Preliminary User’s Manual

V850E/CA1^{TM} ATOMIC

32-/16-bit Single-Chip Microcontroller

Hardware

µPD703123,
µPD70F3123

Document No. U14913EE1V0UM00
Date Published October 2001

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

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

Note:

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 V_{DD} 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

Note:

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.

The related documents in this publication may include preliminary versions. However, preliminary versions are not marked as such.

The export of this product from Japan is regulated by the Japanese government. To export this product may be prohibited without governmental license, the need for which must be judged by th customer. The export or re-export of this product from a country other than Japan may also be prohibited without a license from that country. Please call an NEC sales representative.

The information in this document is current as of 20.09.2001. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC's data sheets or data books, etc., for the most up-to-date specifications of NEC semiconductor 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. NEC assumes no responsibility for any errors that may appear in this document. NEC 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 semiconductor products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC 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 assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information. While NEC endeavours to enhance the quality, reliability and safety of NEC semiconductor 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 semiconductor products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features. NEC semiconductor products are classified into the following three quality grades: "Standard", "Special" and "Specific". The "Specific" quality grade applies only to semiconductor products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of a semiconductor product depend on its quality grade, as indicated below. Customers must check the quality grade of each semiconductor 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": Aircrafts, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems or medical equipment for life support, etc.

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

Notes: (1) "NEC" as used in this statement means NEC Corporation and also includes its majority-owned subsidiaries.
(2) "NEC semiconductor products" means any semiconductor product developed or manufactured by or for NEC (as defined above).

M5 2000.03

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 (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.l.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Vélizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 91-504-2787
Fax: 91-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

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.
United Square, Singapore 1130
Tel: 65-253-8311
Fax: 65-250-3583

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

NEC do Brasil S.A.
Electron Devices Division
Rodovia Presidente Dutra, Km 214
07210-902-Guarulhos-SP Brasil
Tel: 55-11-6465-6810
Fax: 55-11-6465-6829

J99.1

Preface

Readers This manual is intended for users who want to understand the functions of the V850E/CA1 (nickname Atomic) (µPD70F3123, µPD703123).

Purpose This manual presents the hardware manual of V850E/CA1.

Organization This system specification describes the following sections:

- Pin function
- CPU function
- Internal peripheral function
- Flash memory

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 : $\overline{xxx}$ (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 : Explanation of (Note) in the text

Caution : Item deserving extra attention

Remark : Supplementary explanation to the text

Numeric notation : Binary . . . xxxx or xxxB
Decimal . . . xxxx
Hexadecimal . . . xxxxH or 0x xxxx

Prefixes representing powers of 2 (address space, memory capacity)

K (kilo): $2^{\mathbf{10}}$ = 1024
M (mega): $2^{\mathbf{20}}$ = $1024^{\mathbf{2}}$ = 1,048,576
G (giga): $2^{\mathbf{30}}$ = $1024^{\mathbf{3}}$ = 1,073.741,824

[MEMO]

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 General

The V850E/CA1 / ATOMIC single chip microcontroller, is a member of NEC's V850 32-bit RISC family, which match the performance gains attainable with RISC-based controllers to the needs of embedded control applications. The V850 CPU offers easy pipeline handling and programming, resulting in compact code size comparable to 16-bit CISC CPUs.

The V850E/CA1 / ATOMIC offers an excellent combination of general purpose peripheral functions, like serial communication interfaces (UART, clocked SI), display drivers and measurement inputs (A/D converter), with dedicated CAN network support. To support more than one CAN network, three CAN interfaces and a CAN gateway are implemented on chip. The CAN gateway - realised in hardware - acts as a configurable message handler and supports the increasing demand of messages without unnecessarily decreasing the CPU performance.
The device offers power-saving modes to manage the power consumption effectively under varying conditions.

(1) V850E CPU

The V850E CPU supports the RISC instruction set, and through the use of basic instructions that can each be executed in 1-clock period and an optimized pipeline, achieves marked improvements in instruction execution speed. In addition, in order to make it ideal for use in digital servo control, a 32-bit hardware multiplier enables this CPU to support multiply instructions, saturated multiply instructions, bit operation instructions, etc.
Also, through 2-byte basic instructions and instructions compatible with high level languages, etc., object code efficiency in a C compiler is increased, and program size can be made more compact.
Further, since the on-chip interrupt controller provides high speed interrupt response, including processing, this device is suited for high level real time control fields.

(2) External memory interface function

The V850E/CA1 contains a non multiplexed external bus interface, including an address bus (24 bits) and data bus (selectable 8 bits or 16 bits). SRAM and ROM can be connected as well as page ROM memories.
The DMA controller allows, data transfers between internal RAM and peripheral I/O. This reduces the CPU load.

(3) On-chip flash memory (µPD70F3123)

The on-chip flash memory version (µPD70F3123) has on-chip an high speed flash memory, which is able to fetch one instruction one clock cycle. It is possible to program the user application direct in the target application, on which the V850E/CA1 is mounted. In such case system development time can be reduced and system maintainability after shipping can be markedly improved.

(4) A full range of development environment products

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

1.2 Device Features

- CPU

- Core:	V850E
- Number of instructions:	81
- Min. instruction execution time:	50 ns (@ f_{CPU} = 20 MHz)
- General registers:	32 bits x 32

- Instruction set:
  - V850E (compatible with V850 plus additional powerful instructions for reducing code and increasing execution speed)
  - Signed multiplication (16 bits x 16 bits → 32 bits or 32 bits x 32 bits → 64 bits): 1 to 2 clocks
  - Saturated operation instructions (with overflow/underflow detection function)
  - 32-bit shift instructions: 1 clock
  - Bit manipulation instructions
  - Load/store instructions with long/short format
  - Signed load instructions

- Internal memory

Part Number	Internal ROM	Internal RAM	Full-CAN RAM	LCD display RAM
µPD70F3123	Flash memory: 256 Kbytes	10 Kbytes	2 Kbytes (64 message buffers)	40 x 4 bits
µPD703123	Mask ROM 256 Kbytes	10 Kbytes	2 Kbytes (64 message buffers)	40 x 4 bits

- Flash selfprogramming support

- Clock Generator

- Internal PLL	4 fold PLL
- Frequency range:	up to 20 MHz
- Crystal frequency range:	4 MHz ≤ $f_{CRYSTAL}$ ≤ 5 MHz

• Built-in power saving modes:	WATCH, HALT, STOP
• Power supply voltage range	4.5 V ≤ V_{DD5} ≤ 5.5 V
• Temperature range:	Ta = - 40 to + 85°C

- Bus control unit:
  - Address/data separated bus (24-bit address/ 16/8-bit data bus)
  - 16/8-bit bus sizing function

• DMA control unit:	4 channels
• I/O lines:	90
• A/D Converter:	10-bit resolution; 12 channels

- Serial Interfaces

- 3-wire mode:	2 channels
- UART mode:	3 channels

- Full-CAN Interface:
  - Full CAN 3 channel

- CAN Bridge for Gateway applications

- Timers
  - 16/32-bit multi purpose timer/event counter: 3 channel
  - 16-bit OS timer 2 channel
  - Watch timer: 1 channel
  - Watchdog timer: 1 channel

- LCD controller/driver
  - Segment signal output: max. 40
  - Common signal output: max. 4
  - Bias: 1/1, 1/2, 1/3
  - Operating supply voltage range: [Note] $4.5\ V \leq V_{LCD} \leq V_{DD}$

- Interrupts $\leq$ 64 vectored interrupts

- Package 144 QFP, 0.5 mm pin-pitch

Note: For an operating voltage below the given minimum value the quality of the displayed segments will decrease.

1.3 Application Fields

The V850E/CA1 / ATOMIC is ideally suited for automotive applications, like dashboard or central gateway applications. It is also an excellent choice for other applications where a combination of sophisticated peripheral functions and CAN network support is required.

1.4 Ordering Information

Part Number	Package	Program memory	Data memory	
		ROM / Flash [bytes]	RAM [bytes]	FCAN RAM [bytes]
µPD703123GJ-UEN	144-pin LQFP (0.5 mm pitch)	Mask ROM 256 K	10 K	2 K (= 64 message buffers)
µPD70F3123GJ-UEN	144-pin LQFP (0.5 mm pitch)	Flash Memory 256 K	10 K	2 K (= 64 message buffers)

1.5 Pin Configuration (Top View)

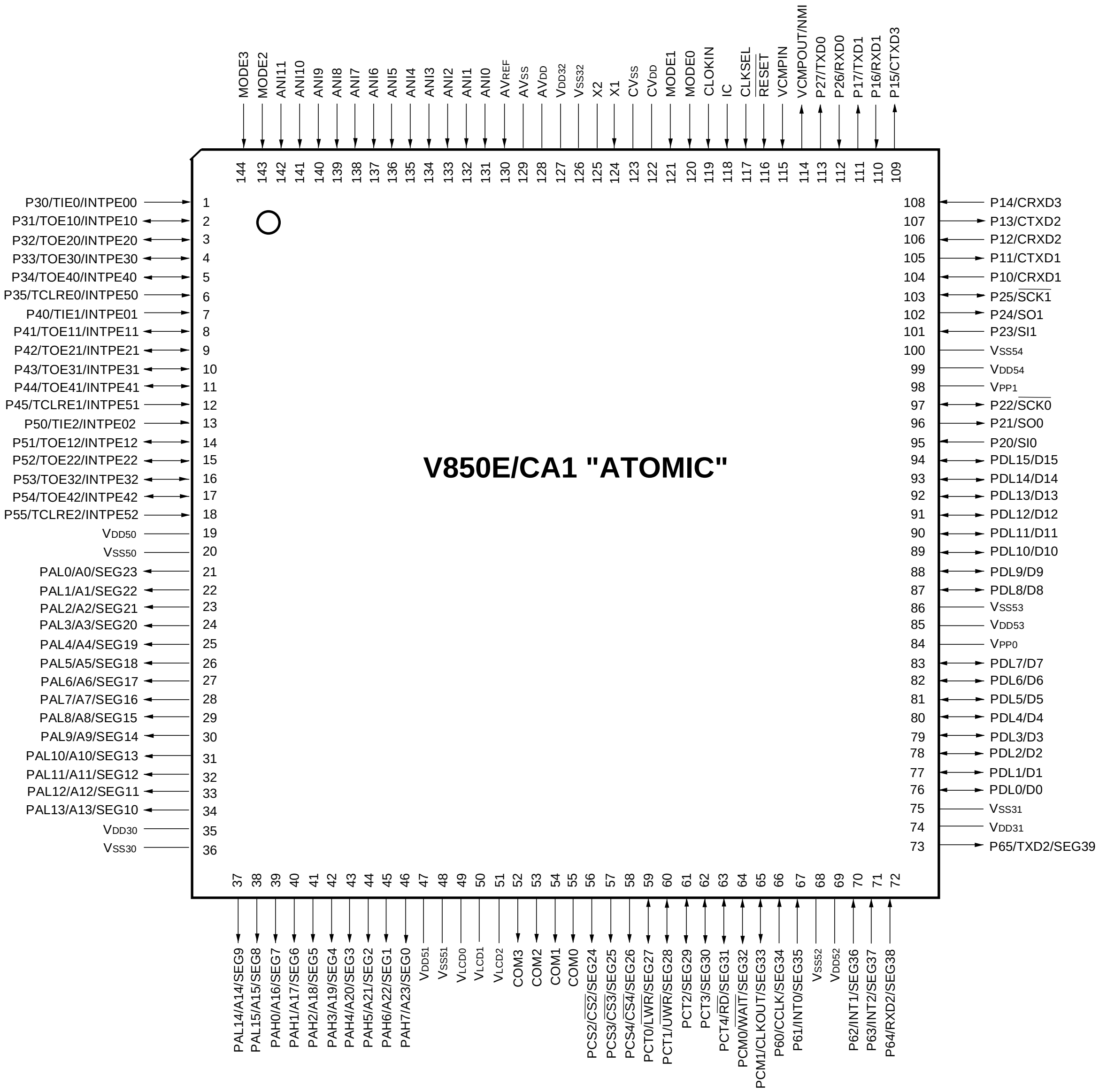


Figure 1-1: Pin configuration of the µPD70(F)3123 microcontroller

Pin Identification

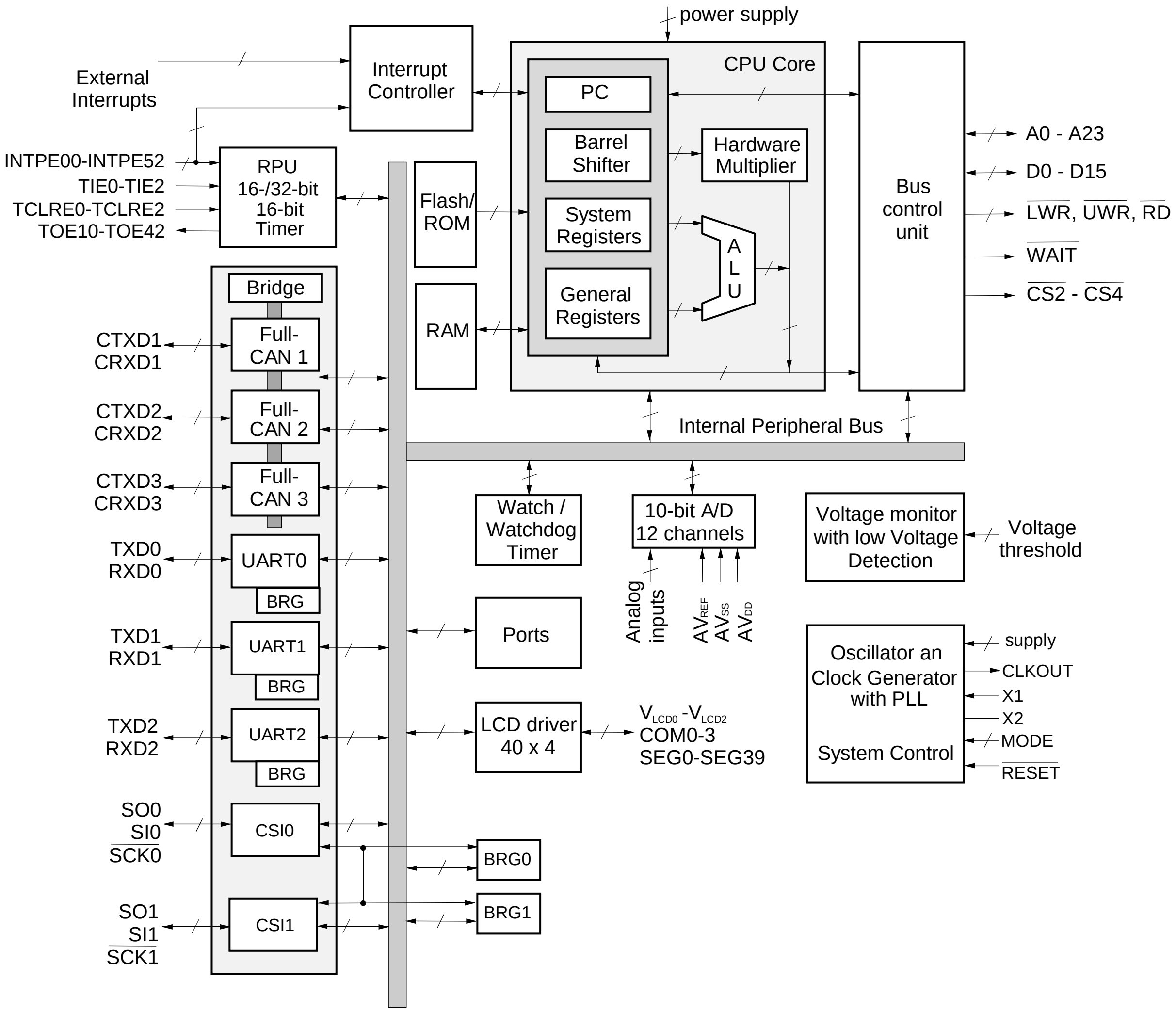
A0 to A23	:	Address Bus	P60 to P65	:	Port 6
D0 to D15	:	Data Bus	PAL0 to PAL15		Port AL
ANI00 to ANI11	:	Analog Input	PAH0 to PAH7	:	Port AH
AV_{DD}	:	Analog Power Supply	PCM0, PCM1	:	Port CM
AV_{REF}	:	Analog Reference Voltage	PCS2 to PCS4	:	Port CS
AV_{SS}	:	Analog Ground	PCT0 to PCT4	:	Port CT
CCLK		CAN clock input	PDL0 to PDL15	:	Port DL
CLKSEL	:	Clock Generator Operating Mode Select	$\overline{RD}$	:	Read
CLOCKIN		External system clock input	IC		Internal connection
CLKOUT	:	Clock Output	$\overline{RESET}$	:	Reset
COM0 to COM3		LCD common line	RXD0 to RXD2	:	Receive Data Input
CRXD1 to CRXD3	:	CAN Receive Line Input	$\overline{SCK0}$, $\overline{SCK1}$	:	Serial Clock
$\overline{CS2}$ to $\overline{CS4}$	:	Chip Select	SEG0 to SEG39		LCD segment pins
CTXD1 to CTXD3	:	CAN Transmit Line Output	SI0, SI1	:	Serial Input
CV_{DD}	:	Clock Generator Power Supply	SO0, SO1	:	Serial Output
CV_{SS}	:	Clock Generator Ground	TCLRE0, TCLRE1, TCLRE2	:	Timer Clear Input
GND_{30} to GND_{33}		Ground for 3 V Power Supply	TIE0 to TIE2	:	Timer Input
GND_{50} to GND_{54}		Ground for 5 V Power Supply	TOEn0, TOEn2	:	Timer Output
INT0 to INT2		External interrupt request	TXD0 to TXD2	:	Transmit Data Output
INTPEn0 to INTPEn2	:	Interrupt Request from Peripherals	VCMPIN		Voltage Comparator Input
MODE0 to MODE3	:	Mode Inputs	VCMPOUT		Cottage Comparator Feedback Output
NMI	:	Non-Maskable Interrupt Request	V_{DD30} to V_{DD33}	:	3 V Power Supply
P10 to P17	:	Port 1	V_{DD50} to V_{DD54}	:	5 V Power Supply
P20 to P27	:	Port 2	V_{LCD0} to V_{LCD2}		LCD driver power supply
P30 to P35	:	Port 3	V_{PP0}, V_{PP1}	:	Programming Power Supply
P40 to P45	:	Port 4	$\overline{WAIT}$	:	Wait
P50 to P55	:	Port 5	$\overline{LWR}$, $\overline{UWR}$		Write Enable
			X1, X2	:	Crystal

Remark: n = 0 to 5

1.6 Configuration of Function Block

1.6.1 Block Diagram of µPD70(F)3123

Figure 1-2: Block diagram of the µPD70(F)3123 microcontroller



1.6.2 On-chip units

(1) CPU

The CPU uses five-stage pipeline control to enable single-clock execution of address calculations, arithmetic logic operations, data transfers, and almost all other instruction processing.
Other dedicated on-chip hardware, such as the multiplier (16 bits x 16 bits → 32 bits or 32 bits x 32 bits → 64 bits) and the barrel shifter (32 bits), help accelerate processing of complex instructions.

(2) Bus control unit (BCU)

BCU starts a required external bus cycle based on the physical address obtained by the CPU.
When an instruction is fetched from external memory area and the CPU does not send a bus cycle start request, the BCU generates a prefetch address and prefetches the instruction code. The prefetched instruction code is stored in an instruction queue in the CPU.
The BCU provides a page ROM controller (ROMC) and a DMA controller (DMAC).

(a) Page ROM controller (ROMC)

This controller supports accessing ROM that includes the page access function.
It performs address comparisons with the immediately preceding bus cycle and executes wait control for normal access (off page)/page access (on page). It can handle page widths of 8 to 128 bytes.

(b) DMA controller (DMAC)

Instead of the CPU, this controller controls data transfer between memory and I/O.
There is one address mode: 2-cycle transfer and there are three bus modes: single transfer, single step transfer, and block transfer.

(3) ROM

The µPD703123 has on-chip mask ROM (256 Kbytes) and the µPD70F3123 on-chip flash memory (256 Kbytes).
During instruction fetch, ROM/flash memory can be accessed from the CPU in 1-clock cycles.
If the single chip mode 0 or flash memory programming mode is set, memory mapping is done from address 00000000H.

(4) RAM

RAM are mapped from address FFFFC000H.
During instruction fetch, data can be accessed from the CPU in 1-clock cycles.

(5) Interrupt controller (INTC)

This controller handles hardware interrupt requests (NMI, INT0, INT1, INT2) from on-chip peripheral I/O and external hardware. Eight levels of interrupt priorities can be specified for these interrupt requests, and multiple-interrupt servicing control can be performed for interrupt sources.

(6) Clock generator (CG)

This clock generator supplies frequencies which are 4 times the input clock (f_{XX}) (used by the internal PLL) and 1/2 the input clock (when an on-chip PLL is not used) as an internal system clock (f_{CPU}). As the input clock, an external oscillator is connected to pins X1 and X2 (only when an internal PLL synthesizer is used) or an external clock is input from pin X1.

(7) Real-time pulse unit (RPU)

This unit has 3 channels of 16/32-bit multi purpose timer/event counter and 2 channels of 16-bit interval timer built in, and it is possible to measure pulse widths or frequency and to output a programmable pulse.

(8) Serial interface (SIO)

A 3-channel asynchronous serial interface (UART), 2-channel clocked serial interface (CSI), and 3-channel FCAN are provided as serial interface.

UART transfers data by using the TXDn and RXDn pins. (n = 0 - 2)
CSI transfers data by using the SOn, SIn, and SCKn pins. (n = 0, 1)
FCAN performs data transfer using CTXDn and CRXDn pins. (n = 1 - 3)

(9) A/D converter (ADC)

One high-resolution 10-bit A/D converter, it includes 12 analog input pins. Conversion uses the successive approximation method.

(10) LCD-controller

The segment LCD 40 x 4 controller/driver allows direct connection of LCD modules.
Segment signal output: Max. 40
Common signal output: Max. 4
Bias: static, 1/2, 1/3 bias switching possible
Segment lines not used for a LCD can be configured to function as port pins.

(11) Ports

As shown below, the following ports have general port functions and control pin functions.

Port	Port Function	Control Function
Port 1	8-bit input/output	Serial interface input/output
Port 2	8-bit input/output	Serial interface input/output
Port 3	6-bit input/output	Real-time pulse unit input/output, external interrupt input, PWM output
Port 4	6-bit input/output	Real-time pulse unit input/output, external interrupt input, PWM output
Port 5	6-bit input/output	Real-time pulse unit input/output, external interrupt input, PWM output
Port 6	6-bit input/output	Serial interface input/output, external interrupt input
Port AL	16-bit input/output	External address bus, LCD-controller driver segment line
Port AH	8-bit input/output	External address bus, LCD-controller driver segment line
Port DL	16-bit input/output	External data bus
Port CS	3-bit input/output	External bus interface control signal output, LCD-controller driver segment line
Port CT	5-bit input/output	External bus interface control signal output, LCD-controller driver segment line
Port CM	2-bit input/output	Wait insertion signal input, internal system clock output, LCD-controller driver segment line

Chapter 2 Pin Functions

2.1 List of Pin Functions

The names and functions of this product’s pins are listed below. These pins can be divided into port pins and non-port pins according to their functions.

