdot f_{PCLK0}} + 1 \right] \cdot (2 + VSWL) \right\} \cdot \frac{1}{f_{VBCLK}}$$

Access W

Formula

$$T_a = \left\{ SUWL + VSWL + 3 + ru\left[\frac{5 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{PCLK0}} + 1 \right] \cdot (2 + VSWL) \right\} \cdot \frac{1}{f_{VBCLK}}$$

Register **TPnCNT**

Access R

Formula

$$T_a = \left\{ SUWL + VSWL + 3 + ru\left[\frac{f_{VBCLK}}{(2 + VSWL) \cdot f_{PCLK0}} + 1 \right] \cdot (2 + VSWL) \right\} \cdot \frac{1}{f_{VBCLK}}$$

Access W

Formula

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

Register **all other**

Access R/W (no write access during timer operation)

Formula

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

B.2 Timer Z

Register **TZnCNT0**

Access R

Formular $T_a = (SUWL + 3 \cdot VSWL + 6) \cdot \frac{1}{f_{VBCLK}} + \frac{2}{f_{PCLK2}}$

Register **TZnCNT1**

Access R

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Register **TZnR**

Access R

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Access W

Formular $T_a = (SUWL + 3 \cdot VSWL + 6) \cdot \frac{1}{f_{VBCLK}} + \frac{2}{f_{PCLK2}}$

Register **TZnCTL**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.3 Timer G

Register **TMGn0, TMGn1**

Access R

Formular $T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{f_{VBCLK}}{(2 + VSWL) \cdot f_{SPCLK0}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$

Access W (no write access during timer operation)

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Register **GCCn[5:0]**

Access R

Formular $T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{f_{VBCLK}}{(2 + VSWL) \cdot f_{SPCLK0}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$

Access W (for GCCn0 and GCCn5 no write access during timer operation)

Formular

- for multiple write within 7 SPCLK0 periods

$$T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{f_{VBCLK}}{(2 + VSWL) \cdot f_{SPCLK0}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$$

- for single write within 7 SPCLK0 periods

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

Register **all other**

Access R/W (no write access during timer operation)

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.4 Watch Timer

Register **WTnCNT1**

Access R

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Register **WTnR**

Access R

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Access W

Formular $T_a = (SUWL + 3 \cdot VSWL + 7) \cdot \frac{1}{f_{VBCLK}}$

Register **CR00**

Access Read-Modify-Write

Formular $T_a = (SUWL + 3 \cdot VSWL + 7) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.5 Watch Calibration Timer

Register **CR01**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Access Read-Modify-Write

Formular $T_a = (SUWL + 3 \cdot VSWL + 7) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.6 Watchdog Timer

Register **all**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.7 Asynchronous Serial Interface (UARTA)

Register **all**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.8 Clocked Serial Interface (CSIB)

Register **all**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.9 I^{2}C Bus

Register **IICSn**

Access R

Formular $T_a = (SUWL + 3 \cdot VSWL + 7) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.10 CAN Controller

Register **CnMDATA[7:0]m**

Access R

Formular $T_a = \left\{ SUWL + VSWL + 3 + ru\left[\frac{4 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{PCLK0}} \right] \cdot (2 + VSWL) \right\} \cdot \frac{1}{f_{VBCLK}}$

Access 8-bit Write

Formular $T_a = \left\{ SUWL + VSWL + 3 + ru\left[\frac{5 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{PCLK0}} \right] \cdot (2 + VSWL) \right\} \cdot \frac{1}{f_{VBCLK}}$

Access 16-bit Write

Formular $T_a = \left\{ SUWL + VSWL + 3 + ru\left[\frac{3 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{PCLK0}} \right] \cdot (2 + VSWL) \right\} \cdot \frac{1}{f_{VBCLK}}$

Register **CnRGPT, CnTGPT, CnLIPT, CnLOPT**

Access R

Formular $T_a = \left\{ SUWL + VSWL + 3 + ru\left[\frac{4 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{PCLK0}} \right] \cdot (2 + VSWL) \right\} \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formular $T_a = \left\{ SUWL + VSWL + 3 + ru\left[\frac{2 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{PCLK0}} \right] \cdot (2 + VSWL) \right\} \cdot \frac{1}{f_{VBCLK}}$

B.11 A/D Converter

Register **ADAM0[2:0], ADACR0n**

Access R

Formular $T_a = \left\{ SUWL + VSWL + 3 + ru\left[\frac{2 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{SPCLK0}} + 1 \right] \cdot (2 + VSWL) \right\} \cdot \frac{1}{f_{VBCLK}}$

Access W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.12 Stepper Motor Controller/Driver

Register **MCNTCn[1:0], MCMPCnk**

Access R

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Access W

Formular $T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{2 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{SPCLK1}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.13 LCD Controller/Driver

Register **all**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.14 LCD Bus Interface

Register **all**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.15 Sound Generator

Register **SG0FL, SG0FH, SG0PWM**

Access R

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Access W

Formular $T_a = (SUWL + 3 \cdot VSWL + 6) \cdot \frac{1}{f_{VBCLK}} + \frac{2}{f_{PCLK0}}$

Register **all other**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.16 Clock Generator

Register **CGSTAT**

Access R

Formular $T_a = (SUWL + 3 \cdot VSWL + 7) \cdot \frac{1}{f_{VBCLK}}$

Access W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.17 All other Registers

Register **all**

Access R/W

Formular $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Revision History

This revision list shows all functional changes of this document U17566EE1V2UM00 compared to the 2nd edition of previous manual version 1.0 U17566EE1V1UM00 (date published 16/05/06).

Chapter	Page	Description
4	26	3 Timer G specified for all devices
4	34	ordering information added
5	71	section concerning external memory interface of µPD70F3427 added
5	76	TOP01 to TOP31 Timer P outputs also available on ports P34 to P37
6	121	size and address of VSB RAM corrected
6	126	section concerning instruction/data access times to various memories added
7	129	Ring oscillator clock corrected to 240 KHz
9	237	caution added concerning usage of devices used for debugging in mass production
9	239	maximum UART transfer rate for flash programming corrected to 153.600 bps
9	240	maximum CSI transfer rate for flash programming corrected to 2.5 MHz
9	241	PG-FP4 pin functions of FLMD0/FLMD1 corrected
10	257	"NPB access timing" section added
10	267	caution added to CSCn settings for µPD70F3426
10	270	caution added to BEC settings for µPD70F3426
10	272	local bus size configuration register LBS added for µPD70F3427
14	368	TPnCTL1 setting for external event count mode corrected: TPnCTL1.TPnEEE = 1
19	537	UART baud rate settings for 300 bps to 31250 bps corrected
22	768	operating precaution concerning CAN wake-up from sleep mode added
23	776	Specification of ADA0M2 settings extended: ADA0M2.ADA0TDM[1:0] = 00: no trigger
23	785	correction: ADA0PFM.ADA0PFC = 1 for ADCR0H0 < ADA0PFT comparison
25	812	LCD-C/D register and bit names changed (LCDC to LCDC0, LCDM to LCDM0, SEGREGk to SEGREG0k)
25	815	LCDM.VAON bit deleted
25	818	LCD driver edge enhancement section deleted
31	877	number of software breakpoints during on-chip debugging specified

Index