(1) Port pins

Port	I/O	Function	Alternate
P10	I/O	Port 1	CRXD1
P11		8-bit input/output port	CTXD1
P12			CRXD2
P13			CTXD2
P14			CRXD3
P15			CTXD3
P16			RXD1
P17			TXD1
P20	I/O	Port 2	SI0
P21		8-bit input/output port	SO0
P22			$\overline{\text{SCK0}}$
P23			SI1
P24			SO1
P25			$\overline{\text{SCK1}}$
P26			RXD0
P27			TXD0
P30	I/O	Port 3	TIE0/INTPE00
P31		6-bit input/output port	TOE10/INTPE10
P32			TOE20/INTPE20
P33			TOE30/INTPE30
P34			TOE40/INTPE40
P35			TCLRE0/INTPE50
P40	I/O	Port 4	TIE1/INTPE01
P41		6-bit input/output port	TOE11/INTPE11
P42			TOE21/INTPE21
P43			TOE31/INTPE31
P44			TOE41/INTPE41
P45			TCLRE1/INTPE51
P50	I/O	Port 5	TIE2/INTPE02
P51		6-bit input/output port	TOE12/INTPE12
P52			TOE22/INTPE22
P53			TOE32/INTPE32
P54			TOE42/INTPE42
P55			TCLRE2/INTPE52

Port	I/O	Function	Alternate
P60	I/O	Port 6 6-bit input/output port	CCLK/SEG34
P61			INT0/SEG35
P62			INT1/SEG36
P63			INT2/SEG37
P64			RXD2/SEG38
P65			TXD2/SEG39
PAL0-PAL15	I/O	Port AL 16-bit input/output port	A0-A15 / SEG23-SEG8
PAH0-PAH7	I/O	Port AH 8-bit input/output port	A16-A23 / SEG7-SEG0
PDL0-PDL15	I/O	Port DL 16-bit input/output port	D0-D15
PCS2-PCS4	I/O	Port CS 3-bit input/output port	$\overline{\text{CS2}}$ - $\overline{\text{CS4}}$ / SEG24-SEG26
PCT0	I/O	Port CT	$\overline{\text{LWR}}$ / SEG27
PCT1		5-bit input/output port	$\overline{\text{UWR}}$ / SEG28
PCT2			SEG29
PCT3			SEG30
PCT4			$\overline{\text{RD}}$/ SEG31
PCM0	I/O	Port CM	$\overline{\text{WAIT}}$ / SEG32
PCM1		2-bit input/output port	CLKOUT / SEG33

(2) Non-port pins

Pin Name	I/O	Function	Alternate
V_{DD50}-V_{DD54}	–	Power supply 5 V	-
V_{SS50}-V_{SS54}	–	GND potential	-
V_{DD30}-$V_{DD33}$$^{\textbf{Note 1}}$	–	Connection for external capacities	-
V_{SS30}-V_{SS33}	–	GND potential	-
$CV_{DD}$$^{\textbf{Note 2}}$	–	Connection for external capacities to stabilize clock oscillator power supply	-
CV_{SS}			-
X1	input	System clock oscillator connection pins.	-
X2	–		-
CLOCKIN	input	External system clock input	-
V_{PP0}, V_{PP1}	–	Flash memory programming voltage	-
IC	–	Internal connection pin (connect directly to V_{SS})	-
MODE0-MODE1	input	Selects operating mode (internal ROM)	-
MODE2-MODE3	input	Have to be fixed to GND	-
$\overline{\text{RESET}}$	input	System reset input	-
CLKOUT	output	Internal CPU system clock output	PCM1/SEG33
CLKSEL	input	Clock generator operation mode (connect to V_{SS} for X1, X2 input)	-
AV_{DD}	–	Power supply for A/D converter	-
AV_{SS}	–	Ground potential for A/D converter	-
AV_{REF}	input	reference voltage input for A/D converter	-
NMI	input	non maskable interrupt input	VCMPOUT
VCMPOUT	output	voltage comparator feedback output	NMI
VCMPIN	input	voltage comparator compare input	-
ANI0-ANI11	input	analog input to A/D converter	-
SI0	input	serial receive data input to CSI0-CSI1	P20
SI1			P23
SO0	output	serial transmit data output from CSI0-CSI1	P21
SO1			P24
$\overline{\text{SCK0}}$	I/O	serial clock I/O from/to CSI0-CSI1	P22
$\overline{\text{SCK1}}$			P25
RXD0	input	serial receive data input to UART0-UART2	P26
RXD1			P16
RXD2			P64/SEG38
TXD0	output	serial transmit data output from UART0-UART2	P27
TXD1			P17
TXD2			P65/SEG39
CRXD1	input	serial receive data input to FCAN1-FCAN3	P10
CRXD2			P12
CRXD3			P14
CTXD1	output	serial transmit data output from FCAN1-FCAN3	P11
CTXD2			P13
CTXD3			P15
CCLK	input	CAN clock input	P60/SEG34
D0-D15	I/O	data bus of external bus	PDL0-PDL15

Pin Name	I/O	Function	Alternate
A0-A7	output	address bus of external bus	PAL0-PAL7/SEG23-SEG16
A8-A15	I/O		PAL8-PAL15/SEG15-SEG8
A16-A23			PAH0-PAH7/SEG7-SEG0
$\overline{\text{LWR}}$		write strobe lower byte (bit 0-7)	PCT0/SEG27
$\overline{\text{UWR}}$		write strobe upper byte (bit 8-15)	PCT1/SEG28
$\overline{\text{RD}}$		read strobe for external bus	PCT4/SEG31
$\overline{\text{WAIT}}$	input	control signal input for external bus	PCM0/SEG32
$\overline{\text{CS2}}$ - $\overline{\text{CS4}}$	output	chip select output for external bus	PCS2-PCS4/SEG24-SEG26
INT0-INT2	input	external interrupt request	P61-P63/SEG35-SEG37
V_{LCD0} - V_{LCD2}	–	LCD controller/driver power supply	-
COM0-COM3	output	LCD controller/driver common line 0-3	-
TIE0	input	Timer E channel 0 capture 0 input	P30/INTPE00
TOE10-TOE40	I/O	Timer E channel 0 capture 1-4 input/compare output	P31-P34/INTPE10-INTPE40
TCLRE0	input	Timer E channel 0 capture 5 input or timer clear input	P35/INTPE50
TIE1	input	Timer E channel 1 capture 0 input	P40/INTPE01
TOE11-TOE41	I/O	Timer E channel 1 capture 1-4 input/compare output	P41-P44/INTPE11-INTPE41
TCLRE1	input	Timer E channel 1 capture 5 input or timer clear input	P45/INTPE51
TIE2	input	Timer E channel 2 capture 0 input	P50/INTPE02
TOE12-TOE42	I/O	Timer E channel 2 capture 1-4 input/compare output	P51-P54/INTPE12-INTPE42
TCLRE2	input	Timer E channel 2 capture 5 input or timer clear input	P55/INTPE52
SEG0-SEG7	I/O	LCD-driver segment lines	A23-A16
SEG8-SEG15	I/O		A15-A8
SEG16-SEG23	I/O		A7-A0
SEG24-SEG26	I/O		PCS2-PCS4/$\overline{\text{CS2}}$ - $\overline{\text{CS4}}$
SEG27	I/O		PCT0/$\overline{\text{LWR}}$
SEG28	I/O		PCT1/$\overline{\text{UWR}}$
SEG29	I/O		PCT2
SEG30	I/O		PCT3
SEG31	I/O		PCT4/ $\overline{\text{RD}}$
SEG32	I/O		PCM0/$\overline{\text{WAIT}}$
SEG33	I/O		PCM1/CLKOUT
SEG34	I/O		P60/CCLK
SEG35-SEG37	I/O		P61-P63/INT0-INT2
SEG38	I/O		P64/RXD2
SEG39	I/O		P65/TXD2

Notes: **1.** All V_{DD3} pins have to be connected to each other. On each pin of V_{DD3}, a capacitor has to be attached as tight as possible to the pin.

2. On CV_{DD}, a capacitor has to be attached as tight as possible to the pin. V_{DD3} and CV_V must not be connected.

The capacitors used should have only very low serial impedance.

(3) Pin related to µPD70(F)3123 status

Operating Status Pin	RESET	STOP	WATCH	IDLE	HALT	idle state (TI)
D0 to D15	N.A.	Hi-Z/--[1]	Hi-Z/--[1]	Hi-Z/--[1]	operate	operate
A0 to A23	N.A.	Hi-Z	Hi-Z	Hi-Z	operate	operate
$\overline{\text{CS2}}$ to $\overline{\text{CS4}}$	N.A.	H	H	H	operate	operate
$\overline{\text{LWR}}$, $\overline{\text{UWR}}$	N.A.	H	H	H	operate	operate
$\overline{\text{RD}}$	N.A.	H	H	H	operate	operate
$\overline{\text{WAIT}}$	N.A.	--	--	--	operate	operate
CLKOUT	N.A.	L	L	L	operate	operate
VCMPOUT	--	operate	operate	operate	operate	operate
VCMPIN	--	operate	operate	operate	operate	operate
SEG[39:0]	N.A.	HOLD	operate	operate	operate	operate
COM[3:0]	Hi-Z	HOLD	operate	operate	operate	operate
TIExy	N.A.	--	--	--	operate	operate
INTPExy INT[2:0] NMI	N.A.	operate	operate	operate	operate	operate
TOExy	N.A.	HOLD	HOLD	HOLD	operate	operate
TCLRE[2:0]	N.A.	--	--	--	operate	operate
SO1, SO0	N.A.	HOLD	HOLD	HOLD	operate	operate
SI1, SI0	N.A.	--	--	--	operate	operate
$\overline{\text{SCK1}}$, $\overline{\text{SCK0}}$	N.A.	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	operate	operate
RXD2 to RXD0	N.A.	--	--	--	operate	operate
TXD2 to TXD0	N.A.	HOLD	HOLD	HOLD	operate	operate
CRXD3 to CRXD1	N.A.	--	--	--	operate	operate
CTXD3 to CTXD1	N.A.	HOLD	HOLD	HOLD	operate	operate
CCLK	N.A.	--	--	--	operate	operate
ANI11 to ANI0	--	--	--	--	operate	operate
P1, P2, P3, P4, P5, P6	Hi-Z	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	operate	operate
PAL, PAH, PDL,PCS, PCT, PCM	Hi-Z	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	operate	operate

Remarks: **1.** N.A. : not available
2. -- : input data is not sampled
3. [1] : output / input
4. Hi-Z : High Impedance
5. x : 0 - 5
6. y : 0 - 2

2.2 Description of Pin Functions

(1) P10 to P17 (Port 1) … Input/output

Port 1 is an 8-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as an input/output port, in control mode, P10 to P17 operate as serial interface (UART1, FCAN) input/output.
An operation mode of port or control mode can be selected for each bit and specified by the port 1 mode control register (PMC1).

(a) Port mode
P10 to P17 can be set to input or output in 1-bit units using the port 1 mode register (PM1).

(b) Control mode
P10 to P17 can be set to port or control mode in 1-bit units using PMC1.

(c) CTXD1, CTXD2, CTXD3 (Transmit data for controller area network) … Output
This pin outputs FCAN serial transmit data.

(d) CRXD1, CRXD2, CRXD3 (Receive data for controller area network) … Input
This pin inputs FCAN serial receive data.

(e) TXD1 (Transmit data) … Output
These pins output serial transmit data of UART1.

(f) RXD1 (Receive data) … Input
These pins input serial receive data of UART1.

(2) P20 to P27 (Port 2) … Input/output

Port 2 is an 8-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as an input/output port, in control mode, P20 to P27 operate as serial interface (CSI0, CSI1, UART0) input/output.
An operation mode of port or control mode can be selected for each bit and specified by the port 2 mode control register (PMC2).

(a) Port mode

P20 to P27 can be set to input or output in 1-bit units using the port 2 mode register (PM2).

(b) Control mode

P20 to P27 can be set to port or control mode in 1-bit units using PMC2.

(c) SO0, SO1 (Serial output) … Output

These pins output CSI0 and CSI1 serial transmit data.

(d) SI0, SI1 (Serial input) … Input

These pins input CSI0 and CSI1 serial receive data.

(e) $\overline{\text{SCK0}}$, $\overline{\text{SCK1}}$ (Serial clock) … Input/output

These are CSI0 and CSI1 serial clock input/output pins.

(f) TXD0 (Transmit data) … Output

These pins output serial transmit data of UART0.

(g) RXD0 (Receive data) … Input

These pins input serial receive data of UART0.

(3) P30 to P35 (Port 3) … Input/output

Port 3 is a 6-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as an input/output port, in control mode, P30 to P35 operate as RPU input/output and external interrupt request input.
An operation mode of port or control mode can be selected for each bit and specified by the port 3 mode control register (PMC3).

(a) Port mode

P30 to P35 can be set to input or output in 1-bit units using the port 3 mode register (PM3).

(b) Control mode

P30 to P35 can be set to port or control mode in 1-bit units using PMC3.

(c) TOE10 to TOE40 (Timer output) … Output

These pins output a timer E pulse signal.

(d) TIE0 (Timer input) … Input

This is a timer E external counter clock input pin.

(e) TCLRE0 (Timer clear) … Input

This is a timer E clear signal input pin.

(f) INPT00 to INTP50 (Interrupt request from peripherals) … Input

These are external interrupt request input pins and timer E external capture trigger input pins.

(4) P40 to P45 (Port 4) … Input/output

Port 4 is a 6-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as an input/output port, in control mode, P40 to P45 operate as RPU input/output and external interrupt request input.
An operation mode of port or control mode can be selected for each bit and specified by the port 4 mode control register (PMC4).

(a) Port mode

P40 to P45 can be set to input or output in 1-bit units using the port 4 mode register (PM4).

(b) Control mode

P40 to P45 can be set to port or control mode in 1-bit units using PMC4.

(c) TOE11 to TOE41 (Timer output) … Output

These pins output a timer E pulse signal.

(d) TIE1 (Timer input) … Input

This is a timer E external counter clock input pin.

(e) TCLRE1 (Timer clear) … Input

This is a timer E clear signal input pin.

(f) INPT01 to INTP51 (Interrupt request from peripherals) … Input

These are external interrupt request input pins and timer E external capture trigger input pins.

(5) P50 to P55 (Port 5) … Input/output

Port 5 is a 6-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as an input/output port, in control mode, P50 to P55 operate as RPU input/output and external interrupt request input.
An operation mode of port or control mode can be selected for each bit and specified by the port 5 mode control register (PMC5).

(a) Port mode
P50 to P55 can be set to input or output in 1-bit units using the port 5 mode register (PM5).

(b) Control mode
P50 to P55 can be set to port or control mode in 1-bit units using PMC5.

(c) TOE12 to TOE42 (Timer output) … Output
These pins output a timer E pulse signal.

(d) TIE2 (Timer input) … Input
This is a timer E external counter clock input pin.

(e) TCLRE2 (Timer clear) … Input
This is a timer E clear signal input pin.

(f) INPT02 to INTP52 (Interrupt request from peripherals) … Input
These are external interrupt request input pins and timer E external capture trigger input pins.

(6) P60 to P65 (Port 6) … Input

Port 6 is an input/output port.
Besides functioning as an input port, in control mode, P60 to P65 operate as segment signal output of LCD controller/driver, external CAN clock supply, serial interface (UART2) input/output and external interrupt request input.
An operation mode of port or control mode can be selected for each bit and specified by the port 6 mode control register (PMC6).

(a) Port mode
P60 to P65 can be set to input or output in 1-bit units using the port 6 mode register (PM6).

(b) Control mode
P60 to P65 can be set to port or control mode in 1-bit units using PMC6.

(c) CCLK (External CAN clock input) … Input
This inputs the external CAN clock supply.

(d) INT0 - INT2 (Interrupt request from peripherals) … Input
These are external interrupt request input pins.

(e) SEG34 - SEG 39 Segment Signal output of LCD controller/driver … Output
These pins functions as segment signal output of LCD controller/driver.

(f) TXD2 (Transmit data) … Output
These pins output serial transmit data of UART2.

(g) RXD2 (Receive data) … Input
These pins input serial receive data of UART2.

(7) PAL0 to PAL15 (Port AL) … Input/output

Port AL is an 16-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as a port, in control mode (external expansion mode), these operate as the address bus (A0 to A15) when memory is expanded externally and segment signal output of LCD controller/driver.
An operation mode of port or control mode can be selected for each bit and specified by the port AL mode control register (PMCAL).

(a) Port mode

PAL0 to PAL15 can be set to input or output in 1-bit units using the port AL mode register (PMAL).

(b) Control mode

PAL0 to PAL15 can be used as A0 to A15 by using PMCAL.

(c) A0 to A15 (Address) … Output

This pin outputs the lower 16-bit address of the 24-bit address in the address bus on an external access.

(d) SEG8 - SEG 23 Segment Signal output of LCD controller/driver … Output

These pins functions as segment signal output of LCD controller/driver.

(8) PAH0 to PAH7 (Port AH) … Input/output

Port AH is an 8-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as a port, in control mode (external expansion mode), these operate as the address bus (A16 to A23) for when memory is expanded externally and segment signal output of LCD controller/driver.
An operation mode of port or control mode can be selected for each bit and specified by the port AH mode control register (PMCAH).

(a) Port mode

PAH0 to PAH7 can be set to input or output in 1-bit units using the port AH mode register (PMAH).

(b) Control mode

PAH0 to PAH7 can be used as A16 to A23 by using PMCAH.

(c) A16 to A23 (Address) … Output

This pin outputs the upper 8-bit address of the 24-bit address in the address bus on an external access.

(d) SEG0 - SEG7 Segment Signal output of LCD controller/driver … Output

These pins functions as segment signal output of LCD controller/driver.

(9) PDL0 to PDL15 (Port DL) … Input/output

Port DL is a 16-/8-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as a port, in control mode (external expansion mode), these operate as the data bus (D0 to D15) for when memory is expanded externally.
An operation mode of port or control mode can be selected for each bit and specified by the port DL mode control register (PMCDL).

(a) Port mode

PDL0 to PDL15 can be set to input or output in 1-bit units using the port DL mode register (PMDL).

(b) Control mode

PDL0 to PDL15 can be used as D0 to D15 by using PMCDL.

(10) PCS2 to PCS4 (Port CS) … Input/output

Port CS is a 3-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as a port, in control mode (external expansion mode), it operates as control signal output when memory is expanded externally and segment signal output of LCD controller/driver.
An operation mode of port or control mode can be selected for each bit and specified by the port CS mode control register (PMCCS).

(a) Port mode

PCS2 to PCS4 can be set to input or output in 1-bit units using the port CS mode register (PMCS).

(b) Control mode

PCS2 to PCS4 can be used as CS2 to CS4 by using PMCCS.

(c) $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$(Chip select) … Output

This is the chip select signal for external SRAM, external ROM, or external peripheral I/O.
The signal CSn is assigned to memory block n (n = 2 to 4).
This is active for the period during which a bus cycle that accesses the corresponding memory block is activated.
It is inactive in an idle state (TI).

(d) SEG24 - SEG 26 Segment Signal output of LCD controller/driver … Output

These pins functions as segment signal output of LCD controller/driver.

(11) PCT0 to PCT4 (Port CT) … Input/output

Port CT is a 5-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as a port, in control mode (external expansion mode), it operates as control signal output when memory is expanded externally and segment signal output of LCD controller/driver.
An operation mode of port or control mode can be selected for each bit and specified by the port CT code control register (PMCCT).

(a) Port mode

PCT0 to PCT4 can be set to input or output in 1-bit units using the port CT mode register (PMCT).

(b) Control mode

PCT0 to PCT4 can be used as $\overline{\text{LWR}}$, $\overline{\text{UWR}}$, $\overline{\text{RD}}$ by using PMCCT.

(c) $\overline{\text{LWR}}$ (Lower byte write strobe) … Output

This is a strobe signal that shows that the executing bus cycle is a write cycle for SRAM, external ROM, or an external peripheral I/O area.
In the data bus, the lower byte is in effect. If the bus cycle is a lower memory write, it becomes active at the falling edge of a T1 state CLKOUT signal and becomes inactive at the falling edge of a T2 state CLKOUT signal.

(d) $\overline{\text{UWR}}$ (Upper byte write strobe) … Output

This is a strobe signal that shows that the executing bus cycle is a write cycle for SRAM, external ROM, or an external peripheral I/O area.
In the data bus, the upper byte is in effect. If the bus cycle is an upper memory write, it becomes active at the falling edge of a T1 state CLKOUT signal and becomes inactive at the falling edge of a T2 state CLKOUT signal.

(e) $\overline{\text{RD}}$ (Read strobe) … Output

This is a strobe signal that shows that the executing bus cycle is a read cycle for SRAM, external ROM, or external peripheral I/O. It is inactive in an idle state (TI).

(f) SEG27 - SEG 31 Segment Signal output of LCD controller/driver … Output

These pins functions as segment signal output of LCD controller/driver.

(12) PCM0 to PCM1 (Port CM) … Input/output

Port CM is a 2-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as a port, in control mode (external expansion mode), it operates as control signal output when memory is expanded externally and segment signal output of LCD controller/driver.
An operation mode of port or control mode can be selected for each bit and specified by the port CM code control register (PMCCM).

(a) Port mode

PCM0, PCM1 can be set to input or output in 1-bit units using the port CM mode register (PMCM).

(b) Control mode

PCM0 to PCM1 can be used as CLKOUT, $\overline{\text{WAIT}}$ by using PMCCM.

(c) $\overline{\text{WAIT}}$ (Wait) … Input

This control signal input pin, which inserts a data wait in a bus cycle, can input asynchronously with respect to a CLKOUT signal. Sampling is done at the falling edge of a CLKOUT signal in a bus cycle in a T1 or TW state. If the setup or hold time is not secured in the sampling timing, wait insertion may not be performed.

(d) CLKOUT (Clock output) … Output

This is an internal system clock output pin. To perform CLKOUT output, set this pin to control mode using the port CM mode control register (PMCCM).

(e) SEG32 - SEG33 Segment Signal output of LCD controller/driver … Output

These pins functions as segment signal output of LCD controller/driver.

(13) ANI00 to ANI11 (Analog input) … Input

These are analog input pins to the A/D converter.

(14) COM0 to COM1 (Common signal)... Output

These are the common signal output of LCD-controller/driver

(15) CLKSEL (Clock generator operating mode select) … Input

This is the input pin that specifies the operation mode of the clock generator. Fix it so that the input level does not change during operation.

(16) VCMPIN (Voltage Comparator Input)... Input

This pin is the input pin for the voltage comparator.

(17) VCMPOUT (Voltage Comparator Output)... Output

This pin is the output pin of the voltage comparator.

(18) MODE0 to MODE3 (Mode) … Input

These are the input pins that specify the operation mode. Operation modes are broadly divided into normal operation modes and flash memory programming mode. The operation mode is determined by sampling the status of each of pins MODE0 to MODE3 on a reset.

Fix these so that the input level does not change during operation.

(19) $\overline{\text{RESET}}$ (Reset) … Input

RESET input is asynchronous input. When a signal having a certain low level width is input in asynchronous with the operation clock, a system reset that takes precedence over all operations occurs. Besides a normal initialize or start, this signal is also used to release a standby mode (HALT, IDLE, Watch, software STOP).

(20) NMI (NON-Maskable Interrupt Request)... input

This is the non-maskable interrupt request input pin.

(21) X1, X2 (Crystal)

These pins connect a resonator for system clock generation.

They also can input external clocks. For external clock input, connect to the X1 pin and leave the X2 pin open.

(22) CV_{DD} (Power supply for clock generator)

This is the positive power supply pin for the clock generator.

(23) CV$_{SS}$ (Ground for clock generator)

This is the ground pin for the clock generator.

(24) V$_{LCD0}$ to V$_{LCD2}$ (LCD driver supply)

These are the positive power supply pins for the LCD controller/driver.

(25) V$_{DD50}$ to V$_{DD54}$ (Power supply)

These are the positive power supply pins for the peripheral interface.

(26) V$_{SS50}$ to V$_{SS54}$ (Ground)

These are the ground pins for the peripheral interface.

(27) V$_{DD30}$ to V$_{DD33}$ (Power supply)

These are the positive power supply pins for the internal CPU.

(28) V$_{SS30}$ to V$_{SS33}$ (Ground)

These are the ground pins for the internal CPU.

(29) AV$_{DD}$ (Analog power supply)

This is the analog positive power supply pin for the A/D converter.

(30) AV$_{SS}$ (Analog ground)

This is the ground pin for the A/D converter.

(31) AV$_{REF}$ (Analog reference voltage) … Input

This is the reference voltage supply pin for the A/D converter.

(32) V$_{PP}$ (Programming power supply)

These are positive power supply pins used for flash memory programming mode. These pins are used for the µPD70F3123.

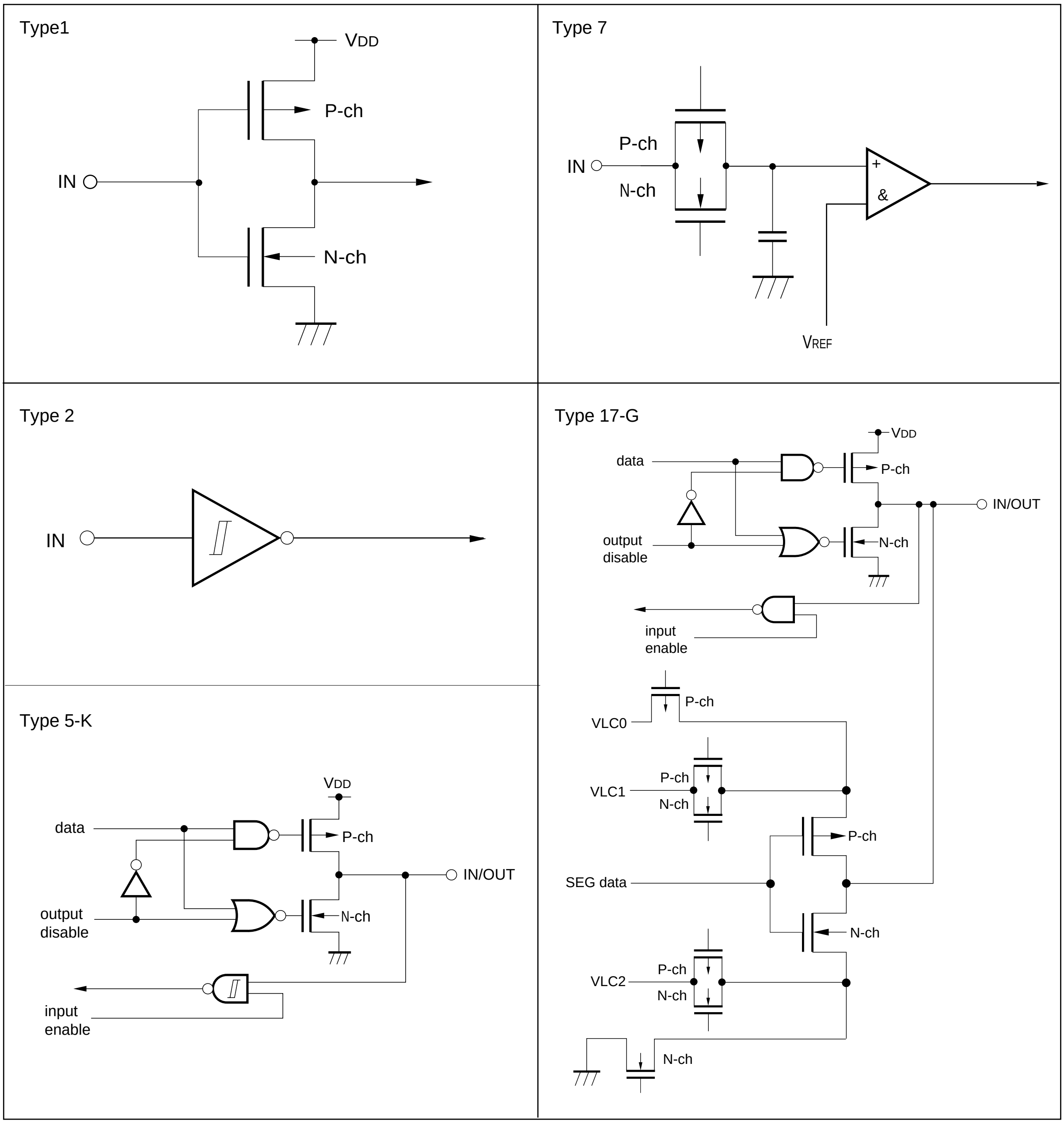
2.3 Types of Pin I/O Circuit and Connection of Unused Pin

Pin			I/O Circuit Type	Recommended Connection
P10	CRXD1		5-K	For input: individually connect to V_{DD5} or V_{SS5} via a resistor. For output: leave open.
P11	CTXD1			
P12	CRXD2			
P13	CTXD2			
P14	CRXD3			
P15	CTXD3			
P16	RXD1			
P17	TXD1			
P20	SI0		5-K	
P21	SO0			
P22	$\overline{\text{SCK0}}$			
P23	SI1			
P24	SO1			
P25	$\overline{\text{SCK1}}$			
P26	RXD0			
P27	TXD0			
P30	TIE0	INTPE00	5-K	
P31	TOE10	INTPE10		
P32	TOE20	INTPE20		
P33	TOE30	INTPE30		
P34	TOE40	INTPE40		
P35	TCLRE0	INTPE50		
P40	TIE1	INTPE01	5-K	
P41	TOE11	INTPE11		
P42	TOE21	INTPE21		
P43	TOE31	INTPE31		
P44	TOE41	INTPE41		
P45	TCLRE1	INTPE51		
P50	TIE2	INTPE02	5-K	
P51	TOE12	INTPE12		
P52	TOE22	INTPE22		
P53	TOE32	INTPE32		
P54	TOE42	INTPE42		
P55	TCLRE2	INTPE52		
P60	CCLK	SEG34	17-G	
P61	INT0	SEG35		
P62	INT1	SEG36		
P63	INT2	SEG37		
P64	RXD2	SEG38		
P65	TXD2	SEG39		

Pin			I/O Circuit Type	Recommended Connection
PAL0	A0	SEG23	17-G	For input: individually connect to V_{DD5} or V_{SS5} via a resistor. For output: leave open.
PAL1	A1	SEG22		
PAL2	A2	SEG21		
PAL3	A3	SEG20		
PAL4	A4	SEG19		
PAL5	A5	SEG18		
PAL6	A6	SEG17		
PAL7	A7	SEG16		
PAL8	A8	SEG15		
PAL9	A9	SEG14		
PAL10	A10	SEG13		
PAL11	A11	SEG12		
PAL12	A12	SEG11		
PAL13	A13	SEG10		
PAL14	A14	SEG9		
PAL15	A15	SEG8		
PAH0	A16	SEG7	17-G	
PAH1	A17	SEG6		
PAH2	A18	SEG5		
PAH3	A19	SEG4		
PAH4	A20	SEG3		
PAH5	A21	SEG2		
PAH6	A22	SEG1		
PAH7	A23	SEG0		
PCM0	$\overline{\text{WAIT}}$	SEG32	17-G	
PCM1	CLKOUT	SEG33		
PCS2	$\overline{\text{CS2}}$	SEG24	17-G	
PCS3	$\overline{\text{CS3}}$	SEG25		
PCS4	$\overline{\text{CS4}}$	SEG26		
PCT0	$\overline{\text{LWR}}$	SEG27	17-G	
PCT1	$\overline{\text{UWR}}$	SEG28		
PCT2		SEG29	17-G	
PCT3		SEG30		
PCT4	$\overline{\text{RD}}$	SEG31		

Pin			I/O Circuit Type	Recommended Connection
PDL0	D0		5-K	For input: individually connect to V_{DD5} or V_{SS5} via a resistor. For output: leave open.
PDL1	D1			
PDL2	D2			
PDL3	D3			
PDL4	D4			
PDL5	D5			
PDL6	D6			
PDL7	D7			
PDL8	D8			
PDL9	D9			
PDL10	D10			
PDL11	D11			
PDL12	D12			
PDL13	D13			
PDL14	D14			
PDL15	D15			
AIN0-AIN11			7	Individually connect to AV_{DD} or AV_{SS} via a resistor.
MODE0			2	V_{SS5x}
MODE1			2	V_{DD5x}
MODE2, MODE3			2	V_{SS5x}
COM0-COM3			17-G	-
V_{PP0},V_{PP1}			-	connect to V_{SS} via a resistor.
V_{LCD0}-V_{LCD2}			-	-
VCPMOUT	NMI		5-K	-
VCMPIN			1	connect to V_{DD5} via a resistor.
$\overline{\text{RESET}}$			2	-
CLKSEL			2	connect to V_{DD5} or V_{SS5} via a resistor.
IC			1	V_{SS5x}
CLOCKIN			2	connect to V_{SS5} via a resistor.
X2			-	Please refer to the datasheet
AV_{DD}			-	V_{DD5x}
AV_{REF}			-	AV_{DD}
AV_{SS}			-	V_{SS5x}

Figure 2-1: Pin I/O Circuits



[MEMO]

Chapter 3 CPU Function

The CPU of the V850E/CA1 / ATOMIC is based on a RISC architecture and executes almost all the instructions in one clock cycle, using a 5-stage pipeline control.

3.1 Features

- Minimum instruction cycle: 50 ns (@ internal 20 MHz operation)
- Memory space
  - Program space: 64 MB linear
  - Data space: 4 GB linear
- Thirty-two 32-bit general registers
- Internal 32-bit architecture
- Five-stage pipeline control
- Multiplication/division instructions
- Saturated operation instructions
- One-clock 32-bit shift instruction (barrel shifter)
- Long/short instruction format
- Four types of bit manipulation instructions
  - Set
  - Clear
  - Not
  - Test

3.2 CPU Register Set

The registers of the V850E/CA1 / ATOMIC can be classified into two categories: a general program register set and a dedicated system register set. All the registers are 32-bit width.
For details, refer to V850E User’s Manual Architecture.

Figure 3-1: CPU Register Set

(1) Program register set

31	0
r0	(Zero Register)
r1	(Reserved for Assembler)
r2	(Interrupt Stack Pointer)
r3	(Stack Pointer (SP))
r4	(Global Pointer (GP))
r5	(Text Pointer (TP))
r6	
r7	
r8	
r9	
r10	
r11	
r12	
r13	
r14	
r15	
r16	
r17	
r18	
r19	
r20	
r21	
r22	
r23	
r24	
r25	
r26	
r27	
r28	
r29	
r30	(Element Pointer (EP))
r31	(Link Pointer (LP))

31	0
PC	(Program Counter)

(2) System register set

31	0
EIPC	(Status Saving Register during interrupt)
EIPSW	(Status Saving Register during interrupt)

FEPC	(Status Saving Register during NMI)
FEPSW	(Status Saving Register during NMI)

ECR	(Interrrupt Source Register)

PSW	(Program Status Word)

CTPC	(Status Saving Register during CALLT execution)
CTPSW	(Status Saving Register during CALLT execution)

DBPC	(Status Saving Register during exception/debug trap)
DBPSW	(Status Saving Register during exception/debug trap)

CTBP	(CALLT Base Pointer)

3.2.1 Program register set

The program register set includes general registers and a program counter.

(1) General registers

Thirty-two general registers, r0 to r31, are available. Any of these registers can be used as a data variable or address variable.
However, r0 and r30 are implicitly used by instructions, and care must be exercised when using these registers. r0 is a register that always holds 0, and is used for operations using 0 and offset 0 addressing. r30 is used, by means of the SLD and SST instructions, as a base pointer for when memory is accessed. Also, r1, r3 to r5, and r31 are implicitly used by the assembler and C compiler. Therefore, before using these registers, 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.

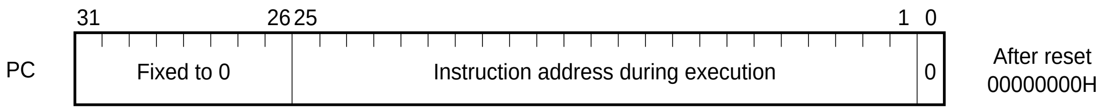
Table 3-1: Program Registers

Name	Usage	Operation
r0	Zero register	Always holds 0
r1	Assembler-reserved register	Working register for generating 32-bit immediate data
r2	Address/data variable registers	
r3	Stack pointer	Used to generate stack frame when function is called
r4	Global pointer	Used to access global variable in data area
r5	Text pointer	Register to indicate the start of the text area (where program code is located)
r6 to r29	Address/data variable registers	
r30	Element pointer	Base pointer when memory is accessed
r31	Link pointer	Used by compiler when calling function
PC	Program counter	Holds instruction address during program execution

(2) Program counter

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

Figure 3-2: Program Counter (PC)



3.2.2 System register set

System registers control the status of the CPU and hold interrupt information.
To read/write these system registers, use the system register load/store instruction (LDSR or STSR instruction) with a specific system register number indicated below.

Table 3-2: System Register Numbers

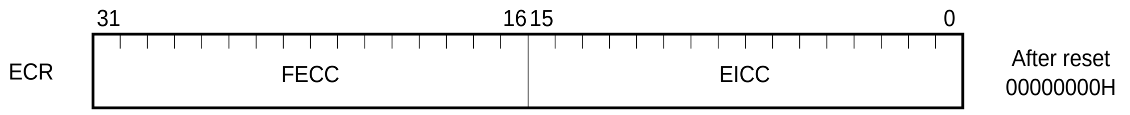
No.	System Register Name	Operand Specification	
		LDSR Instruction	STSR Instruction
0	Status saving register during interrupt (EIPC)[Note 1]	O	O
1	Status saving register during interrupt (EIPSW)	O	O
2	Status saving register during NMI (FEPC)	O	O
3	Status saving register during NMI (FEPSW)	O	O
4	Interrupt source register (ECR)	×	O
5	Program status word (PSW)	O	O
6 to 15	Reserved number for future function expansion (operations that access these register numbers cannot be guaranteed).	×	×
16	Status saving register during CALLT execution (CTPC)	O	O
17	Status saving register during CALLT execution (CTPSW)	O	O
18	Status saving register during exception/debug trap (DBPC)	O[Note 2]	O
19	Status saving register during exception/debug trap (DBPSW)	O[Note 2]	O
20	CALLT base pointer (CTBP)	O	O
21 to 31	Reserved number for future function expansion (operations that access these register numbers cannot be guaranteed).	×	×

Notes: **1.** Because this register has only one set, to approve multiple interrupts, it is necessary to save this register by program.
2. Access is only possible while the DBTRAP instruction is executed.

Caution: Even if bit 0 of EIPC, FEPC, or CTPC is set to 1 with the LDSR instruction, bit 0 will be ignored when the program returned by RETI instruction after interrupt servicing (because bit 0 of the PC is fixed to 0). When setting the value of EIPC, FEPC, or CTPC, use the even value (bit 0 = 0).

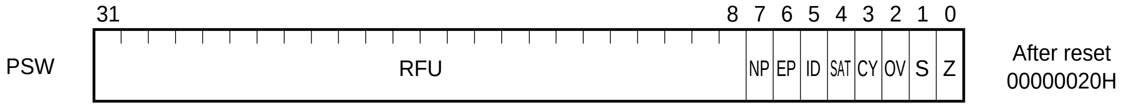
Remark: O:Access allowed
×:Access prohibited

Figure 3-3: Interrupt Source Register (ECR)



Bit Position	Bit Name	Function
31 to 16	FECC	Exception code of non-maskable interrupt (NMI)
15 to 0	EICC	Exception code of exception/maskable interrupt

Figure 3-4: Program Status Word (PSW)



Bit Position	Flag	Function
31 to 8	RFU	Reserved field (fixed to 0).
7	NP	Indicates that non-maskable interrupt (NMI) processing is in progress. This flag is set when NMI is accepted, and disables multiple interrupts. 0: NMI servicing not under execution. 1: NMI servicing under execution.
6	EP	Indicates that exception processing is in progress. This flag is set when an exception is generated. Moreover, interrupt requests can be accepted when this bit is set. 0: Exception processing not under execution. 1: Exception processing under execution.
5	ID	Displays whether a maskable interrupt request has been acknowledged or not. 0: Interrupt enabled. 1: Interrupt disabled.
4	SAT[Note]	Displays that the operation result of a saturated operation processing instruction is saturated due to overflow. Due to the cumulative flag, if the operation result is saturated by the saturation operation instruction, this bit is set (1), but is not cleared (0) even if the operation results of subsequent instructions are not saturated. To clear (0) this bit, load the data in PSW. Note that in a general arithmetic operation, this bit is neither set (1) nor cleared (0). 0: Not saturated. 1: Saturated.
3	CY	This flag is set if carry or borrow occurs as result of operation (if carry or borrow does not occur, it is reset). 0: Carry or borrow does not occur. 1: Carry or borrow occurs.
2	OV[Note]	This flag is set if overflow occurs during operation (if overflow does not occur, it is reset). 0: Overflow does not occur. 1: Overflow occurs.
1	S[Note]	This flag is set if the result of operation is negative (it is reset if the result is positive). 0: The operation result was positive or 0. 1: The operation result was negative.
0	Z	This flag is set if the result of operation is zero (if the result is not zero, it is reset). 0: The operation result was not 0. 1: The operation result was 0.

Note: The result of a saturation-processed operation is determined by the contents of the OV and S flags in the saturation operation. Simply setting the OV flag (1) will set the SAT flag (1) in a saturation operation.

Status of Operation Result	Flag Status			Saturation-Processed Operation Result
	SAT	OV	S	
Maximum positive value exceeded	1	1	0	7FFFFFFFH
Maximum negative value exceeded	1	1	1	80000000H
Positive (maximum not exceeded)	Retains the value before operation	0	0	Operation result itself
Negative (maximum not exceeded)			1	

3.3 Operation Modes

3.3.1 Operation modes

The V850E/CA1 / ATOMIC has the following operations modes. Mode specification is carried out by the MODE0 to MODE3 pins.

(1) Normal operation mode

(a) Single-chip mode

Access to the internal ROM is enabled.

In single-chip mode 0, after system reset is cleared, each pin related to the bus interface enters the port mode, program execution branches to the reset entry address of the internal ROM, and instruction processing starts. By setting the PMCAL, PMCAH, PMCDL, PMCCS, PMCCT, and PMCCM registers to control mode by instruction, an external device can be connected to the external memory area.

The initial value of the register differs depending on the mode.

Table 3-3: Register Initial Values by Operation Modes

Operation Mode		PMCAL	PMCAH	PMCDL	PMCCS	PMCCT	PMCCM	BSC
Normal operation mode	Single-chip mode	0000H	0000H	0000H	00H	00H	00H	5555H

(2) Flash memory programming mode (µPD70F3123 only)

If this mode is specified, it becomes possible for the flash programmer to run a program to the internal flash memory.

3.3.2 Operation mode specification

The operation mode is specified according to the status of pins MODE0 to MODE3. In an application system fix the specification of these pins and do not change them during operation. Operation is not guaranteed if these pins are changed during operation.

(a) μPD703123

MODE3	MODE2	MODE1	MODE0	Operation Mode		Remarks
L	L	H	L	Normal operation mode	Single-chip mode	–
Other than above				Setting prohibited		

(b) μPD70F3123

Vpp0/ Vpp1	MODE3	MODE2	MODE1	MODE0	Operation Mode		Remarks
0 V	L	L	H	L	Normal operation mode	Single-chip mode	–
7.8 V	L	L	H	L	Flash memory programming mode		–
Other than above					Setting prohibited		

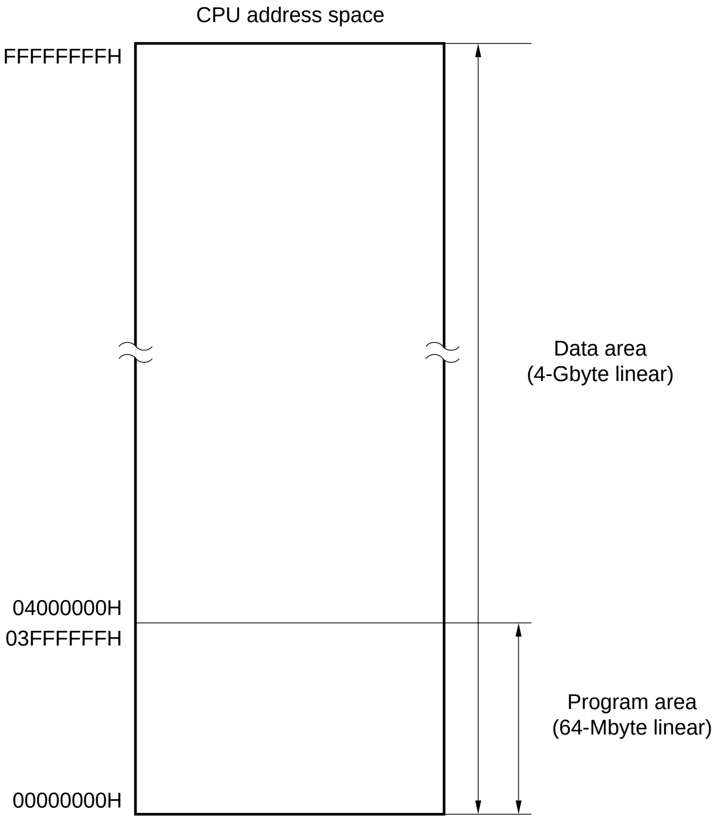
Remarks: **1.** L:Low-level input
2. H:High-level input
3. H/L:High-level, or low-level input (optional)

3.4 Address Space

3.4.1 CPU address space

The CPU of the V850E/CA1 / ATOMIC is of 32-bit architecture and supports up to 4 GB of linear address space (data space) during operand addressing (data access). Also, in instruction address addressing, a maximum of 64 MB of linear address space (program space) is supported.
Figure 3-5 shows the CPU address space.

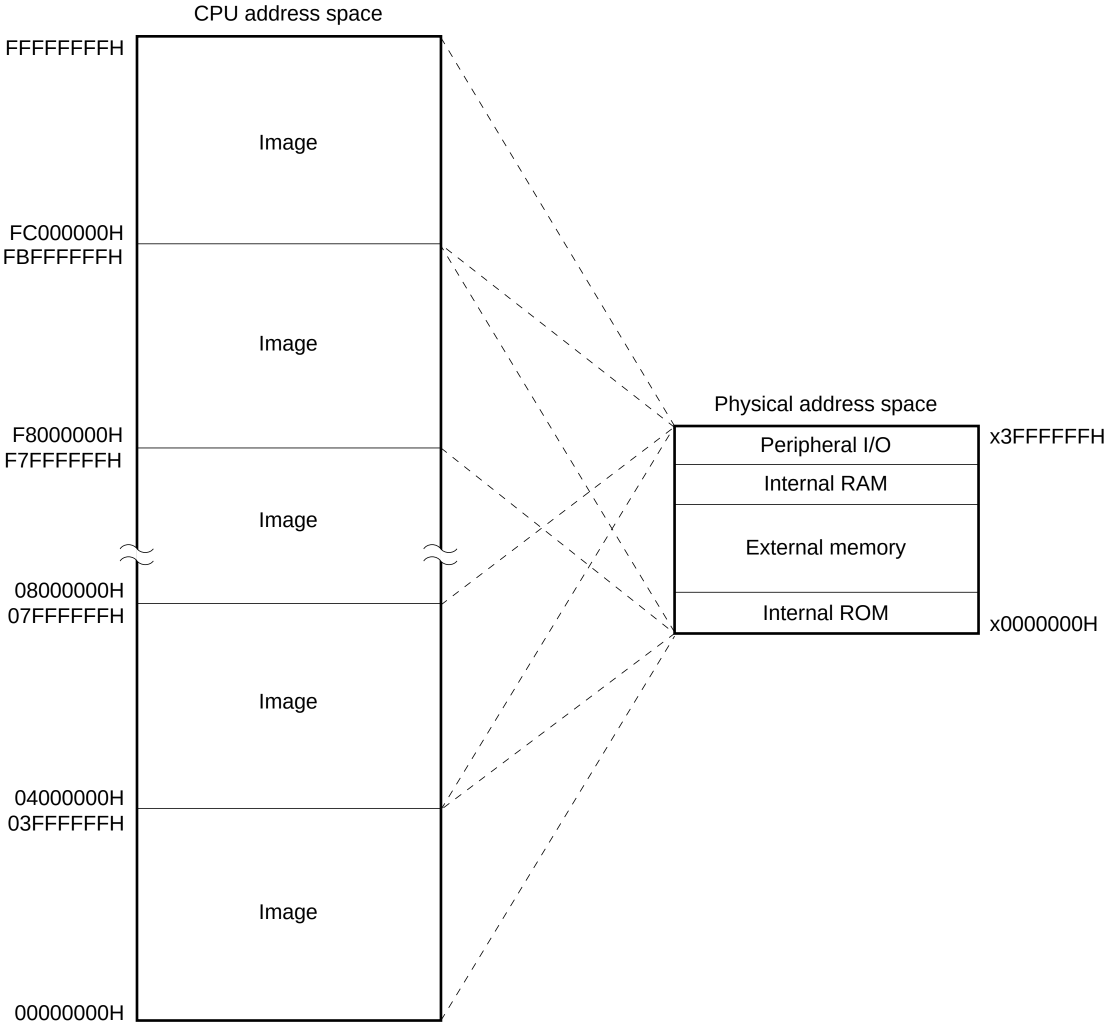
Figure 3-5: CPU Address Space



3.4.2 Image

64 MB physical address space is seen as 64 images in the 4 GB CPU address space. In actuality, the same 64 MB physical address space is accessed regardless of the values of bits 31 to 26 of the CPU address. Figure 3-6 shows the image of the virtual addressing space.
Physical address x0000000H can be seen as CPU address 00000000H, and in addition, can be seen as address 04000000H, address 08000000H, …, address F8000000H, or address FC000000H.

Figure 3-6: Image on Address Space



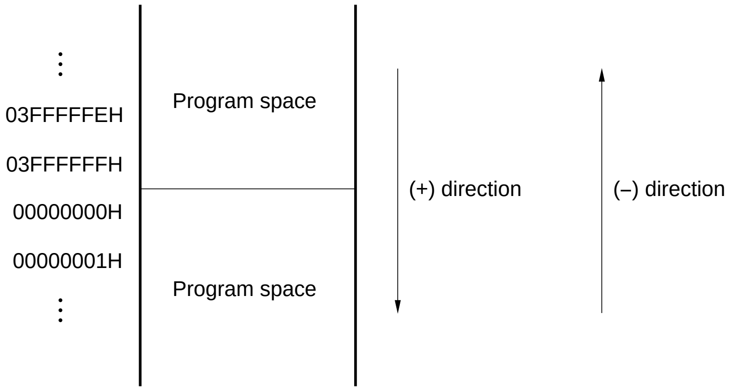
3.4.3 Wrap-around of CPU address space

(1) Program space

Of the 32 bits of the PC (program counter), the higher 6 bits are set to “0”, and only the lower 26 bits are valid. Even if a carry or borrow occurs from bit 25 to 26 as a result of branch address calculation, the higher 6 bits ignore the carry or borrow.

Therefore, the lower-limit address of the program space, address 00000000H, and the upper-limit address 03FFFFFFH become contiguous addresses. Wrap-around refers to the situation that the lower-limit address and upper-limit address become contiguous like this.

Figure 3-7: Wrap-around of Program Space



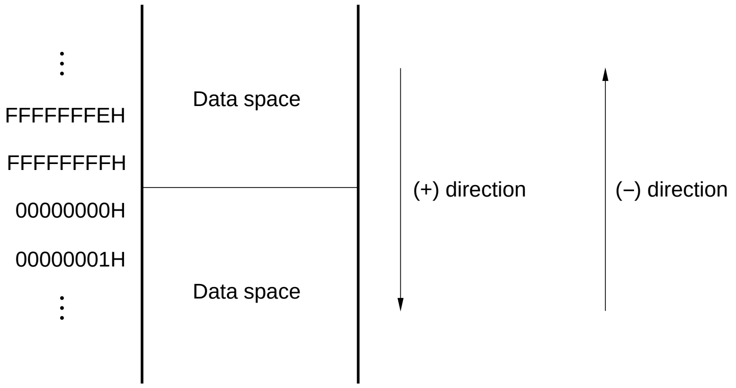
Caution: No instruction can be fetched from the 4 KB area of 03FFF000H to 03FFFFFFH because this area is defined as peripheral I/O area. Therefore, do not execute any branch address calculation in which the result will reside in any part of this area.

(2) Data space

The result of operand address calculation that exceeds 32 bits is ignored.

Therefore, the lower-limit address of the program space, address 00000000H, and the upper-limit address FFFFFFFFH are contiguous addresses, and the data space is wrapped around at the boundary of these addresses.

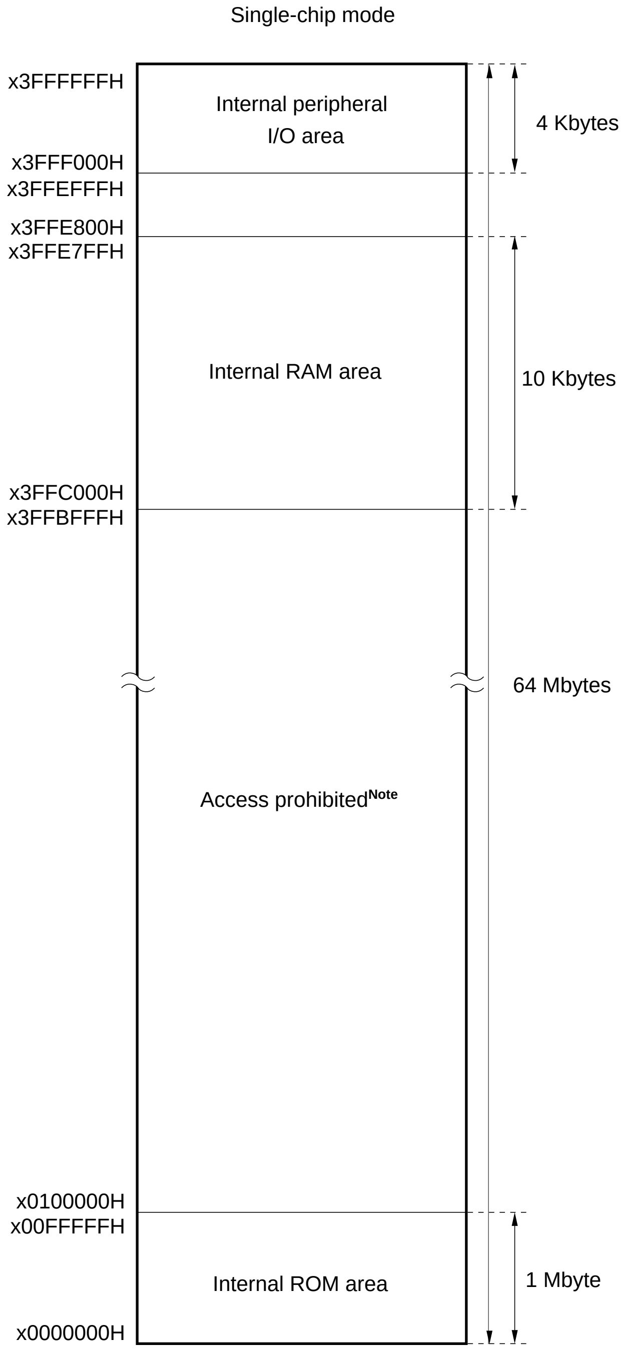
Figure 3-8: Wrap-around of Data Space



3.4.4 Memory map

The V850E/CA1 / ATOMIC reserves areas as shown in Figure 3-9. Each mode is specified by the MODE0 to MODE3 pins.

Figure 3-9: Memory Map (µPD703123, 703F123)



Note: By setting the PMCAL, PMCAH, PMCDL, PMCCS, PMCCT, and PMCCM registers to control mode by instruction, this area can be used as external memory area.

3.4.5 Area

(1) Internal ROM area

(a) Memory map (µPD703123, 70F3123)

1 MB of internal ROM area, addresses 00000H to FFFFFH, is reserved.

<1> µPD703123

256 KB are provided in the following addresses as physical internal ROM (mask ROM).

- Addresses 000000H to 03FFFFH

<2> µPD70F3123

256 KB are provided in the following addresses as physical internal ROM (flash memory).

- Addresses 000000H to 03FFFFH

(b) Interrupt/exception table

The V850E/CA1 / ATOMIC increases the interrupt response speed by assigning handler addresses corresponding to interrupts/exceptions.

The collection of these handler addresses is called an interrupt/exception table, which is located in the internal ROM area. When an interrupt/exception request is accepted, execution jumps to the handler address, and the program written at that memory is executed. Table 3-4 shows the sources of interrupts/exceptions, and the corresponding addresses.

Table 3-3: Interrupt/Exception Table (1/2)

Start Address of Interrupt/ Exception Table	Interrupt/Exception Source
00000000H	RESET
00000010H	NMIVC
00000020H	INTWDT
00000040H	TRAP0n (n = 0 to F)
00000050H	TRAP1n (n = 0 to F)
00000060H	ILGOP/DBTRAP
00000080H	CINTLPOW
00000090H	AD/INTDET
000000A0H	INTWT
000000B0H	TINTCMD0
000000C0H	TINTCMD1
000000D0H	INTWTI
000000E0H	INT0
000000F0H	INT1
00000100H	INT2
00000110H	TINTOVE00
00000120H	TINTOVE10
00000130H	TINTCCE00/INTPE00
00000140H	TINTCCE10/INTPE10
00000150H	TINTCCE20/INTPE20
00000160H	TINTCCE30/INTPE30
00000170H	TINTCCE40/INTPE40
00000180H	TINTCCE50/INTPE50
00000190H	TINTOVE01
000001A0H	TINTOVE11
000001B0H	TINTCCE01/INTPE01
000001C0H	TINTCCE11/INTPE11
000001D0H	TINTCCE21/INTPE21
000001E0H	TINTCCE31/INTPE31
000001F0H	TINTCCE41/INTPE41
00000200H	TINTCCE51/INTPE51
00000210H	TINTOVE02
00000220H	TINTOVE12
00000230H	TINTCCE02/INTPE02
00000240H	TINTCCE12/INTPE12
00000250H	TINTCCE22/INTPE22
00000260H	TINTCCE32/INTPE32
00000270H	TINTCCE42/INTPE42
00000280H	TINTCCE52/INTPE52
00000290H	INTAD
000002A0H	INTMAC
000002B0H	INTACT
000002C0H	CAN1REC

Table 3-3: Interrupt/Exception Table (2/2)

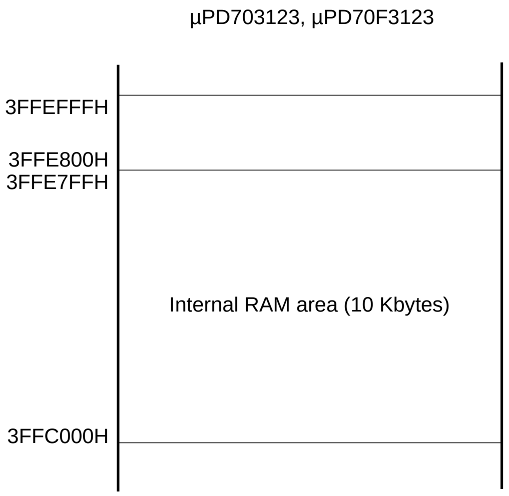
Start Address of Interrupt/ Exception Table	Interrupt/Exception Source
000002D0H	CAN1TRX
000002E0H	CAN1ERR
000002F0H	CAN2REC
00000300H	CAN2TRX
00000310H	CAN2ERR
00000320H	CAN3REC
00000330H	CAN3TRX
00000340H	CAN3ERR
00000350H	INTCSI0
00000360H	INTCSI1
00000370H	INTSER0
00000380H	INTSR0
00000390H	INTST0
000003A0H	INTSER1
000003B0H	INTSR1
000003C0H	INTST1
000003D0H	INTSER2
000003E0H	INTSR2
000003F0H	INTST2
00000400H	INTDMA0
00000410H	INTDMA1
00000420H	INTDMA2
00000430H	INTDMA3
00000440H	INT60[Note]
00000450H	INT61
00000460H	INT62
00000470H	INT63

Note: Reserved for internal use only please leave at RESET value

(2) Internal RAM area

12 KB of memory, addresses 3FFC000H to 3FFEFFFH, are reserved for the internal RAM area.
In the µPD70F3123/µPD703123 the 10 KB of addresses 3FFC000H to 3FFE7FFH are provided as internal physical RAM.

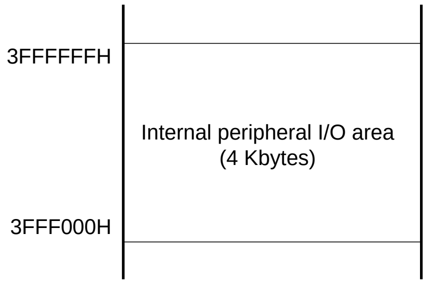
Figure 3-10: Internal RAM Area



(3) Internal peripheral I/O area

4 KB of memory, addresses 3FFF000H to 3FFFFFFH, is provided as an internal peripheral I/O area.

Figure 3-11: Internal Peripheral I/O Area



Peripheral I/O registers associated with the operation mode specification and the state monitoring for the internal peripherals I/O are all memory-mapped to the internal peripheral I/O area. Program fetches cannot be executed from this area.

Cautions: 1. In the V850E/CA1, no registers exist which are capable of word access. But if a register is word accessed, half word access is performed twice in the order of lower address, then higher address of the word area, ignoring the lower 2 bits of the address.

2. For registers in which byte access is possible, if half word access is executed, the higher 8 bits become undefined during the read operation, and the lower 8 bits of data are written to the register during the write operation.

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

4. Addresses 3FFF000H to 3FFFFFFH cannot be specified as the source/destination address of DMA transfer. Be sure to use addresses FFFF000H to FFFFFFFH for source/destination address of DMA transfer.

Additionally to the peripheral I/O area, a 16 KB area is provided as a programmable peripheral I/O area (refer to **3.4.9 Programmable peripheral I/O registers**).

(4) External memory area

The following areas can be used as external memory area.

(a) µPD703123, 70F3123

x0100000H to xFFFBFFFH

Access to the external memory area uses the chip select signal assigned to each memory block (which is carried out in the CS unit set by chip area selection control registers 0 and 1 (CSC0, CSC1)).
Furthermore, the internal ROM, internal RAM, and internal peripheral I/O areas cannot be accessed as external memory areas.

3.4.6 External memory expansion

By setting the port n mode control register (PMCn) to control mode, an external memory device can be connected to the external memory space using each pin of ports AL, AH, DL, CS, CT, and CM. Each register is set by selecting control mode for each pin of these ports using PMCn (n = AL, AH, DL, CS, CT, CM).
Furthermore, the status after reset differs as shown below in accordance with the operating mode specification set by pins MODE0 to MODE3 (please refer to Operation Modes).

(a) In the case of single-chip mode

After reset, since the internal ROM area is accessed, each pin of ports AL, AH, DL, CS, CT, and CM enters the port mode and external devices cannot be used.
To use external memory, set the port n mode control register (PMCn).

Remark: n = AL, AH, DL, CS, CT, CM

3.4.7 Recommended use of address space

The architecture of the V850E/CA1 / ATOMIC requires that a register is utilized for address generation when accessing operand data in the data space. Operand data access from instruction can be directly executed at the address in this pointer register ±32 KB. However, the use of general registers as pointer registers decreases the number of usable general registers for handling variables, but minimizes the deterioration of address calculation performance when changing the pointer value and minimizes the program size as well.
To enhance the efficiency of using the pointer in consideration of the memory map of the V850E/CA1 / ATOMIC, the following points are recommended:

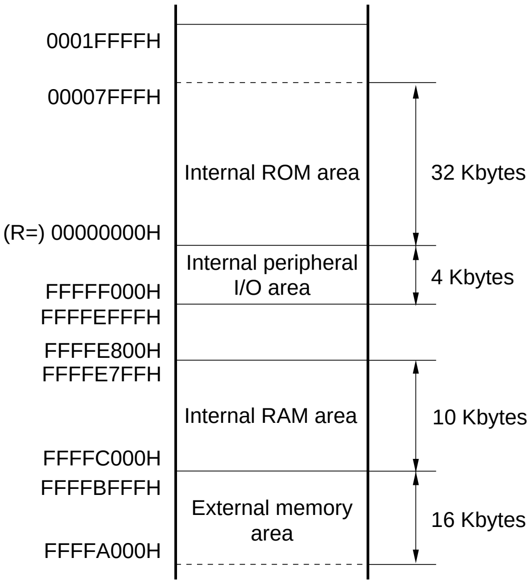
(1) Program space

Of the 32 bits of the PC (program counter), the higher 6 bits are fixed to zero (0), and only the lower 26 bits are valid. Therefore, a contiguous 64 MB space, starting from address 00000000H, unconditionally corresponds to the memory map of the program space.

(2) Data space

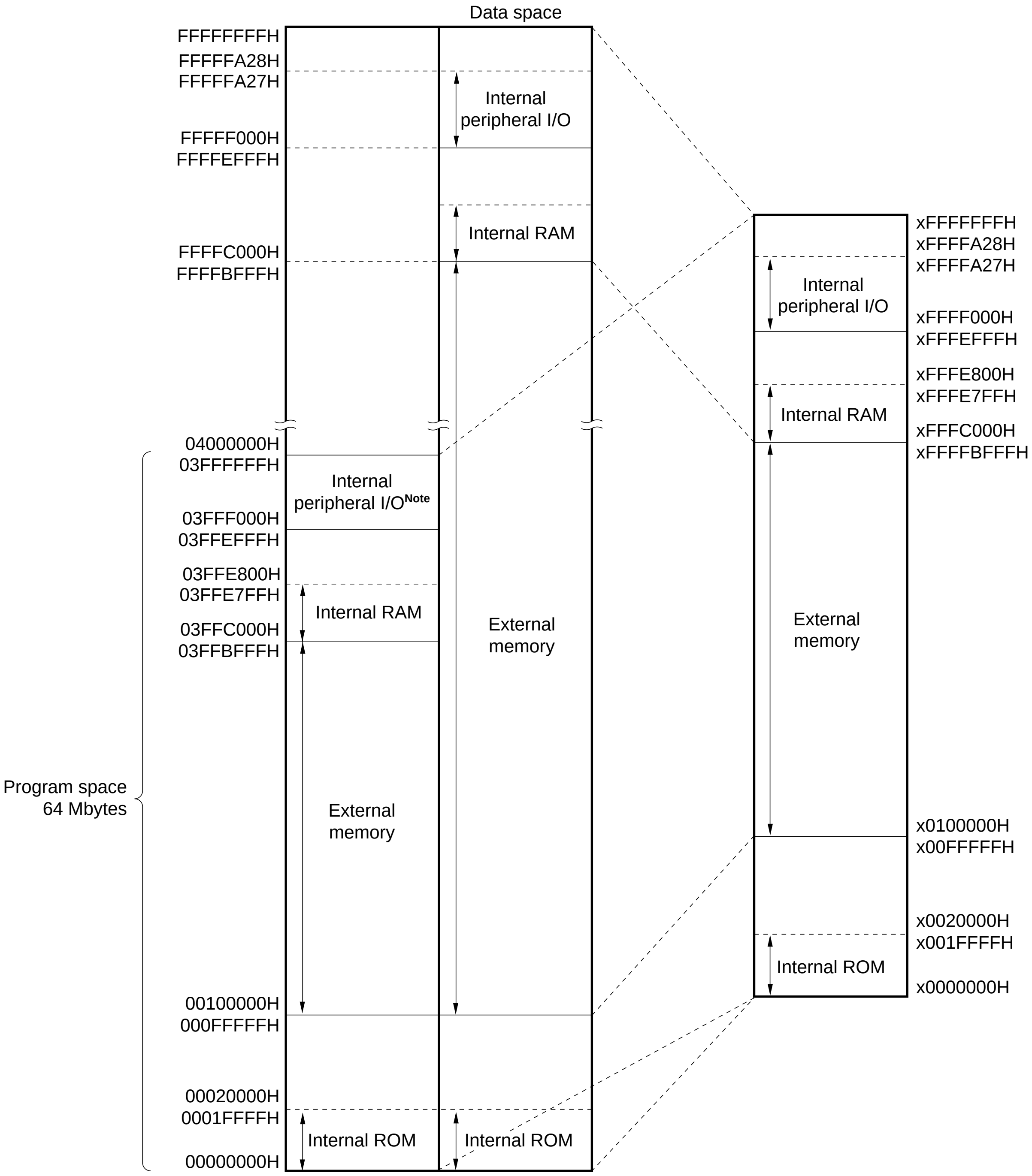
For the efficient use of resources to be performed through the wrap-around feature of the data space, the continuous 16 MB address spaces 00000000H to 00FFFFFFH and FF000000H to FFFFFFFFH of the 4 GB CPU address space are used as the data space. With the V850E/CA1 / ATOMIC, 64 MB physical address space is seen as 64 images in the 4 GB CPU address space. The highest bit (bit 25) of this 26-bit address is assigned as address sign-extended to 32 bits.

Figure 3-12: Example Application of wrap-around (µPD703123)



When R = r0 (zero register) is specified with the LD/ST disp16 [R] instruction, an addressing range of 00000000H ±32 KB can be referenced with the sign-extended, 16-bit displacement value. By mapping the external memory into the 16 KB area in Figure 3-12, all resources including internal hardware can be accessed with the same pointer.
The zero register (r0) is a register set to 0 by hardware, and eliminates the need for additional registers for the pointer.

Figure 3-13: Recommended Memory Map



Note: This area cannot be used as a program area.

Remarks: **1.** The arrows indicate the recommended area.
2. This is a recommended memory map when the µPD703123 is set to single-chip mode, and used as external expansion mode.

3.4.8 Peripheral I/O registers

Table 3-4: List of Peripheral I/O Registers (Sheet 1 of 10)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
FFFFF000H	Port AL	PAL	R/W			×	Undefined
FFFFF000H	Port ALL	PALL	R/W	×	×		Undefined
FFFFF001H	Port ALH	PALH	R/W	×	×		Undefined
FFFFF002H	Port AH	PAH	R/W			×	Undefined
FFFFF002H	Port AHL	PAHL	R/W	×	×		Undefined
FFFFF003H	Port AHH	PAHH	R/W	×	×		Undefined
FFFFF004H	Port DL	PDL	R/W			×	Undefined
FFFFF004H	Port DLL	PDLL	R/W	×	×		Undefined
FFFFF005H	Port DLH	PDLH	R/W	×	×		Undefined
FFFFF008H	Port CS	PCS	R/W	×	×		Undefined
FFFFF00AH	Port CT	PCT	R/W	×	×		Undefined
FFFFF00CH	Port CM	PCM	R/W	×	×		Undefined
FFFFF020H	Port AL mode register	PMAL	R/W			×	FFFFH
FFFFF020H	Port AL mode register L	PMALL	R/W	×	×		FFH
FFFFF021H	Port AL mode register H	PMALH	R/W	×	×		FFH
FFFFF022H	Port AH mode register	PMAH	R/W			×	FFFFH
FFFFF022H	Port AH mode register L	PMAHL	R/W	×	×		FFH
FFFFF023H	Port AH mode register H	PMAHH	R/W	×	×		FFH
FFFFF024H	Port DL mode register	PMDL	R/W			×	FFFFH
FFFFF024H	Port DL mode register L	PMDLL	R/W	×	×		FFH
FFFFF025H	Port DL mode register H	PMDLH	R/W	×	×		FFH
FFFFF028H	Port CS mode register	PMCS	R/W	×	×		FFH
FFFFF02AH	Port CT mode register	PMCT	R/W	×	×		FFH
FFFFF02CH	Port CM mode register	PMCM	R/W	×	×		FFH
FFFFF040H	Port AL mode control register	PMCAL	R/W			×	0000H
FFFFF040H	Port AL mode control register L	PMCALL	R/W	×	×		00H
FFFFF041H	Port AL mode control register H	PMCALH	R/W	×	×		00H
FFFFF042H	Port AH mode control register	PMCAH	R/W			×	0000H
FFFFF042H	Port AH mode control register L	PMCAHL	R/W	×	×		00H
FFFFF043H	Port AH mode control register H	PMCAHH	R/W	×	×		00H
FFFFF044H	Port DL mode control register	PMCDL	R/W			×	0000H
FFFFF044H	Port DL mode control register L	PMCDLL	R/W	×	×		00H
FFFFF045H	Port DL mode control register H	PMCDLH	R/W	×	×		00H
FFFFF048H	Port CS mode control register	PMCCS	R/W	×	×		00H
FFFFF04AH	Port CT mode control register	PMCCT	R/W	×	×		00H
FFFFF04CH	Port CM mode control register	PMCCM	R/W	×	×		00H
FFFFF060H	Chip area select control register 0	CSC0	R/W			×	2C11H
FFFFF062H	Chip area select control register 1	CSC1	R/W			×	2C11H
FFFFF064H	Peripheral area select control register	BPC	R/W			×	0FFFH
FFFFF066H	Bus size configuration register	BSC	R/W			×	5555H
FFFFF068H	Endian configuration register	BEC	R/W			×	0000H

Table 3-4: List of Peripheral I/O Registers (Sheet 2 of 10)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
FFFFF06EH	System wait control register	VSWC	R/W	×	×		77H
FFFFF080H	DMA source address register L0	DSAL0	R/W			×	Undefined
FFFFF082H	DMA source address register H0	DSAH0	R/W			×	Undefined
FFFFF084H	DMA destination address register L0	DDAL0	R/W			×	Undefined
FFFFF086H	DMA destination address register H0	DDAH0	R/W			×	Undefined
FFFFF088H	DMA source address register L1	DSAL1	R/W			×	Undefined
FFFFF08AH	DMA source address register H1	DSAH1	R/W			×	Undefined
FFFFF08CH	DMA destination address register L1	DDAL1	R/W			×	Undefined
FFFFF08EH	DMA destination address register H1	DDAH1	R/W			×	Undefined
FFFFF090H	DMA source address register L2	DSAL2	R/W			×	Undefined
FFFFF092H	DMA source address register H2	DSAH2	R/W			×	Undefined
FFFFF094H	DMA destination address register L2	DDAL2	R/W			×	Undefined
FFFFF096H	DMA destination address register H2	DDAH2	R/W			×	Undefined
FFFFF098H	DMA source address register L3	DSAL3	R/W			×	Undefined
FFFFF09AH	DMA source address register H3	DSAH3	R/W			×	Undefined
FFFFF09CH	DMA destination address register L3	DDAL3	R/W			×	Undefined
FFFFF09EH	DMA destination address register H3	DDAH3	R/W			×	Undefined
FFFFF0C0H	DMA transfer count register 0	DBC0	R/W			×	Undefined
FFFFF0C2H	DMA transfer count register 1	DBC1	R/W			×	Undefined
FFFFF0C4H	DMA transfer count register 2	DBC2	R/W			×	Undefined
FFFFF0C6H	DMA transfer count register 3	DBC3	R/W			×	Undefined
FFFFF0D0H	DMA addressing control register 0	DADC0	R/W			×	0000H
FFFFF0D2H	DMA addressing control register 1	DADC1	R/W			×	0000H
FFFFF0D4H	DMA addressing control register 2	DADC2	R/W			×	0000H
FFFFF0D6H	DMA addressing control register 3	DADC3	R/W			×	0000H
FFFFF0E0H	DMA channel control register 0	DCHC0	R/W	×	×		00H
FFFFF0E2H	DMA channel control register 1	DCHC1	R/W	×	×		00H
FFFFF0E4H	DMA channel control register 2	DCHC2	R/W	×	×		00H
FFFFF0E6H	DMA channel control register 3	DCHC3	R/W	×	×		00H
FFFFF0F0H	DMA disable status register	DDIS	R	×	×		00H
FFFFF0F2H	DMA restart register	DRST	R/W	×	×		00H
FFFFF100H	Interrupt mask register 0	IMR0	R/W			×	FFFFH
FFFFF100H	Interrupt mask register 0L	IMR0L	R/W	×	×		FFH
FFFFF101H	Interrupt mask register 0H	IMR0H	R/W	×	×		FFH
FFFFF102H	Interrupt mask register 1	IMR1	R/W	×	×	×	FFFFH
FFFFF102H	Interrupt mask register 1L	IMR1L	R/W	×	×		FFH
FFFFF103H	Interrupt mask register 1H	IMR1H	R/W	×	×		FFH
FFFFF104H	Interrupt mask register 2	IMR2	R/W	×	×	×	FFFFH
FFFFF104H	Interrupt mask register 2L	IMR2L	R/W	×	×		FFH
FFFFF105H	Interrupt mask register 2H	IMR2H	R/W	×	×		FFH
FFFFF106H	Interrupt mask register 3	IMR3	R/W		×	×	FFFFH
FFFFF106H	Interrupt mask register 3L	IMR3L	R/W	×	×		FFH
FFFFF107H	Interrupt mask register 3H	IMR3H	R/W	×	×		FFH

Table 3-4: List of Peripheral I/O Registers (Sheet 3 of 10)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
FFFFF110H	Interrupt control register	PIC0	R/W	×	×		47H
FFFFF112H	Interrupt control register	PIC1	R/W	×	×		47H
FFFFF114H	Interrupt control register	PIC2	R/W	×	×		47H
FFFFF116H	Interrupt control register	PIC3	R/W	×	×		47H
FFFFF118H	Interrupt control register	PIC4	R/W	×	×		47H
FFFFF11AH	Interrupt control register	PIC5	R/W	×	×		47H
FFFFF11CH	Interrupt control register	PIC6	R/W	×	×		47H
FFFFF11EH	Interrupt control register	PIC7	R/W	×	×		47H
FFFFF120H	Interrupt control register	PIC8	R/W	×	×		47H
FFFFF122H	Interrupt control register	PIC9	R/W	×	×		47H
FFFFF124H	Interrupt control register	PIC10	R/W	×	×		47H
FFFFF126H	Interrupt control register	PIC11	R/W	×	×		47H
FFFFF128H	Interrupt control register	PIC12	R/W	×	×		47H
FFFFF12AH	Interrupt control register	PIC13	R/W	×	×		47H
FFFFF12CH	Interrupt control register	PIC14	R/W	×	×		47H
FFFFF12EH	Interrupt control register	PIC15	R/W	×	×		47H
FFFFF130H	Interrupt control register	PIC16	R/W	×	×		47H
FFFFF132H	Interrupt control register	PIC17	R/W	×	×		47H
FFFFF134H	Interrupt control register	PIC18	R/W	×	×		47H
FFFFF136H	Interrupt control register	PIC19	R/W	×	×		47H
FFFFF138H	Interrupt control register	PIC20	R/W	×	×		47H
FFFFF13AH	Interrupt control register	PIC21	R/W	×	×		47H
FFFFF13CH	Interrupt control register	PIC22	R/W	×	×		47H
FFFFF13EH	Interrupt control register	PIC23	R/W	×	×		47H
FFFFF140H	Interrupt control register	PIC24	R/W	×	×		47H
FFFFF142H	Interrupt control register	PIC25	R/W	×	×		47H
FFFFF144H	Interrupt control register	PIC26	R/W	×	×		47H
FFFFF146H	Interrupt control register	PIC27	R/W	×	×		47H
FFFFF148H	Interrupt control register	PIC28	R/W	×	×		47H
FFFFF14AH	Interrupt control register	PIC29	R/W	×	×		47H
FFFFF14CH	Interrupt control register	PIC30	R/W	×	×		47H
FFFFF14EH	Interrupt control register	PIC31	R/W	×	×		47H
FFFFF150H	Interrupt control register	PIC32	R/W	×	×		47H
FFFFF152H	Interrupt control register	PIC33	R/W	×	×		47H
FFFFF154H	Interrupt control register	PIC34	R/W	×	×		47H
FFFFF156H	Interrupt control register	PIC35	R/W	×	×		47H
FFFFF158H	Interrupt control register	PIC36	R/W	×	×		47H
FFFFF15AH	Interrupt control register	PIC37	R/W	×	×		47H
FFFFF15CH	Interrupt control register	PIC38	R/W	×	×		47H
FFFFF15EH	Interrupt control register	PIC39	R/W	×	×		47H
FFFFF160H	Interrupt control register	PIC40	R/W	×	×		47H
FFFFF162H	Interrupt control register	PIC41	R/W	×	×		47H
FFFFF164H	Interrupt control register	PIC42	R/W	×	×		47H

Table 3-4: List of Peripheral I/O Registers (Sheet 4 of 10)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
FFFFF166H	Interrupt control register	PIC43	R/W	×	×		47H
FFFFF168H	Interrupt control register	PIC44	R/W	×	×		47H
FFFFF16AH	Interrupt control register	PIC45	R/W	×	×		47H
FFFFF16CH	Interrupt control register	PIC46	R/W	×	×		47H
FFFFF16EH	Interrupt control register	PIC47	R/W	×	×		47H
FFFFF170H	Interrupt control register	PIC48	R/W	×	×		47H
FFFFF172H	Interrupt control register	PIC49	R/W	×	×		47H
FFFFF174H	Interrupt control register	PIC50	R/W	×	×		47H
FFFFF176H	Interrupt control register	PIC51	R/W	×	×		47H
FFFFF178H	Interrupt control register	PIC52	R/W	×	×		47H
FFFFF17AH	Interrupt control register	PIC53	R/W	×	×		47H
FFFFF17CH	Interrupt control register	PIC54	R/W	×	×		47H
FFFFF17EH	Interrupt control register	PIC55	R/W	×	×		47H
FFFFF180H	Interrupt control register	PIC56	R/W	×	×		47H
FFFFF182H	Interrupt control register	PIC57	R/W	×	×		47H
FFFFF184H	Interrupt control register	PIC58	R/W	×	×		47H
FFFFF186H	Interrupt control register	PIC59	R/W	×	×		47H
FFFFF1FAH	In-service priority register	ISPR	R	×	×		00H
FFFFF1FCH	Command register	PRCMD	W		×		Undefined
FFFFF1FEH	Power save control register	PSC	R/W	×	×		00H
FFFFF200H	A/D converter with scan mode register 0	ADSCM0	R/W	×	×	×	0000H
FFFFF202H	A/D converter with scan mode register 1	ADSCM1	R/W	×	×	×	0000H
FFFFF204H	A/D Voltage Detect mode register	ADETM	R/W	×	×	×	0000H
FFFFF210H	A/D conversion result register 0	ADCR0	R			×	Undefined
FFFFF212H	A/D conversion result register 1	ADCR1	R			×	Undefined
FFFFF214H	A/D conversion result register 2	ADCR2	R			×	Undefined
FFFFF216H	A/D conversion result register 3	ADCR3	R			×	Undefined
FFFFF218H	A/D conversion result register 4	ADCR4	R			×	Undefined
FFFFF21AH	A/D conversion result register 5	ADCR5	R			×	Undefined
FFFFF21CH	A/D conversion result register 6	ADCR6	R			×	Undefined
FFFFF21EH	A/D conversion result register 7	ADCR7	R			×	Undefined
FFFFF220H	A/D conversion result register 8	ADCR8	R			×	Undefined
FFFFF222H	A/D conversion result register 9	ADCR9	R			×	Undefined
FFFFF224H	A/D conversion result register 10	ADCR10	R			×	Undefined
FFFFF226H	A/D conversion result register 11	ADCR11	R			×	Undefined
FFFFF400H	Port1	P1	R/W	×	×		Undefined
FFFFF402H	Port2	P2	R/W	×	×		Undefined
FFFFF404H	Port3	P3	R/W	×	×		Undefined
FFFFF406H	Port4	P4	R/W	×	×		Undefined
FFFFF408H	Port5	P5	R/W	×	×		Undefined
FFFFF40AH	Port6	P6	R/W	×	×		Undefined

Table 3-4: List of Peripheral I/O Registers (Sheet 5 of 10)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
FFFFF420H	Port1 mode	PM1	R/W	×	×		FFH
FFFFF422H	Port2 mode	PM2	R/W	×	×		FFH
FFFFF424H	Port3 mode	PM3	R/W	×	×		FFH
FFFFF426H	Port4 mode	PM4	R/W	×	×		FFH
FFFFF428H	Port5 mode	PM5	R/W	×	×		FFH
FFFFF42AH	Port6 mode	PM6	R/W	×	×		FFH
FFFFF440H	Port1 mode control	PMC1	R/W	×	×		00H
FFFFF442H	Port2 mode control	PMC2	R/W	×	×		00H
FFFFF444H	Port3 mode control	PMC3	R/W	×	×		00H
FFFFF446H	Port4 mode control	PMC4	R/W	×	×		00H
FFFFF448H	Port5 mode control	PMC5	R/W	×	×		00H
FFFFF44AH	Port6 mode control	PMC6	R/W	×	×		00H
FFFFF480H	Bus Cycle Type Control register 0	BCT0	R/W			×	8888H
FFFFF482H	Bus Cycle Type Control register 1	BCT1	R/W			×	8888H
FFFFF484H	Data Wait Control register 0	DWC0	R/W			×	7777H
FFFFF486H	Data Wait Control register 1	DWC1	R/W			×	7777H
FFFFF488H	Bus Cycle Control register	BCC	R/W			×	FFFFH
FFFFF48AH	Address Setup Wait Control register	ASC	R/W			×	FFFFH
FFFFF49AH	Page-ROM Configuration register	PRC				×	7000H
FFFFF540H	Timer D0 counter	TMD0	R			×	0000H
FFFFF542H	Timer D0 compare register	CMD0	R/W			×	0000H
FFFFF544H	Timer D0 Control register	TMCD0	R/W	×	×		0000H
FFFFF550H	Timer D1 counter	TMD1	R			×	0000H
FFFFF552H	Timer D1 compare register	CMD1	R/W			×	0000H
FFFFF554H	Timer D1 Control register	TMCD1	R/W	×	×		0000H
FFFFF560H	Watch timer mode register	WTM		×	×		0000H
FFFFF570H	Watch dog timer mode register	WDTM		×	×		0000H
FFFFF640H	Timer Macro Clock Stop Register	STOPTE0	R/W	×	×	×	0000H
FFFFF642H	Count Clock / Control Edge Selection Register	CSE0	R/W	×	×	×	0000H
FFFFF644H	Subchannel Input Event Edge Selection Register	SESE0	R/W	×	×	×	0000H
FFFFF646H	Timebases Control Register	TCRE0	R/W	×	×	×	0000H
FFFFF648H	Output Control Register	OCTLE0	R/W	×	×	×	0000H
FFFFF64AH	Capture/Compare Control Register of Subchannels 0 and 5	CMSE050	R/W			×	0000H
FFFFF64CH	Capture/Compare Control Register of Subchannels 1 and 2	CMSE120	R/W			×	0000H
FFFFF64EH	Capture/Compare Control Register of Subchannels 3 and 4	CMSE340	R/W			×	0000H
FFFFF650H	Secondary Capture/Compare Register of Subchannel 1	CVSE10	R/W			×	0000H
FFFFF652H	Primary Capture/Compare Register of Subchannel 1	CVPE10	R			×	0000H

Table 3-4: List of Peripheral I/O Registers (Sheet 6 of 10)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
FFFFF654H	Secondary Capture/Compare Register of Subchannel 2	CVSE20	R/W			×	0000H
FFFFF656H	Primary Capture/Compare Register of Subchannel 2	CVPE20	R			×	0000H
FFFFF658H	Secondary Capture/Compare Register of Subchannel 3	CVSE30	R/W			×	0000H
FFFFF65AH	Primary Capture/Compare Register of Subchannel 3	CVPE30	R			×	0000H
FFFFF65CH	Secondary Capture/Compare Register of Subchannel 4	CVSE40	R/W			×	0000H
FFFFF65EH	Primary Capture/Compare Register of Subchannel 4	CVPE40	R			×	0000H
FFFFF660H	Capture/Compare Register of Subchannel 0	CVSE00	R/W			×	0000H
FFFFF662H	Capture/Compare Register of Subchannel 5	CVSE50	R/W			×	0000H
FFFFF664H	Timebase Status Register	TBSTATE0	R/(W)			×	0000H
FFFFF666H	Capture/Compare Subchannels 1-4 Status Register	CCSTATE0	R/(W)			×	0000H
FFFFF668H	Output Delay Register	ODELE0	R/W			×	0000H
FFFFF66AH	Software Event Capture Register	CSCE0	R/W			×	0000H
FFFFF670H	Counter value of timebasecouter_0	TBASE00	R			×	0000H
FFFFF672H	Counter value of timebasecouter_1	TBASE10	R			×	0000H
FFFFF680H	Timer Macro Clock Stop Register	STOPTE1	R/W	×	×	×	0000H
FFFFF682H	Count Clock / Control Edge Selection Register	CSE1	R/W	×	×	×	0000H
FFFFF684H	Subchannel Input Event Edge Selection Register	SESE1	R/W	×	×	×	0000H
FFFFF686H	Timebases Control Register	TCRE1	R/W	×	×	×	0000H
FFFFF688H	Output Control Register	OCTLE1	R/W	×	×	×	0000H
FFFFF68AH	Capture/Compare Control Register of Subchannels 0 and 5	CMSE051	R/W			×	0000H
FFFFF68CH	Capture/Compare Control Register of Subchannels 1 and 2	CMSE121	R/W			×	0000H
FFFFF68EH	Capture/Compare Control Register of Subchannels 3 and 4	CMSE341	R/W			×	0000H
FFFFF690H	Secondary Capture/Compare Register of Subchannel 1	CVSE11	R/W			×	0000H
FFFFF692H	Primary Capture/Compare Register of Subchannel 1	CVPE11	R			×	0000H
FFFFF694H	Secondary Capture/Compare Register of Subchannel 2	CVSE21	R/W			×	0000H
FFFFF696H	Primary Capture/Compare Register of Subchannel 2	CVPE21	R			×	0000H
FFFFF698H	Secondary Capture/Compare Register of Subchannel 3	CVSE31	R/W			×	0000H
FFFFF69AH	Primary Capture/Compare Register of Subchannel 3	CVPE31	R			×	0000H

Table 3-4: List of Peripheral I/O Registers (Sheet 7 of 10)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
FFFFF69CH	Secondary Capture/Compare Register of Subchannel 4	CVSE41	R/W			×	0000H
FFFFF69EH	Primary Capture/Compare Register of Subchannel 4	CVPE41	R			×	0000H
FFFFF6A0H	Capture/Compare Register of Subchannel 0	CVSE01	R/W			×	0000H
FFFFF6A2H	Capture/Compare Register of Subchannel 5	CVSE51	R/W			×	0000H
FFFFF6A4H	Timebase Status Register	TBSTATE1	R/(W)			×	0000H
FFFFF6A6H	Capture/Compare Subchannels 1-4 Status Register	CCSTATE1	R/(W)			×	0000H
FFFFF6A8H	Output Delay Register	ODELE1	R/W			×	0000H
FFFFF6AAH	Software Event Capture Register	CSCE1	R/W			×	0000H
FFFFF6B0H	Counter value of timebasecouter_0	TBASE01	R			×	0000H
FFFFF6B2H	Counter value of timebasecouter_1	TBASE11	R			×	0000H
FFFFF6C0H	Timer Macro Clock Stop Register	STOPTE2	R/W	×	×	×	0000H
FFFFF6C2H	Count Clock / Control Edge Selection Register	CSE2	R/W	×	×	×	0000H
FFFFF6C4H	Subchannel Input Event Edge Selection Register	SESE2	R/W	×	×	×	0000H
FFFFF6C6H	Timebases Control Register	TCRE2	R/W	×	×	×	0000H
FFFFF6C8H	Output Control Register	OCTLE2	R/W	×	×	×	0000H
FFFFF6CAH	Capture/Compare Control Register of Subchannels 0 and 5	CMSE052	R/W			×	0000H
FFFFF6CCH	Capture/Compare Control Register of Subchannels 1 and 2	CMSE122	R/W			×	0000H
FFFFF6CEH	Capture/Compare Control Register of Subchannels 3 and 4	CMSE342	R/W			×	0000H
FFFFF6D0H	Secondary Capture/Compare Register of Subchannel 1	CVSE12	R/W			×	0000H
FFFFF6D2H	Primary Capture/Compare Register of Subchannel 1	CVPE12	R			×	0000H
FFFFF6D4H	Secondary Capture/Compare Register of Subchannel 2	CVSE22	R/W			×	0000H
FFFFF6D6H	Primary Capture/Compare Register of Subchannel 2	CVPE22	R			×	0000H
FFFFF6D8H	Secondary Capture/Compare Register of Subchannel 3	CVSE32	R/W			×	0000H
FFFFF6DAH	Primary Capture/Compare Register of Subchannel 3	CVPE32	R			×	0000H
FFFFF6DCH	Secondary Capture/Compare Register of Subchannel 4	CVSE42	R/W			×	0000H
FFFFF6DEH	Primary Capture/Compare Register of Subchannel 4	CVPE42	R			×	0000H
FFFFF6E0H	Capture/Compare Register of Subchannel 0	CVSE02	R/W			×	0000H
FFFFF6E2H	Capture/Compare Register of Subchannel 5	CVSE52	R/W			×	0000H

Table 3-4: List of Peripheral I/O Registers (Sheet 8 of 10)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
FFFFF6E4H	Timebase Status Register	TBSTATE2	R/(W)			×	0000H
FFFFF6E6H	Capture/Compare Subchannels 1-4 Status Register	CCSTATE2	R/(W)			×	0000H
FFFFF6E8H	Output Delay Register	ODELE2	R/W			×	0000H
FFFFF6EAH	Software Event Capture Register	CSCE2	R/W			×	0000H
FFFFF6ECH	Counter value of timebasecouter_0	TBASE02	R			×	0000H
FFFFF6EEH	Counter value of timebasecouter_1	TBASE12	R			×	0000H
FFFFF700H	PORT / LCD selector 0	LSEG0	R/W	×	×		00H
FFFFF702H	PORT / LCD selector 1	LSEG1	R/W	×	×		00H
FFFFF704H	PORT / LCD selector 2	LSEG2	R/W	×	×		00H
FFFFF706H	PORT / LCD selector 3	LSEG3	R/W	×	×		00H
FFFFF708H	PORT / LCD selector 4	LSEG4	R/W	×	×		00H
FFFFF710H	LCD display mode register	LCDM	R/W	×	×	×	0000H
FFFFF720H	LCD segment register00	SEGREG00	R/W	×	×	×	0000H
FFFFF722H	LCD segment register10	SEGREG10	R/W	×	×	×	0000H
FFFFF724H	LCD segment register20	SEGREG20	R/W	×	×	×	0000H
FFFFF726H	LCD segment register30	SEGREG30	R/W	×	×	×	0000H
FFFFF730H	LCD segment register01	SEGREG01	R/W	×	×	×	0000H
FFFFF732H	LCD segment register11	SEGREG11	R/W	×	×	×	0000H
FFFFF734H	LCD segment register21	SEGREG21	R/W	×	×	×	0000H
FFFFF736H	LCD segment register31	SEGREG31	R/W	×	×	×	0000H
FFFFF740H	LCD segment register02	SEGREG02	R/W	×	×	×	0000H
FFFFF742H	LCD segment register12	SEGREG12	R/W	×	×	×	0000H
FFFFF744H	LCD segment register22	SEGREG22	R/W	×	×	×	0000H
FFFFF746H	LCD segment register32	SEGREG32	R/W	×	×	×	0000H
FFFFF800H	Peripheral command register	PHCMD	W		×		Undefined
FFFFF802H	Peripheral status register	PHS	R/W	×	×		00H
FFFFF810H	DMA trigger source select register 0	DTFR0	R/W	×	×		00H
FFFFF812H	DMA trigger source select register 1	DTFR1	R/W	×	×		00H
FFFFF814H	DMA trigger source select register 2	DTFR2	R/W	×	×		00H
FFFFF816H	DMA trigger source select register 3	DTFR3	R/W	×	×		00H
FFFFF820H	Power save mode register	PSM	R/W	×	×		00H
FFFFF822H	Clock control register	CKC	R/W	×	×		00H
FFFFF824H	PLL status register	PSTAT	R	×	×		0xH
FFFFF860H	Voltage comparator mode	VCMPM					00H
FFFFF880H	Filter edge mode channel 00	FEM00	R/W	×	×	×	
FFFFF881H	Filter edge mode channel 10	FEM10	R/W	×	×		
FFFFF882H	Filter edge mode channel 20	FEM20	R/W	×	×	×	
FFFFF883H	Filter edge mode channel 30	FEM30	R/W	×	×		
FFFFF884H	Filter edge mode channel 40	FEM40	R/W	×	×	×	
FFFFF885H	Filter edge mode channel 50	FEM50	R/W	×	×		
FFFFF890H	Filter edge mode channel 01	FEM01	R/W	×	×	×	
FFFFF891H	Filter edge mode channel 11	FEM11	R/W	×	×		

Table 3-4: List of Peripheral I/O Registers (Sheet 9 of 10)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
FFFFF892H	Filter edge mode channel 21	FEM21	R/W	×	×	×	
FFFFF893H	Filter edge mode channel 31	FEM31	R/W	×	×		
FFFFF894H	Filter edge mode channel 41	FEM41	R/W	×	×	×	
FFFFF895H	Filter edge mode channel 51	FEM51	R/W	×	×		
FFFFF8A0H	Filter edge mode channel 02	FEM02	R/W	×	×	×	
FFFFF8A1H	Filter edge mode channel 12	FEM12	R/W	×	×		
FFFFF8A2H	Filter edge mode channel 22	FEM22	R/W	×	×	×	
FFFFF8A3H	Filter edge mode channel 32	FEM32	R/W	×	×		
FFFFF8A4H	Filter edge mode channel 42	FEM42	R/W	×	×	×	
FFFFF8A5H	Filter edge mode channel 52	FEM52	R/W	×	×		
FFFFF8B0H	Filter edge mode channel 03	FEM03	R/W	×	×	×	
FFFFF8B1H	Filter edge mode channel 13	FEM13	R/W	×	×		
FFFFF8B2H	Filter edge mode channel 23	FEM23	R/W	×	×	×	
FFFFF900H	CSI operation mode register	CSIM0	R/W	×	×	×	00H
FFFFF901H	Clock selection register	CSIC0	R/W	×	×		00H
FFFFF902H	Reception data buffer register	SIRB0/	R		×	×	0000H
		SIRBL0	R		×		00H
FFFFF904H	Transmission data buffer register	SOTB0/	R/W		×	×	0000H
		SOTBL0	R/W		×		00H
FFFFF906H	Reception data buffer register for emulation read	SIRBE0/	R		×	×	0000H
		SIRBEL0	R		×		00H
FFFFF908H	First transmission data buffer register	SOTBF0/	R/W		×	×	0000H
		SOTBFL0	R/W		×		00H
FFFFF90AH	Shift register	SIO0/	R/W		×	×	0000H
		SIOL0	R/W		×		00H
FFFFF910H	CSI operation mode register	CSIM1	R/W	×	×	×	00H
FFFFF911H	Clock selection register	CSIC1	R/W	×	×		00H
FFFFF912H	Reception data buffer register	SIRB1/	R		×	×	0000H
		SIRBL1	R		×		00H
FFFFF914H	Transmission data buffer register	SOTB1/	R/W		×	×	0000H
		SOTBL1	R/W		×		00H
FFFFF916H	Reception data buffer register for emulation read	SIRBE1/	R		×	×	0000H
		SIRBEL1	R		×		00H
FFFFF918H	First transmission data buffer register	SOTBF1/	R/W		×	×	0000H
		SOTBFL1	R/W		×		00H
FFFFF91AH	Shift register	SIO1/	R/W		×	×	0000H
		SIOL1	R/W		×		00H
FFFFF920H	BRG0 pre-scalar mode register 0	PRSM0	R/W	×	×		00H
FFFFF922H	Pre-scalar compare register 0	PRSCM0	R/W	×	×		00H
FFFFF930H	BRG1 pre-scalar mode register1	PRSM1	R/W	×	×		00H
FFFFF932H	Pre-scalar compare register1	PRSCM1	R/W	×	×		00H
FFFFFA00H	UART operation mode register	ASIM0	R/W	×	×		00H
FFFFFA02H	Reception buffer register	RXB0	R		×	×	FFH

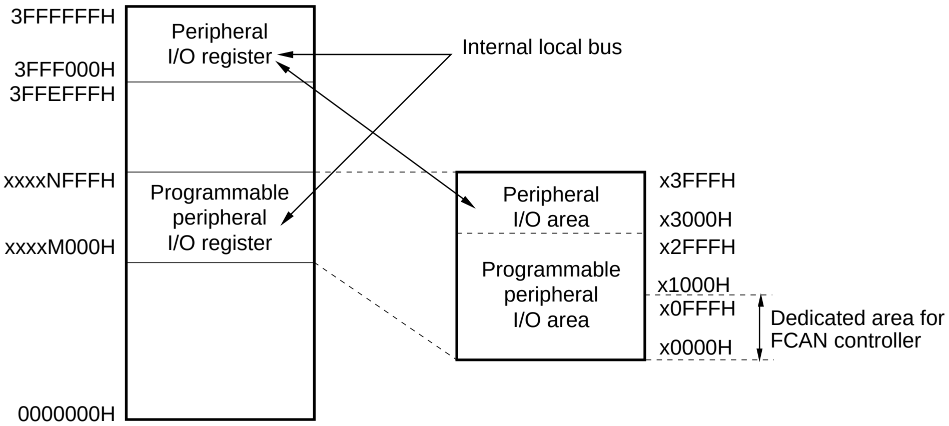
Table 3-4: List of Peripheral I/O Registers (Sheet 10 of 10)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
FFFFFA03H	UART reception error status register	ASIS0	R		×		00H
FFFFFA04H	Transmission buffer register	TXB0	R/W		×	×	FFH
FFFFFA05H	UART transmission error status register	ASIF0	R		×		00H
FFFFFA06H	Clock selection register	CHKSR0	R/W	×	×	×	FFH
FFFFFA07H	Baudrate definition register	BRGC0	R/W	×	×		00H
FFFFFA10H	UART operation mode register	ASIM1	R/W	×	×		00H
FFFFFA12H	Reception buffer register	RXB1	R		×	×	FFH
FFFFFA13H	UART reception error status register	ASIS1	R		×		00H
FFFFFA14H	Transmission buffer register	TXB1	R/W		×	×	00H
FFFFFA15H	UART transmission error status register	ASIF1	R		×		FFH
FFFFFA16H	Clock selection register	CHKSR1	R/W	×	×	×	00H
FFFFFA17H	Baudrate definition register	BRGC1	R/W	×	×		FFH
FFFFFA20H	UART operation mode register	ASIM2	R/W	×	×		00H
FFFFFA22H	Reception buffer register	RXB2	R		×	×	FFH
FFFFFA23H	UART reception error status register	ASIS2	R		×		00H
FFFFFA24H	Transmission buffer register	TXB2	R/W		×	×	FFH
FFFFFA25H	UART transmission error status register	ASIF2	R		×		00H
FFFFFA26H	Clock selection register	CHKSR2	R/W	×	×	×	00H
FFFFFA27H	Baudrate definition register	BRGC2	R/W	×	×		FFH

3.4.9 Programmable peripheral I/O registers

In the V850E/CA1, the 16 KB area of x0000H to x3FFFH is provided as a programmable peripheral I/O area. In this area, the area between x0000H and x0FFFH is used exclusively for the FCAN controller. The internal bus of the V850E/CA1 becomes active when the peripheral I/O register area (FFFF000H to FFFFFFFH) or the programmable peripheral I/O register area (xxxxm000H to xxxxnFFFH) is accessed (m = xx00B, n = xx11B). Note that when data is written to the peripheral I/O register area, the written contents is reflected on the peripheral I/O register since peripheral I/O register area is allocated to the last 4 KB of the programmable peripheral I/O register area.

Figure 3-14: Programmable Peripheral I/O Register (Outline)



Cautions: 1. The CAN message buffer register can allocate address xxxx freely as a programmable peripheral I/O register. but once the address xxxx is set, it cannot be changed.

2. If the programmable peripheral I/O area overlaps the following areas, the programmable peripheral I/O area becomes ineffective.

- **Peripheral I/O area**
- **ROM area**
- **RAM area**

Remark: M = xx00B
N = xx11B

(1) Peripheral area selection control register (BPC)

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

Figure 3-15: Peripheral Area Selection Control Register (BPC)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
BPC	PA15	0	PA13	PA12	PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	FFFFF064H	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15</td><td>PA15</td><td>Enables/disables usage of programmable peripheral I/O area
<table>
<tr><th>PA15</th><th>Usage of Programmable Peripheral I/O Area</th></tr>
<tr><td>0</td><td>Disables usage of programmable peripheral I/O area</td></tr>
<tr><td>1</td><td>Enables usage of programmable peripheral I/O area</td></tr>
</table></td></tr>
<tr><td>13 to 0</td><td>PA13 to PA0</td><td>Specifies an address in programmable peripheral I/O area (corresponds to A27 to A14, respectively).</td></tr>
</table>

A list of the programmable peripheral I/O registers is shown below:

Table 3-5: List of programmable peripheral I/O registers (Sheet 1 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn000H	CAN message event pointer 000	M_EVT000	R/W		x		Undefined
xxxxn001H	CAN message event pointer 001	M_EVT001	R/W		x		Undefined
xxxxn002H	CAN message event pointer 002	M_EVT002	R/W		x		Undefined
xxxxn003H	CAN message event pointer 003	M_EVT003	R/W		x		Undefined
xxxxn004H	CAN message data length register 00	M_DLC00	R/W		x		Undefined
xxxxn005H	CAN message control register 00	M_CTRL00	R/W		x		Undefined
xxxxn006H	CAN message time stamp register 00	M_TIME00	R/W			x	Undefined
xxxxn008H	CAN message data register 000	M_DATA000	R/W		x	x	Undefined
xxxxn009H	CAN message data register 001	M_DATA001	R/W		x		Undefined
xxxxn00AH	CAN message data register 002	M_DATA002	R/W		x		Undefined
xxxxn00BH	CAN message data register 003	M_DATA003	R/W		x		Undefined
xxxxn00CH	CAN message data register 004	M_DATA004	R/W		x		Undefined
xxxxn00DH	CAN message data register 005	M_DATA005	R/W		x		Undefined
xxxxn00EH	CAN message data register 006	M_DATA006	R/W		x		Undefined
xxxxn00FH	CAN message data register 007	M_DATA007	R/W		x		Undefined
xxxxn010H	CAN message ID register L00	M_IDL00	R/W			x	Undefined
xxxxn012H	CAN message ID register H00	M_IDH00	R/W			x	Undefined
xxxxn014H	CAN message configuration register 00	M_CONF00	R/W		x		Undefined
xxxxn015H	CAN message status register 00	M_STAT00	R		x		Undefined
xxxxn016H	CAN status set/cancel register 00	SC_STAT00	W			x	0000H
xxxxn020H	CAN message event pointer 010	M_EVT010	R/W		x		Undefined
xxxxn021H	CAN message event pointer 011	M_EVT011	R/W		x		Undefined
xxxxn022H	CAN message event pointer 012	M_EVT012	R/W		x		Undefined
xxxxn023H	CAN message event pointer 013	M_EVT013	R/W		x		Undefined
xxxxn024H	CAN message data length register 01	M_DLC01	R/W		x		Undefined
xxxxn025H	CAN message control register 01	M_CTRL01	R/W		x		Undefined
xxxxn026H	CAN message time stamp register 01	M_TIME01	R/W			x	Undefined
xxxxn028H	CAN message data register 010	M_DATA010	R/W		x		Undefined
xxxxn029H	CAN message data register 011	M_DATA011	R/W		x		Undefined
xxxxn02AH	CAN message data register 012	M_DATA012	R/W		x		Undefined
xxxxn02BH	CAN message data register 013	M_DATA013	R/W		x		Undefined
xxxxn02CH	CAN message data register 014	M_DATA014	R/W		x		Undefined
xxxxn02DH	CAN message data register 015	M_DATA015	R/W		x		Undefined
xxxxn02EH	CAN message data register 016	M_DATA016	R/W		x		Undefined
xxxxn02FH	CAN message data register 017	M_DATA017	R/W		x		Undefined
xxxxn030H	CAN message ID register L01	M_IDL01	R/W			x	Undefined
xxxxn032H	CAN message ID register H01	M_IDH01	R/W			x	Undefined
xxxxn034H	CAN message configuration register 01	M_CONF01	R/W		x		Undefined
xxxxn035H	CAN message status register 01	M_STAT01	R		x		Undefined
xxxxn036H	CAN status set/cancel register 01	SC_STAT01	W			x	0000H
xxxxn040H	CAN message event pointer 020	M_EVT020	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 2 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn041H	CAN message event pointer 021	M_EVT021	R/W		x		Undefined
xxxxn042H	CAN message event pointer 022	M_EVT022	R/W		x		Undefined
xxxxn043H	CAN message event pointer 023	M_EVT023	R/W		x		Undefined
xxxxn044H	CAN message data length register 02	M_DLC02	R/W		x		Undefined
xxxxn045H	CAN message control register 02	M_CTRL02	R/W		x		Undefined
xxxxn046H	CAN message time stamp register 02	M_TIME02	R/W			x	Undefined
xxxxn048H	CAN message data register 020	M_DATA020	R/W		x		Undefined
Xxxxn049H	CAN message data register 021	M_DATA021	R/W		x		Undefined
xxxxn04AH	CAN message data register 022	M_DATA022	R/W		x		Undefined
Xxxxn04BH	CAN message data register 023	M_DATA023	R/W		x		Undefined
xxxxn04CH	CAN message data register 024	M_DATA024	R/W		x		Undefined
xxxxn04DH	CAN message data register 025	M_DATA025	R/W		x		Undefined
xxxxn04EH	CAN message data register 026	M_DATA026	R/W		x		Undefined
xxxxn04FH	CAN message data register 027	M_DATA027	R/W		x		Undefined
xxxxn050H	CAN message ID register L02	M_IDL02	R/W			x	Undefined
xxxxn052H	CAN message ID register H02	M_IDH02	R/W			x	Undefined
xxxxn054H	CAN message configuration register 02	M_CONF02	R/W		x		Undefined
xxxxn055H	CAN message status register 02	M_STAT02	R		x		Undefined
xxxxn056H	CAN status set/cancel register 02	SC_STAT02	W			x	0000H
xxxxn060H	CAN message event pointer 030	M_EVT030	R/W		x		Undefined
xxxxn061H	CAN message event pointer 031	M_EVT031	R/W		x		Undefined
xxxxn062H	CAN message event pointer 032	M_EVT032	R/W		x		Undefined
xxxxn063H	CAN message event pointer 033	M_EVT033	R/W		x		Undefined
xxxxn064H	CAN message data length register 03	M_DLC03	R/W		x		Undefined
xxxxn065H	CAN message control register 03	M_CTRL03	R/W		x		Undefined
xxxxn066H	CAN message time stamp register 03	M_TIME03	R/W			x	Undefined
xxxxn068H	CAN message data register 030	M_DATA030	R/W		x		Undefined
xxxxn069H	CAN message data register 031	M_DATA031	R/W		x		Undefined
xxxxn06AH	CAN message data register 032	M_DATA032	R/W		x		Undefined
xxxxn06BH	CAN message data register 033	M_DATA033	R/W		x		Undefined
xxxxn06CH	CAN message data register 034	M_DATA034	R/W		x		Undefined
xxxxn06DH	CAN message data register 035	M_DATA035	R/W		x		Undefined
xxxxn06EH	CAN message data register 036	M_DATA036	R/W		x		Undefined
xxxxn06FH	CAN message data register 037	M_DATA037	R/W		x		Undefined
xxxxn070H	CAN message ID register L03	M_IDL03	R/W			x	Undefined
xxxxn072H	CAN message ID register H03	M_IDH03	R/W			x	Undefined
xxxxn074H	CAN message configuration register 03	M_CONF03	R/W		x		Undefined
xxxxn075H	CAN message status register 03	M_STAT03	R		x		Undefined
xxxxn076H	CAN status set/cancel register 03	SC_STAT03	W			x	0000H
xxxxn080H	CAN message event pointer 040	M_EVT040	R/W		x		Undefined
xxxxn081H	CAN message event pointer 041	M_EVT041	R/W		x		Undefined
xxxxn082H	CAN message event pointer 042	M_EVT042	R/W		x		Undefined
xxxxn083H	CAN message event pointer 043	M_EVT043	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 3 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn084H	CAN message data length register 04	M_DLC04	R/W		x		Undefined
xxxxn085H	CAN message control register 04	M_CTRL04	R/W		x		Undefined
xxxxn086H	CAN message time stamp register 04	M_TIME04	R/W			x	Undefined
xxxxn088H	CAN message data register 040	M_DATA040	R/W		x		Undefined
xxxxn089H	CAN message data register 041	M_DATA041	R/W		x		Undefined
xxxxn08AH	CAN message data register 042	M_DATA042	R/W		x		Undefined
xxxxn08BH	CAN message data register 043	M_DATA043	R/W		x		Undefined
xxxxn08CH	CAN message data register 044	M_DATA044	R/W		x		Undefined
xxxxn08DH	CAN message data register 045	M_DATA045	R/W		x		Undefined
xxxxn08EH	CAN message data register 046	M_DATA046	R/W		x		Undefined
xxxxn08FH	CAN message data register 047	M_DATA047	R/W		x		Undefined
Xxxxn090H	CAN message ID register L04	M_IDL04	R/W			x	Undefined
xxxxn092H	CAN message ID register H04	M_IDH04	R/W			x	Undefined
xxxxn094H	CAN message configuration register 04	M_CONF04	R/W		x		Undefined
xxxxn095H	CAN message status register 04	M_STAT04	R		x		Undefined
xxxxn096H	CAN status set/cancel register 04	M_STAT04	W			x	0000H
xxxxn0A0H	CAN message event pointer 050	M_EVT050	R/W		x		Undefined
xxxxn0A1H	CAN message event pointer 051	M_EVT051	R/W		x		Undefined
xxxxn0A2H	CAN message event pointer 052	M_EVT052	R/W		x		Undefined
xxxxn0A3H	CAN message event pointer 053	M_EVT053	R/W		x		Undefined
xxxxn0A4H	CAN message data length register 05	M_DLC05	R/W		x		Undefined
xxxxn0A5H	CAN message control register 05	M_DTRL05	R/W		x		Undefined
xxxxn0A6H	CAN message time stamp register 05	M_TIME05	R/W			x	Undefined
xxxxn0A8H	CAN message data register 050	M_DATA050	R/W		x		Undefined
xxxxn0A9H	CAN message data register 051	M_DATA051	R/W		x		Undefined
xxxxn0AAH	CAN message data register 052	M_DATA052	R/W		x		Undefined
xxxxn0ABH	CAN message data register 053	M_DATA053	R/W		x		Undefined
xxxxn0ACH	CAN message data register 054	M_DATA054	R/W		x		Undefined
xxxxn0ADH	CAN message data register 055	M_DATA055	R/W		x		Undefined
xxxxn0AEH	CAN message data register 056	M_DATA056	R/W		x		Undefined
xxxxn0AFH	CAN message data register 057	M_DATA057	R/W		x		Undefined
xxxxn0B0H	CAN message ID register L05	M_IDL05	R/W			x	Undefined
xxxxn0B2H	CAN message ID register H05	M_IDH05	R/W			x	Undefined
xxxxn0B4H	CAN message configuration register 05	M_CONF05	R/W		x		Undefined
xxxxn0B5H	CAN message status register 05	M_STAT05	R		x		Undefined
xxxxn0B6H	CAN status set/cancel register 05	SC_STAT05	W			x	0000H
xxxxn0C0H	CAN message event pointer 060	M_EVT060	R/W		x		Undefined
xxxxn0C1H	CAN message event pointer 061	M_EVT061	R/W		x		Undefined
xxxxn0C2H	CAN message event pointer 062	M_EVT062	R/W		x		Undefined
xxxxn0C3H	CAN message event pointer 063	M_EVT063	R/W		x		Undefined
xxxxn0C4H	CAN message data length register 06	M_DLC06	R/W		x		Undefined
xxxxn0C5H	CAN message control register 06	M_DTRL06	R/W		x		Undefined
xxxxn0C6H	CAN message time stamp register 06	M_TIME06	R/W			x	Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 4 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn0C8H	CAN message data register 060	M_DATA060	R/W		x		Undefined
xxxxn0C9H	CAN message data register 061	M_DATA061	R/W		x		Undefined
xxxxn0CAH	CAN message data register 062	M_DATA062	R/W		x		Undefined
xxxxn0CBH	CAN message data register 063	M_DATA063	R/W		x		Undefined
xxxxn0CCH	CAN message data register 064	M_DATA064	R/W		x		Undefined
xxxxn0CDH	CAN message data register 065	M_DATA065	R/W		x		Undefined
xxxxn0CEH	CAN message data register 066	M_DATA066	R/W		x		Undefined
xxxxn0CFH	CAN message data register 067	M_DATA067	R/W		x		Undefined
xxxxn0D0H	CAN message ID register L06	M_IDL06	R/W			x	Undefined
xxxxn0D2H	CAN message ID register H06	M_IDH06	R/W			x	Undefined
xxxxn0D4H	CAN message configuration register 06	M_CONF06	R/W		x		Undefined
xxxxn0D5H	CAN message status register 06	M_STAT06	R		x		Undefined
xxxxn0D6H	CAN status set/cancel register 06	SC_STAT06	W			x	0000H
xxxxn0E0H	CAN message event pointer 070	M_EVT070	R/W		x		Undefined
xxxxn0E1H	CAN message event pointer 071	M_EVT071	R/W		x		Undefined
xxxxn0E2H	CAN message event pointer 072	M_EVT072	R/W		x		Undefined
xxxxn0E3H	CAN message event pointer 073	M_EVT073	R/W		x		Undefined
xxxxn0E4H	CAN message data length register 07	M_DLC07	R/W		x		Undefined
xxxxn0E5H	CAN message control register 07	M_CTRL07	R/W		x		Undefined
xxxxn0E6H	CAN message time stamp register 07	M_TIME07	R/W			x	Undefined
xxxxn0E8H	CAN message data register 070	M_DATA070	R/W		x		Undefined
xxxxn0E9H	CAN message data register 071	M_DATA071	R/W		x		Undefined
xxxxn0EAH	CAN message data register 072	M_DATA072	R/W		x		Undefined
xxxxn0EBH	CAN message data register 073	M_DATA073	R/W		x		Undefined
xxxxn0ECH	CAN message data register 074	M_DATA074	R/W		x		Undefined
xxxxn0EDH	CAN message data register 075	M_DATA075	R/W		x		Undefined
xxxxn0EEH	CAN message data register 076	M_DATA076	R/W		x		Undefined
xxxxn0EFH	CAN message data register 077	M_DATA077	R/W		x		Undefined
xxxxn0F0H	CAN message ID register L07	M_IDL07	R/W			x	Undefined
xxxxn0F2H	CAN message ID register H07	M_IDH07	R/W			x	Undefined
xxxxn0F4H	CAN message configuration register 07	M_CONF07	R/W		x		Undefined
xxxxn0F5H	CAN message status register 07	M_STAT07	R		x		Undefined
xxxxn0F6H	CAN status set/cancel register 07	SC_STAT07	W			x	0000H
xxxxn100H	CAN message event pointer 080	M_EVT080	R/W		x		Undefined
xxxxn101H	CAN message event pointer 081	M_EVT081	R/W		x		Undefined
xxxxn102H	CAN message event pointer 082	M_EVT082	R/W		x		Undefined
xxxxn103H	CAN message event pointer 083	M_EVT083	R/W		x		Undefined
xxxxn104H	CAN message data length register 08	M_DLC08	R/W		x		Undefined
xxxxn105H	CAN message control register 08	M_CTRL08	R/W		x		Undefined
xxxxn106H	CAN message time stamp register 08	M_TIME08	R/W			x	Undefined
xxxxn108H	CAN message data register 080	M_DATA080	R/W		x		Undefined
xxxxn109H	CAN message data register 081	M_DATA081	R/W		x		Undefined
xxxxn10AH	CAN message data register 082	M_DATA082	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 5 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn10BH	CAN message data register 083	M_DATA083	R/W		x		Undefined
xxxxn10CH	CAN message data register 084	M_DATA084	R/W		x		Undefined
xxxxn10DH	CAN message data register 085	M_DATA085	R/W		x		Undefined
xxxxn10EH	CAN message data register 086	M_DATA086	R/W		x		Undefined
xxxxn10FH	CAN message data register 087	M_DATA087	R/W		x		Undefined
xxxxn110H	CAN message ID register L08	M_IDL08	R/W			x	Undefined
xxxxn112H	CAN message ID register H08	M_IDH08	R/W			x	Undefined
xxxxn114H	CAN message configuration register 08	M_CONF08	R/W		x		Undefined
xxxxn115H	CAN message status register 08	M_STAT08	R		x		Undefined
xxxxn116H	CAN status set/cancel register 08	SC_STAT08	W			x	0000H
xxxxn120H	CAN message event pointer 090	M_EVT090	R/W		x		Undefined
xxxxn121H	CAN message event pointer 091	M_EVT091	R/W		x		Undefined
xxxxn122H	CAN message event pointer 092	M_EVT092	R/W		x		Undefined
xxxxn123H	CAN message event pointer 093	M_EVT093	R/W		x		Undefined
xxxxn124H	CAN message data length register 09	M_DLC09	R/W		x		Undefined
xxxxn125H	CAN message control register 09	M_CTRL09	R/W		x		Undefined
xxxxn126H	CAN message time stamp register 09	M_TIME09	R/W			x	Undefined
xxxxn128H	CAN message data register 090	M_DATA090	R/W		x		Undefined
xxxxn129H	CAN message data register 091	M_DATA091	R/W		x		Undefined
xxxxn12AH	CAN message data register 092	M_DATA092	R/W		x		Undefined
xxxxn12BH	CAN message data register 093	M_DATA093	R/W		x		Undefined
xxxxn12CH	CAN message data register 094	M_DATA094	R/W		x		Undefined
xxxxn12DH	CAN message data register 095	M_DATA095	R/W		x		Undefined
xxxxn12EH	CAN message data register 096	M_DATA096	R/W		x		Undefined
xxxxn12FH	CAN message data register 097	M_DATA097	R/W		x		Undefined
xxxxn130H	CAN message ID register L09	M_IDL09	R/W			x	Undefined
xxxxn132H	CAN message ID register H09	M_IDH09	R/W			x	Undefined
xxxxn134H	CAN message configuration register 09	M_CONF09	R/W		x		Undefined
xxxxn135H	CAN message status register 09	M_STAT09	R		x		Undefined
xxxxn136H	CAN status set/cancel register 09	SC_STAT09	W			x	0000H
xxxxn140H	CAN message event pointer 100	M_EVT100	R/W		x		Undefined
xxxxn141H	CAN message event pointer 101	M_EVT101	R/W		x		Undefined
xxxxn142H	CAN message event pointer 102	M_EVT102	R/W		x		Undefined
xxxxn143H	CAN message event pointer103	M_EVT103	R/W		x		Undefined
xxxxn144H	CAN message data length register 10	M_DLC10	R/W		x		Undefined
xxxxn145H	CAN message control register 10	M_CTRL10	R/W		x		Undefined
xxxxn146H	CAN message time stamp register 10	M_TIME10	R/W			x	Undefined
xxxxn148H	CAN message data register 100	M_DATA100	R/W		x		Undefined
xxxxn149H	CAN message data register 101	M_DATA101	R/W		x		Undefined
xxxxn14AH	CAN message data register 102	M_DATA102	R/W		x		Undefined
xxxxn14BH	CAN message data register 103	M_DATA103	R/W		x		Undefined
xxxxn14CH	CAN message data register 104	M_DATA104	R/W		x		Undefined
xxxxn14DH	CAN message data register 105	M_DATA105	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 6 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn14EH	CAN message data register 106	M_DATA106	R/W		x		Undefined
xxxxn14FH	CAN message data register 107	M_DATA107	R/W		x		Undefined
xxxxn150H	CAN message ID register L10	M_IDL10	R/W			x	Undefined
xxxxn152H	CAN message ID register H10	M_IDH10	R/W			x	Undefined
xxxxn154H	CAN message configuration register 10	M_CONF10	R/W		x		Undefined
xxxxn155H	CAN message status register 10	M_STAT10	R		x		Undefined
xxxxn156H	CAN status set/cancel register 10	SC_STAT10	W			x	0000H
xxxxn160H	CAN message event pointer 110	M_EVT110	R/W		x		Undefined
xxxxn161H	CAN message event pointer111	M_EVT111	R/W		x		Undefined
xxxxn162H	CAN message event pointer 112	M_EVT112	R/W		x		Undefined
xxxxn163H	CAN message event pointer 113	M_EVT113	R/W		x		Undefined
xxxxn164H	CAN message data length register 11	M_DLC11	R/W		x		Undefined
xxxxn165H	CAN message control register 11	M_CTRL11	R/W		x		Undefined
xxxxn166H	CAN message time stamp register 11	M_TIME11	R/W			x	Undefined
xxxxn168H	CAN message data register 110	M_DATA110	R/W		x		Undefined
xxxxn169H	CAN message data register 111	M_DATA111	R/W		x		Undefined
xxxxn16AH	CAN message data register 112	M_DATA112	R/W		x		Undefined
xxxxn16BH	CAN message data register 113	M_DATA113	R/W		x		Undefined
xxxxn16CH	CAN message data register 114	M_DATA114	R/W		x		Undefined
xxxxn16DH	CAN message data register 115	M_DATA115	R/W		x		Undefined
xxxxn16EH	CAN message data register 116	M_DATA116	R/W		x		Undefined
xxxxn16FH	CAN message data register 117	M_DATA117	R/W		x		Undefined
xxxxn170H	CAN message ID register L11	M_IDL11	R/W			x	Undefined
xxxxn172H	CAN message ID register H11	M_IDH11	R/W			x	Undefined
xxxxn174H	CAN message configuration register 11	M_CONF11	R/W		x		Undefined
xxxxn175H	CAN message status register 11	M_STAT11	R		x		Undefined
xxxxn176H	CAN status set/cancel register 11	SC_STAT11	W			x	0000H
xxxxn180H	CAN message event pointer 120	M_EVT120	R/W		x		Undefined
xxxxn181H	CAN message event pointer 121	M_EVT121	R/W		x		Undefined
xxxxn182H	CAN message event pointer 122	M_EVT122	R/W		x		Undefined
xxxxn183H	CAN message event pointer 123	M_EVT123	R/W		x		Undefined
xxxxn184H	CAN message data length register 12	M_DLC12	R/W		x		Undefined
xxxxn185H	CAN message control register 12	M_CTRL12	R/W		x		Undefined
xxxxn186H	CAN message time stamp register 12	M_TIME12	R/W			x	Undefined
xxxxn188H	CAN message data register 120	M_DATA120	R/W		x		Undefined
xxxxn189H	CAN message data register 121	M_DATA121	R/W		x		Undefined
xxxxn18AH	CAN message data register 122	M_DATA122	R/W		x		Undefined
xxxxn18BH	CAN message data register 123	M_DATA123	R/W		x		Undefined
xxxxn18CH	CAN message data register 124	M_DATA124	R/W		x		Undefined
xxxxn18DH	CAN message data register 125	M_DATA125	R/W		x		Undefined
xxxxn18EH	CAN message data register 126	M_DATA126	R/W		x		Undefined
xxxxn18FH	CAN message data register 127	M_DATA127	R/W		x		Undefined
xxxxn190H	CAN message ID register L12	M_IDL12	R/W			x	Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 7 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn192H	CAN message ID register H12	M_IDH12	R/W			x	Undefined
xxxxn194H	CAN message configuration register 12	M_CONF12	R/W		x		Undefined
xxxxn195H	CAN message status register 12	M_STAT12	R		x		Undefined
xxxxn196H	CAN status set/cancel register 12	SC_STAT12	W			x	0000H
xxxxn1A0H	CAN message event pointer 130	M_EVT130	R/W		x		Undefined
xxxxn1A1H	CAN message event pointer 131	M_EVT131	R/W		x		Undefined
xxxxn1A2H	CAN message event pointer 132	M_EVT132	R/W		x		Undefined
xxxxn1A3H	CAN message event pointer 133	M_EVT133	R/W		x		Undefined
xxxxn1A4H	CAN message data length register 13	M_DLC13	R/W		x		Undefined
xxxxn1A5H	CAN message control register 13	M_CTRL13	R/W		x		Undefined
xxxxn1A6H	CAN message time stamp register 13	M_TIME13	R/W			x	Undefined
xxxxn1A8H	CAN message data register 130	M_DATA130	R/W		x		Undefined
xxxxn1A9H	CAN message data register 131	M_DATA131	R/W		x		Undefined
xxxxn1AAH	CAN message data register 132	M_DATA132	R/W		x		Undefined
xxxxn1ABH	CAN message data register 133	M_DATA133	R/W		x		Undefined
xxxxn1ACH	CAN message data register 134	M_DATA134	R/W		x		Undefined
xxxxn1ADH	CAN message data register 135	M_DATA135	R/W		x		Undefined
xxxxn1AEH	CAN message data register 136	M_DATA136	R/W		x		Undefined
xxxxn1AFH	CAN message data register 137	M_DATA137	R/W		x		Undefined
xxxxn1B0H	CAN message ID register L13	M_IDL13	R/W			x	Undefined
xxxxn1B2H	CAN message ID register H13	M_IDH13	R/W			x	Undefined
xxxxn1B4H	CAN message configuration register 13	M_CONF13	R/W		x		Undefined
xxxxn1B5H	CAN message status register 13	M_STAT13	R		x		Undefined
xxxxn1B6H	CAN status set/cancel register 13	SC_STAT13	W			x	0000H
xxxxn1C0H	CAN message event pointer 140	M_EVT140	R/W		x		Undefined
xxxxn1C1H	CAN message event pointer 141	M_EVT141	R/W		x		Undefined
xxxxn1C2H	CAN message event pointer 142	M_EVT142	R/W		x		Undefined
xxxxn1C3H	CAN message event pointer 143	M_EVT143	R/W		x		Undefined
xxxxn1C4H	CAN message data length register 14	M_DLC14	R/W		x		Undefined
xxxxn1C5H	CAN message control register 14	M_CTRL14	R/W		x		Undefined
xxxxn1C6H	CAN message time stamp register 14	M_TIME14	R/W			x	Undefined
xxxxn1C8H	CAN message data register 140	M_DATA140	R/W		x		Undefined
xxxxn1C9H	CAN message data register 141	M_DATA141	R/W		x		Undefined
xxxxn1CAH	CAN message data register 142	M_DATA142	R/W		x		Undefined
xxxxn1CBH	CAN message data register 143	M_DATA143	R/W		x		Undefined
xxxxn1CCH	CAN message data register 144	M_DATA144	R/W		x		Undefined
xxxxn1CDH	CAN message data register 145	M_DATA145	R/W		x		Undefined
xxxxn1CEH	CAN message data register 146	M_DATA146	R/W		x		Undefined
xxxxn1CFH	CAN message data register 147	M_DATA147	R/W		x		Undefined
xxxxn1D0H	CAN message ID register L14	M_IDL14	R/W			x	Undefined
xxxxn1D2H	CAN message ID register H14	M_IDH14	R/W			x	Undefined
xxxxn1D4H	CAN message configuration register 14	M_CONF14	R/W		x		Undefined
xxxxn1D5H	CAN message status register 14	M_STAT14	R		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 8 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn1D6H	CAN status set/cancel register 14	SC_STAT14	W			x	0000H
xxxxn1E0H	CAN message event pointer 150	M_EVT150	R/W		x		Undefined
xxxxn1E1H	CAN message event pointer 151	M_EVT151	R/W		x		Undefined
xxxxn1E2H	CAN message event pointer 152	M_EVT152	R/W		x		Undefined
xxxxn1E3H	CAN message event pointer 153	M_EVT153	R/W		x		Undefined
xxxxn1E4H	CAN message data length register 15	M_DLC15	R/W		x		Undefined
Xxxxn1E5H	CAN message control register 15	M_CTRL15	R/W		x		Undefined
Xxxxn1E6H	CAN message time stamp register 15	M_TIME15	R/W			x	Undefined
xxxxn1E8H	CAN message data register 150	M_DATA150	R/W		x		Undefined
xxxxn1E9H	CAN message data register 151	M_DATA151	R/W		x		Undefined
xxxxn1EAH	CAN message data register 152	M_DATA152	R/W		x		Undefined
xxxxn1EBH	CAN message data register 153	M_DATA153	R/W		x		Undefined
xxxxn1ECH	CAN message data register 154	M_DATA154	R/W		x		Undefined
xxxxn1EDH	CAN message data register 155	M_DATA155	R/W		x		Undefined
xxxxn1EEH	CAN message data register 156	M_DATA156	R/W		x		Undefined
xxxxn1EFH	CAN message data register 157	M_DATA157	R/W		x		Undefined
xxxxn1F0H	CAN message ID register L15	M_IDL15	R/W			x	Undefined
xxxxn1F2H	CAN message ID register H15	M_IDH15	R/W			x	Undefined
xxxxn1F4H	CAN message configuration register 15	M_CONF15	R/W		x		Undefined
xxxxn1F5H	CAN message status register 15	M_STAT15	R		x		Undefined
xxxxn1F6H	CAN status set/cancel register 15	SC_STAT15	W			x	0000H
xxxxn200H	CAN message event pointer 160	M_EVT160	R/W		x		Undefined
xxxxn201H	CAN message event pointer 161	M_EVT161	R/W		x		Undefined
xxxxn202H	CAN message event pointer 162	M_EVT162	R/W		x		Undefined
xxxxn203H	CAN message event pointer 163	M_EVT163	R/W		x		Undefined
xxxxn204H	CAN message data length register 16	M_DLC16	R/W		x		Undefined
xxxxn205H	CAN message control register 16	M_CTRL16	R/W		x		Undefined
xxxxn206H	CAN message time stamp register 16	M_TIME16	R/W			x	Undefined
xxxxn208H	CAN message data register 160	M_DATA160	R/W		x		Undefined
xxxxn209H	CAN message data register 161	M_DATA161	R/W		x		Undefined
xxxxn20AH	CAN message data register 162	M_DATA162	R/W		x		Undefined
xxxxn20BH	CAN message data register 163	M_DATA163	R/W		x		Undefined
xxxxn20CH	CAN message data register 164	M_DATA164	R/W		x		Undefined
xxxxn20DH	CAN message data register 165	M_DATA165	R/W		x		Undefined
xxxxn20EH	CAN message data register 166	M_DATA166	R/W		x		Undefined
xxxxn20FH	CAN message data register 167	M_DATA167	R/W		x		Undefined
xxxxn210H	CAN message ID register L16	M_IDL16	R/W			x	Undefined
xxxxn212H	CAN message ID register H16	M_IDH16	R/W			x	Undefined
xxxxn214H	CAN message configuration register 16	M_CONF16	R/W		x		Undefined
xxxxn215H	CAN message status register 16	M_STAT16	R		x		Undefined
xxxxn216H	CAN status set/cancel register 16	SC_STAT16	W			x	0000H
xxxxn220H	CAN message event pointer 170	M_EVT170	R/W		x		Undefined
xxxxn221H	CAN message event pointer 171	M_EVT171	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 9 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn222H	CAN message event pointer 172	M_EVT172	R/W		x		Undefined
xxxxn223H	CAN message event pointer 173	M_EVT173	R/W		x		Undefined
xxxxn224H	CAN message data length register 17	M_DLC17	R/W		x		Undefined
xxxxn225H	CAN message control register 17	M_CTRL17	R/W		x		Undefined
xxxxn226H	CAN message time stamp register 17	M_TIME17	R/W			x	Undefined
xxxxn228H	CAN message data register 170	M_DATA170	R/W		x		Undefined
xxxxn229H	CAN message data register 171	M_DATA171	R/W		x		Undefined
xxxxn22AH	CAN message data register 172	M_DATA172	R/W		x		Undefined
xxxxn22BH	CAN message data register 173	M_DATA173	R/W		x		Undefined
xxxxn22CH	CAN message data register 174	M_DATA174	R/W		x		Undefined
xxxxn22DH	CAN message data register 175	M_DATA175	R/W		x		Undefined
xxxxn22EH	CAN message data register 176	M_DATA176	R/W		x		Undefined
xxxxn22FH	CAN message data register 177	M_DATA177	R/W		x		Undefined
xxxxn230H	CAN message ID register L17	M_IDL17	R/W			x	Undefined
xxxxn232H	CAN message ID register H17	M_IDH17	R/W			x	Undefined
xxxxn234H	CAN message configuration register 17	M_CONF17	R/W		x		Undefined
xxxxn235H	CAN message status register 17	M_STAT17	R		x		Undefined
xxxxn236H	CAN status set/cancel register 17	SC_STAT17	W			x	0000H
xxxxn240H	CAN message event pointer 180	M_EVT180	R/W		x		Undefined
xxxxn241H	CAN message event pointer 181	M_EVT181	R/W		x		Undefined
xxxxn242H	CAN message event pointer 182	M_EVT182	R/W		x		Undefined
xxxxn243H	CAN message event pointer 183	M_EVT183	R/W		x		Undefined
xxxxn244H	CAN message data length register 18	M_DLC18	R/W		x		Undefined
xxxxn245H	CAN message control register 18	M_CTRL18	R/W		x		Undefined
xxxxn246H	CAN message time stamp register 18	M_TIME18	R/W			x	Undefined
xxxxn248H	CAN message data register 180	M_DATA180	R/W		x		Undefined
xxxxn249H	CAN message data register 181	M_DATA181	R/W		x		Undefined
xxxxn24AH	CAN message data register 182	M_DATA182	R/W		x		Undefined
xxxxn24BH	CAN message data register 183	M_DATA183	R/W		x		Undefined
xxxxn24CH	CAN message data register 184	M_DATA184	R/W		x		Undefined
xxxxn24DH	CAN message data register 185	M_DATA185	R/W		x		Undefined
xxxxn24EH	CAN message data register 186	M_DATA186	R/W		x		Undefined
xxxxn24FH	CAN message data register 187	M_DATA187	R/W		x		Undefined
xxxxn250H	CAN message ID register L18	M_IDL18	R/W			x	Undefined
xxxxn252H	CAN message ID register H18	M_IDH18	R/W			x	Undefined
xxxxn254H	CAN message configuration register 18	M_CONF18	R/W		x		Undefined
xxxxn255H	CAN message status register 18	M_STAT18	R		x		Undefined
xxxxn256H	CAN status set/cancel register 18	SC_STAT18	W			x	0000H
xxxxn260H	CAN message event pointer 260	M_EVT260	R/W		x		Undefined
xxxxn261H	CAN message event pointer 261	M_EVT261	R/W		x		Undefined
xxxxn262H	CAN message event pointer 262	M_EVT262	R/W		x		Undefined
xxxxn263H	CAN message event pointer 263	M_EVT263	R/W		x		Undefined
xxxxn264H	CAN message data length register 19	M_DLC19	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 10 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn265H	CAN message control register 19	M_CTRL19	R/W		x		Undefined
xxxxn266H	CAN message time stamp register 19	M_TIME19	R/W			x	Undefined
xxxxn268H	CAN message data register 190	M_DATA190	R/W		x		Undefined
xxxxn269H	CAN message data register 191	M_DATA191	R/W		x		Undefined
xxxxn26AH	CAN message data register 192	M_DATA192	R/W		x		Undefined
xxxxn26BH	CAN message data register 193	M_DATA193	R/W		x		Undefined
xxxxn26CH	CAN message data register 194	M_DATA194	R/W		x		Undefined
xxxxn26DH	CAN message data register 195	M_DATA195	R/W		x		Undefined
xxxxn26EH	CAN message data register 196	M_DATA196	R/W		x		Undefined
xxxxn26FH	CAN message data register 197	M_DATA197	R/W		x		Undefined
xxxxn270H	CAN message ID register L19	M_IDL19	R/W			x	Undefined
xxxxn272H	CAN message ID register H19	M_IDH19	R/W			x	Undefined
xxxxn274H	CAN message configuration register 19	M_CONF19	R/W		x		Undefined
xxxxn275H	CAN message status register 19	M_STAT19	R		x		Undefined
xxxxn276H	CAN status set/cancel register 19	SC_STAT19	W			x	0000H
xxxxn280H	CAN message event pointer 200	M_EVT200	R/W		x		Undefined
xxxxn281H	CAN message event pointer 201	M_EVT201	R/W		x		Undefined
xxxxn282H	CAN message event pointer 202	M_EVT202	R/W		x		Undefined
xxxxn283H	CAN message event pointer 203	M_EVT203	R/W		x		Undefined
xxxxn284H	CAN message data length register 20	M_DLC20	R/W		x		Undefined
xxxxn285H	CAN message control register 20	M_CTRL20	R/W		x		Undefined
xxxxn286H	CAN message time stamp register 20	M_TIME20	R/W			x	Undefined
xxxxn288H	CAN message data register 200	M_DATA200	R/W		x		Undefined
xxxxn289H	CAN message data register 201	M_DATA201	R/W		x		Undefined
xxxxn28AH	CAN message data register 202	M_DATA202	R/W		x		Undefined
xxxxn28BH	CAN message data register 203	M_DATA203	R/W		x		Undefined
xxxxn28CH	CAN message data register 204	M_DATA204	R/W		x		Undefined
xxxxn28DH	CAN message data register 205	M_DATA205	R/W		x		Undefined
xxxxn28EH	CAN message data register 206	M_DATA206	R/W		x		Undefined
xxxxn28FH	CAN message data register 207	M_DATA207	R/W		x		Undefined
xxxxn290H	CAN message ID register L20	M_IDL20	R/W			x	Undefined
xxxxn292H	CAN message ID register H20	M_IDH20	R/W			x	Undefined
xxxxn294H	CAN message configuration register 20	M_CONF20	R/W		x		Undefined
xxxxn295H	CAN message status register 20	M_STAT20	R		x		Undefined
xxxxn296H	CAN status set/cancel register 20	SC_STAT20	W			x	0000H
xxxxn2A0H	CAN message event pointer 210	M_EVT210	R/W		x		Undefined
xxxxn2A1H	CAN message event pointer 211	M_EVT211	R/W		x		Undefined
xxxxn22AH	CAN message event pointer 212	M_EVT212	R/W		x		Undefined
xxxxn2A3H	CAN message event pointer 213	M_EVT213	R/W		x		Undefined
xxxxn2A4H	CAN message data length register 21	M_DLC21	R/W		x		Undefined
xxxxn2A5H	CAN message control register 21	M_CTRL21	R/W		x		Undefined
xxxxn2A6H	CAN message time stamp register 21	M_TIME21	R/W			x	Undefined
xxxxn2A8H	CAN message data register 210	M_DATA210	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 11 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn2A9H	CAN message data register 211	M_DATA211	R/W		x		Undefined
xxxxn2AAH	CAN message data register 212	M_DATA212	R/W		x		Undefined
xxxxn2ABH	CAN message data register 213	M_DATA213	R/W		x		Undefined
xxxxn2ACH	CAN message data register 214	M_DATA214	R/W		x		Undefined
xxxxn2ADH	CAN message data register 215	M_DATA215	R/W		x		Undefined
xxxxn2AEH	CAN message data register 216	M_DATA216	R/W		x		Undefined
xxxxn2AFH	CAN message data register 217	M_DATA217	R/W		x		Undefined
xxxxn2B0H	CAN message ID register L21	M_IDL21	R/W			x	Undefined
xxxxn2B2H	CAN message ID register H21	M_IDH21	R/W			x	Undefined
xxxxn2B4H	CAN message configuration register 21	M_CONF21	R/W		x		Undefined
xxxxn2B5H	CAN message status register 21	M_STAT21	R		x		Undefined
xxxxn2B6H	CAN status set/cancel register 21	SC_STAT21	W			x	0000H
xxxxn2C0H	CAN message event pointer 220	M_EVT220	R/W		x		Undefined
xxxxn2C1H	CAN message event pointer 221	M_EVT221	R/W		x		Undefined
xxxxn2C2H	CAN message event pointer 222	M_EVT222	R/W		x		Undefined
xxxxn2C3H	CAN message event pointer 223	M_EVT223	R/W		x		Undefined
xxxxn2C4H	CAN message data length register 22	M_DLC22	R/W		x		Undefined
xxxxn2C5H	CAN message control register 22	M_CTRL22	R/W		x		Undefined
xxxxn2C6H	CAN message time stamp register 22	M_TIME22	R/W			x	Undefined
xxxxn2C8H	CAN message data register 220	M_DATA220	R/W		x		Undefined
xxxxn2C9H	CAN message data register 221	M_DATA221	R/W		x		Undefined
xxxxn2CAH	CAN message data register 222	M_DATA222	R/W		x		Undefined
xxxxn2CBH	CAN message data register 223	M_DATA223	R/W		x		Undefined
xxxxn2CCH	CAN message data register 224	M_DATA224	R/W		x		Undefined
xxxxn2CDH	CAN message data register 225	M_DATA225	R/W		x		Undefined
xxxxn2CEH	CAN message data register 226	M_DATA226	R/W		x		Undefined
xxxxn2CFH	CAN message data register 227	M_DATA227	R/W		x		Undefined
xxxxn2D0H	CAN message ID register L22	M_IDL22	R/W			x	Undefined
xxxxn2D2H	CAN message ID register H22	M_IDH22	R/W			x	Undefined
xxxxn2D4H	CAN message configuration register 22	M_CONF22	R/W		x		Undefined
xxxxn2D5H	CAN message status register 22	M_STAT22	R		x		Undefined
xxxxn2D6H	CAN status set/cancel register 22	SC_STAT22	W			x	0000H
xxxxn2E0H	CAN message event pointer 230	M_EVT230	R/W		x		Undefined
xxxxn2E1H	CAN message event pointer 231	M_EVT231	R/W		x		Undefined
xxxxn2EH	CAN message event pointer 232	M_EVT232	R/W		x		Undefined
xxxxn2E3H	CAN message event pointer 233	M_EVT233	R/W		x		Undefined
xxxxn2E4H	CAN message data length register 23	M_DLC23	R/W		x		Undefined
xxxxn2E5H	CAN message control register 23	M_CTRL23	R/W		x		Undefined
xxxxn2E6H	CAN message time stamp register 23	M_TIME23	R/W			x	Undefined
xxxxn2E8H	CAN message data register 230	M_DATA230	R/W		x		Undefined
xxxxn2E9H	CAN message data register 231	M_DATA231	R/W		x		Undefined
xxxxn2EAH	CAN message data register 232	M_DATA232	R/W		x		Undefined
xxxxn2EBH	CAN message data register 233	M_DATA233	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 12 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn2ECH	CAN message data register 234	M_DATA234	R/W		x		Undefined
xxxxn2EDH	CAN message data register 235	M_DATA235	R/W		x		Undefined
xxxxn2EEH	CAN message data register 236	M_DATA236	R/W		x		Undefined
xxxxn2EFH	CAN message data register 237	M_DATA237	R/W		x		Undefined
xxxxn2F0H	CAN message ID register L23	M_IDL23	R/W			x	Undefined
xxxxn2F2H	CAN message ID register H23	M_IDH23	R/W			x	Undefined
xxxxn2F4H	CAN message configuration register 23	M_CONF23	R/W		x		Undefined
xxxxn2F5H	CAN message status register 23	M_STAT23	R		x		Undefined
xxxxn2F6H	CAN status set/cancel register 23	SC_STAT23	W			x	0000H
xxxxn300H	CAN message event pointer 240	M_EVT240	R/W		x		Undefined
xxxxn301H	CAN message event pointer 241	M_EVT241	R/W		x		Undefined
xxxxn302H	CAN message event pointer 242	M_EVT242	R/W		x		Undefined
xxxxn303H	CAN message event pointer 243	M_EVT243	R/W		x		Undefined
xxxxn304H	CAN message data length register 24	M_DLC24	R/W		x		Undefined
xxxxn305H	CAN message control register 24	M_CTRL24	R/W		x		Undefined
xxxxn306H	CAN message time stamp register 24	M_TIME24	R/W			x	Undefined
xxxxn308H	CAN message data register 240	M_DATA240	R/W		x		Undefined
xxxxn309H	CAN message data register 241	M_DATA241	R/W		x		Undefined
xxxxn30AH	CAN message data register 242	M_DATA242	R/W		x		Undefined
xxxxn30BH	CAN message data register 243	M_DATA243	R/W		x		Undefined
xxxxn30CH	CAN message data register 244	M_DATA244	R/W		x		Undefined
xxxxn30DH	CAN message data register 245	M_DATA245	R/W		x		Undefined
xxxxn30EH	CAN message data register 246	M_DATA246	R/W		x		Undefined
xxxxn30FH	CAN message data register 247	M_DATA247	R/W		x		Undefined
xxxxn310H	CAN message ID register L24	M_IDL24	R/W			x	Undefined
xxxxn312H	CAN message ID register H24	M_IDH24	R/W			x	Undefined
xxxxn314H	CAN message configuration register 24	M_CONF24	R/W		x		Undefined
xxxxn315H	CAN message status register 24	M_STAT24	R		x		Undefined
xxxxn316H	CAN status set/cancel register 24	SC_STAT24	W			x	0000H
xxxxn320H	CAN message event pointer 250	M_EVT250	R/W		x		Undefined
xxxxn321H	CAN message event pointer 251	M_EVT251	R/W		x		Undefined
xxxxn322H	CAN message event pointer 252	M_EVT252	R/W		x		Undefined
xxxxn323H	CAN message event pointer 253	M_EVT253	R/W		x		Undefined
xxxxn324H	CAN message data length register 25	M_DLC25	R/W		x		Undefined
xxxxn325H	CAN message control register 25	M_CTRL25	R/W		x		Undefined
xxxxn326H	CAN message time stamp register 25	M_TIME25	R/W			x	Undefined
xxxxn328H	CAN message data register 250	M_DATA250	R/W		x		Undefined
xxxxn329H	CAN message data register 251	M_DATA251	R/W		x		Undefined
xxxxn32AH	CAN message data register 252	M_DATA252	R/W		x		Undefined
xxxxn32BH	CAN message data register 253	M_DATA253	R/W		x		Undefined
xxxxn32CH	CAN message data register 254	M_DATA254	R/W		x		Undefined
xxxxn32DH	CAN message data register 255	M_DATA255	R/W		x		Undefined
xxxxn32EH	CAN message data register 256	M_DATA256	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 13 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn32FH	CAN message data register 257	M_DATA257	R/W		x		Undefined
xxxxn330H	CAN message ID register L25	M_IDL25	R/W			x	Undefined
xxxxn332H	CAN message ID register H25	M_IDH25	R/W			x	Undefined
xxxxn334H	CAN message configuration register 25	M_CONF25	R/W		x		Undefined
xxxxn335H	CAN message status register 25	M_STAT25	R		x		Undefined
xxxxn336H	CAN status set/cancel register 25	SC_STAT25	W			x	0000H
xxxxn340H	CAN message event pointer 260	M_EVT260	R/W		x		Undefined
xxxxn341H	CAN message event pointer 261	M_EVT261	R/W		x		Undefined
xxxxn342H	CAN message event pointer 262	M_EVT262	R/W		x		Undefined
xxxxn343H	CAN message event pointer 263	M_EVT263	R/W		x		Undefined
xxxxn344H	CAN message data length register 26	M_DLC26	R/W		x		Undefined
xxxxn345H	CAN message control register 26	M_CTRL26	R/W		x		Undefined
xxxxn346H	CAN message time stamp register 26	M_TIME26	R/W			x	Undefined
xxxxn348H	CAN message data register 260	M_DATA260	R/W		x		Undefined
xxxxn349H	CAN message data register 261	M_DATA261	R/W		x		Undefined
xxxxn34AH	CAN message data register 262	M_DATA262	R/W		x		Undefined
xxxxn34BH	CAN message data register 263	M_DATA263	R/W		x		Undefined
xxxxn34CH	CAN message data register 264	M_DATA264	R/W		x		Undefined
xxxxn34DH	CAN message data register 265	M_DATA265	R/W		x		Undefined
xxxxn34EH	CAN message data register 266	M_DATA266	R/W		x		Undefined
xxxxn34FH	CAN message data register 267	M_DATA267	R/W		x		Undefined
xxxxn350H	CAN message ID register L26	M_IDL26	R/W			x	Undefined
xxxxn352H	CAN message ID register H26	M_IDH26	R/W			x	Undefined
xxxxn354H	CAN message configuration register 26	M_CONF26	R/W		x		Undefined
xxxxn355H	CAN message status register 26	M_STAT26	R		x		Undefined
xxxxn356H	CAN status set/cancel register 26	SC_STAT26	W			x	0000H
xxxxn360H	CAN message event pointer 270	M_EVT270	R/W		x		Undefined
xxxxn361H	CAN message event pointer 271	M_EVT271	R/W		x		Undefined
xxxxn362H	CAN message event pointer 272	M_EVT272	R/W		x		Undefined
xxxxn363H	CAN message event pointer 273	M_EVT273	R/W		x		Undefined
xxxxn364H	CAN message data length register 27	M_DLC27	R/W		x		Undefined
xxxxn365H	CAN message control register 27	M_CTRL27	R/W		x		Undefined
xxxxn366H	CAN message time stamp register 27	M_TIME27	R/W			x	Undefined
xxxxn368H	CAN message data register 270	M_DATA270	R/W		x		Undefined
xxxxn369H	CAN message data register 271	M_DATA271	R/W		x		Undefined
xxxxn36AH	CAN message data register 272	M_DATA272	R/W		x		Undefined
xxxxn36BH	CAN message data register 273	M_DATA273	R/W		x		Undefined
xxxxn36CH	CAN message data register 274	M_DATA274	R/W		x		Undefined
xxxxn36DH	CAN message data register 275	M_DATA275	R/W		x		Undefined
xxxxn36EH	CAN message data register 276	M_DATA276	R/W		x		Undefined
xxxxn36FH	CAN message data register 277	M_DATA277	R/W		x		Undefined
xxxxn370H	CAN message ID register L27	M_IDL27	R/W			x	Undefined
xxxxn372H	CAN message ID register H27	M_IDH27	R/W			x	Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 14 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn374H	CAN message configuration register 27	M_CONF27	R/W		x		Undefined
xxxxn375H	CAN message status register 27	M_STAT27	R		x		Undefined
xxxxn376H	CAN status set/cancel register 27	SC_STAT27	W			x	0000H
xxxxn380H	CAN message event pointer 280	M_EVT280	R/W		x		Undefined
xxxxn381H	CAN message event pointer 281	M_EVT281	R/W		x		Undefined
xxxxn382H	CAN message event pointer 282	M_EVT282	R/W		x		Undefined
xxxxn383H	CAN message event pointer 283	M_EVT283	R/W		x		Undefined
xxxxn384H	CAN message data length register 28	M_DLC28	R/W		x		Undefined
xxxxn385H	CAN message control register 28	M_CTRL28	R/W		x		Undefined
xxxxn386H	CAN message time stamp register 28	M_TIME28	R/W			x	Undefined
xxxxn388H	CAN message data register 280	M_DATA280	R/W		x		Undefined
xxxxn389H	CAN message data register 281	M_DATA281	R/W		x		Undefined
xxxxn38AH	CAN message data register 282	M_DATA282	R/W		x		Undefined
xxxxn38BH	CAN message data register 283	M_DATA283	R/W		x		Undefined
xxxxn38CH	CAN message data register 284	M_DATA284	R/W		x		Undefined
xxxxn38DH	CAN message data register 285	M_DATA285	R/W		x		Undefined
xxxxn38EH	CAN message data register 286	M_DATA286	R/W		x		Undefined
xxxxn38FH	CAN message data register 287	M_DATA287	R/W		x		Undefined
xxxxn390H	CAN message ID register L28	M_IDL28	R/W			x	Undefined
xxxxn392H	CAN message ID register H28	M_IDH28	R/W			x	Undefined
xxxxn394H	CAN message configuration register 28	M_CONF28	R/W		x		Undefined
xxxxn395H	CAN message status register 28	M_STAT28	R		x		Undefined
xxxxn396H	CAN status set/cancel register 28	SC_STAT28	W			x	0000H
xxxxn3A0H	CAN message event pointer 290	M_EVT290	R/W		x		Undefined
xxxxn3A1H	CAN message event pointer 291	M_EVT291	R/W		x		Undefined
xxxxn3A2H	CAN message event pointer 292	M_EVT292	R/W		x		Undefined
xxxxn3A3H	CAN message event pointer 293	M_EVT293	R/W		x		Undefined
xxxxn3A4H	CAN message data length register 29	M_DLC29	R/W		x		Undefined
xxxxn3A5H	CAN message control register 29	M_CTRL29	R/W		x		Undefined
xxxxn3A6H	CAN message time stamp register 29	M_TIME29	R/W			x	Undefined
xxxxn3A8H	CAN message data register 290	M_DATA290	R/W		x		Undefined
xxxxn3A9H	CAN message data register 291	M_DATA291	R/W		x		Undefined
xxxxn3AAH	CAN message data register 292	M_DATA292	R/W		x		Undefined
xxxxn3ABH	CAN message data register 293	M_DATA293	R/W		x		Undefined
xxxxn3ACH	CAN message data register 294	M_DATA294	R/W		x		Undefined
xxxxn3ADH	CAN message data register 295	M_DATA295	R/W		x		Undefined
xxxxn3AEH	CAN message data register 296	M_DATA296	R/W		x		Undefined
xxxxn3AFH	CAN message data register 297	M_DATA297	R/W		x		Undefined
xxxxn3B0H	CAN message ID register L29	M_IDL29	R/W			x	Undefined
xxxxn3B2H	CAN message ID register H29	M_IDH29	R/W			x	Undefined
xxxxn3B4H	CAN message configuration register 29	M_CONF29	R/W		x		Undefined
xxxxn3B5H	CAN message status register 29	M_STAT29	R		x		Undefined
xxxxn3B6H	CAN status set/cancel register 29	SC_STAT29	W			x	0000H

Table 3-5: List of programmable peripheral I/O registers (Sheet 15 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn3C0H	CAN message event pointer 300	M_EVT300	R/W		x		Undefined
xxxxn3C1H	CAN message event pointer 301	M_EVT301	R/W		x		Undefined
xxxxn3C2H	CAN message event pointer 302	M_EVT302	R/W		x		Undefined
xxxxn3C3H	CAN message event pointer 303	M_EVT303	R/W		x		Undefined
xxxxn3C4H	CAN message data length register 30	M_DLC30	R/W		x		Undefined
xxxxn3C5H	CAN message control register 30	M_CTRL30	R/W		x		Undefined
xxxxn3C6H	CAN message time stamp register 30	M_TIME30	R/W			x	Undefined
xxxxn3C8H	CAN message data register 300	M_DATA300	R/W		x		Undefined
xxxxn3C9H	CAN message data register 301	M_DATA301	R/W		x		Undefined
xxxxn3CAH	CAN message data register 302	M_DATA302	R/W		x		Undefined
xxxxn3CBH	CAN message data register 303	M_DATA303	R/W		x		Undefined
xxxxn3CCH	CAN message data register 304	M_DATA304	R/W		x		Undefined
xxxxn3CDH	CAN message data register 305	M_DATA305	R/W		x		Undefined
xxxxn3CEH	CAN message data register 306	M_DATA306	R/W		x		Undefined
xxxxn3CFH	CAN message data register 307	M_DATA307	R/W		x		Undefined
xxxxn3D0H	CAN message ID register L30	M_IDL30	R/W			x	Undefined
xxxxn3D2H	CAN message ID register H30	M_IDH30	R/W			x	Undefined
xxxxn3D4H	CAN message configuration register 30	M_CONF30	R/W		x		Undefined
xxxxn3D5H	CAN message status register 30	M_STAT30	R		x		Undefined
xxxxn3D6H	CAN status set/cancel register 30	SC_STAT30	W			x	0000H
xxxxn3E0H	CAN message event pointer 310	M_EVT310	R/W		x		Undefined
xxxxn3E1H	CAN message event pointer 311	M_EVT311	R/W		x		Undefined
xxxxn3E2H	CAN message event pointer 312	M_EVT312	R/W		x		Undefined
xxxxn3E3H	CAN message event pointer 313	M_EVT313	R/W		x		Undefined
xxxxn3E4H	CAN message data length register 31	M_DLC31	R/W		x		Undefined
xxxxn3E5H	CAN message control register 31	M_CTRL31	R/W		x		Undefined
xxxxn3E6H	CAN message time stamp register 31	M_TIME31	R/W			x	Undefined
xxxxn3E8H	CAN message data register 310	M_DATA310	R/W		x		Undefined
xxxxn3E9H	CAN message data register 311	M_DATA311	R/W		x		Undefined
xxxxn3EAH	CAN message data register 312	M_DATA312	R/W		x		Undefined
xxxxn3EBH	CAN message data register 313	M_DATA313	R/W		x		Undefined
xxxxn3ECH	CAN message data register 314	M_DATA314	R/W		x		Undefined
xxxxn3EDH	CAN message data register 315	M_DATA315	R/W		x		Undefined
xxxxn3EEH	CAN message data register 316	M_DATA316	R/W		x		Undefined
xxxxn3EFH	CAN message data register 317	M_DATA317	R/W		x		Undefined
xxxxn3FCH	CAN message ID register L31	M_IDL31	R/W			x	Undefined
xxxxn3F2H	CAN message ID register H31	M_IDH31	R/W			x	Undefined
xxxxn3F4H	CAN message configuration register 31	M_CONF31	R/W		x		Undefined
xxxxn3F5H	CAN message status register 31	M_STAT31	R		x		Undefined
xxxxn3F6H	CAN status set/cancel register 31	SC_STAT31	W			x	0000H
xxxxn400H	CAN message event pointer 320	M_EVT320	R/W		x		Undefined
xxxxn401H	CAN message event pointer 321	M_EVT321	R/W		x		Undefined
xxxxn402H	CAN message event pointer 322	M_EVT322	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 16 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn403H	CAN message event pointer 323	M_EVT323	R/W		x		Undefined
xxxxn404H	CAN message data length register 32	M_DLC32	R/W		x		Undefined
xxxxn405H	CAN message control register 32	M_CTRL32	R/W		x		Undefined
xxxxn406H	CAN message time stamp register 32	M_TIME32	R/W			x	Undefined
xxxxn408H	CAN message data register 320	M_DATA320	R/W		x		Undefined
xxxxn409H	CAN message data register 321	M_DATA321	R/W		x		Undefined
xxxxn40AH	CAN message data register 322	M_DATA322	R/W		x		Undefined
xxxxn40BH	CAN message data register 323	M_DATA323	R/W		x		Undefined
xxxxn40CH	CAN message data register 324	M_DATA324	R/W		x		Undefined
xxxxn40DH	CAN message data register 325	M_DATA325	R/W		x		Undefined
xxxxn40EH	CAN message data register 326	M_DATA326	R/W		x		Undefined
xxxxn40FH	CAN message data register 327	M_DATA327	R/W		x		Undefined
xxxxn410H	CAN message ID register L32	M_IDL32	R/W			x	Undefined
xxxxn412H	CAN message ID register H32	M_IDH32	R/W			x	Undefined
xxxxn414H	CAN message configuration register 32	M_CONF32	R/W		x		Undefined
xxxxn415H	CAN message status register 32	M_STAT32	R		x		Undefined
xxxxn416H	CAN status set/cancel register 32	SC_STAT32	W			x	0000H
xxxxn420H	CAN message event pointer 330	M_EVT330	R/W		x		Undefined
xxxxn421H	CAN message event pointer 331	M_EVT331	R/W		x		Undefined
xxxxn422H	CAN message event pointer 332	M_EVT332	R/W		x		Undefined
xxxxn423H	CAN message event pointer 333	M_EVT333	R/W		x		Undefined
xxxxn424H	CAN message data length register 33	M_DLC33	R/W		x		Undefined
xxxxn425H	CAN message control register 33	M_CTRL33	R/W		x		Undefined
xxxxn426H	CAN message time stamp register 33	M_TIME33	R/W			x	Undefined
xxxxn428H	CAN message data register 330	M_DATA330	R/W		x		Undefined
xxxxn429H	CAN message data register 331	M_DATA331	R/W		x		Undefined
xxxxn42AH	CAN message data register 332	M_DATA332	R/W		x		Undefined
xxxxn42BH	CAN message data register 333	M_DATA333	R/W		x		Undefined
xxxxn42CH	CAN message data register 334	M_DATA334	R/W		x		Undefined
xxxxn42DH	CAN message data register 335	M_DATA335	R/W		x		Undefined
xxxxn42EH	CAN message data register 336	M_DATA336	R/W		x		Undefined
xxxxn42FH	CAN message data register 337	M_DATA337	R/W		x		Undefined
xxxxn430H	CAN message ID register L33	M_IDL33	R/W			x	Undefined
xxxxn432H	CAN message ID register H33	M_IDH33	R/W			x	Undefined
xxxxn434H	CAN message configuration register 33	M_CONF33	R/W		x		Undefined
xxxxn435H	CAN message status register 33	M_STAT33	R		x		Undefined
xxxxn436H	CAN status set/cancel register 33	SC_STAT33	W			x	0000H
xxxxn440H	CAN message event pointer 340	M_EVT340	R/W		x		Undefined
xxxxn441H	CAN message event pointer 341	M_EVT341	R/W		x		Undefined
xxxxn442H	CAN message event pointer 342	M_EVT342	R/W		x		Undefined
xxxxn443H	CAN message event pointer 343	M_EVT343	R/W		x		Undefined
xxxxn444H	CAN message data length register 34	M_DLC34	R/W		x		Undefined
xxxxn445H	CAN message control register 34	M_CTRL34	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 17 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn446H	CAN message time stamp register 34	M_TIME34	R/W			x	Undefined
xxxxn448H	CAN message data register 340	M_DATA340	R/W		x		Undefined
Xxxxn449H	CAN message data register 341	M_DATA341	R/W		x		Undefined
xxxxn44AH	CAN message data register 342	M_DATA342	R/W		x		Undefined
Xxxxn44BH	CAN message data register 343	M_DATA343	R/W		x		Undefined
xxxxn44CH	CAN message data register 344	M_DATA344	R/W		x		Undefined
xxxxn44DH	CAN message data register 345	M_DATA345	R/W		x		Undefined
xxxxn44EH	CAN message data register 346	M_DATA346	R/W		x		Undefined
xxxxn44FH	CAN message data register 347	M_DATA347	R/W		x		Undefined
xxxxn450H	CAN message ID register L34	M_IDL34	R/W			x	Undefined
xxxxn452H	CAN message ID register H34	M_IDH34	R/W			x	Undefined
xxxxn454H	CAN message configuration register 34	M_CONF34	R/W		x		Undefined
xxxxn455H	CAN message status register 34	M_STAT34	R		x		Undefined
xxxxn456H	CAN status set/cancel register 34	SC_STAT34	W			x	0000H
xxxxn460H	CAN message event pointer 350	M_EVT350	R/W		x		Undefined
xxxxn461H	CAN message event pointer 351	M_EVT351	R/W		x		Undefined
xxxxn462H	CAN message event pointer 352	M_EVT352	R/W		x		Undefined
xxxxn463H	CAN message event pointer 353	M_EVT353	R/W		x		Undefined
xxxxn464H	CAN message data length register 35	M_DLC35	R/W		x		Undefined
xxxxn465H	CAN message control register 35	M_CTRL35	R/W		x		Undefined
xxxxn466H	CAN message time stamp register 35	M_TIME35	R/W			x	Undefined
xxxxn468H	CAN message data register 350	M_DATA350	R/W		x		Undefined
xxxxn469H	CAN message data register 351	M_DATA351	R/W		x		Undefined
xxxxn46AH	CAN message data register 352	M_DATA352	R/W		x		Undefined
xxxxn46BH	CAN message data register 353	M_DATA353	R/W		x		Undefined
xxxxn46CH	CAN message data register 354	M_DATA354	R/W		x		Undefined
xxxxn46DH	CAN message data register 355	M_DATA355	R/W		x		Undefined
xxxxn46EH	CAN message data register 356	M_DATA356	R/W		x		Undefined
xxxxn46FH	CAN message data register 357	M_DATA357	R/W		x		Undefined
xxxxn470H	CAN message ID register L35	M_IDL35	R/W			x	Undefined
xxxxn472H	CAN message ID register H35	M_IDH35	R/W			x	Undefined
xxxxn474H	CAN message configuration register 35	M_CONF35	R/W		x		Undefined
xxxxn475H	CAN message status register 35	M_STAT35	R		x		Undefined
xxxxn476H	CAN status set/cancel register 35	SC_STAT35	W			x	0000H
xxxxn480H	CAN message event pointer 360	M_EVT360	R/W		x		Undefined
xxxxn481H	CAN message event pointer 361	M_EVT361	R/W		x		Undefined
xxxxn482H	CAN message event pointer 362	M_EVT362	R/W		x		Undefined
xxxxn483H	CAN message event pointer 363	M_EVT363	R/W		x		Undefined
xxxxn484H	CAN message data length register 36	M_DLC36	R/W		x		Undefined
xxxxn485H	CAN message control register 36	M_CTRL36	R/W		x		Undefined
xxxxn486H	CAN message time stamp register 36	M_TIME36	R/W			x	Undefined
xxxxn488H	CAN message data register 360	M_DATA360	R/W		x		Undefined
xxxxn489H	CAN message data register 361	M_DATA361	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 18 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn48AH	CAN message data register 362	M_DATA362	R/W		x		Undefined
xxxxn48BH	CAN message data register 363	M_DATA363	R/W		x		Undefined
xxxxn48CH	CAN message data register 364	M_DATA364	R/W		x		Undefined
xxxxn48DH	CAN message data register 365	M_DATA365	R/W		x		Undefined
xxxxn48EH	CAN message data register 366	M_DATA366	R/W		x		Undefined
xxxxn48FH	CAN message data register 367	M_DATA367	R/W		x		Undefined
Xxxxn490H	CAN message ID register L36	M_IDL36	R/W			x	Undefined
xxxxn482H	CAN message ID register H36	M_IDH36	R/W			x	Undefined
xxxxn494H	CAN message configuration register 36	M_CONF36	R/W		x		Undefined
xxxxn495H	CAN message status register 36	M_STAT36	R		x		Undefined
xxxxn496H	CAN status set/cancel register 36	M_STAT36	W			x	0000H
xxxxn4A0H	CAN message event pointer 370	M_EVT370	R/W		x		Undefined
xxxxn4A1H	CAN message event pointer 371	M_EVT371	R/W		x		Undefined
xxxxn4A2H	CAN message event pointer 372	M_EVT372	R/W		x		Undefined
xxxxn4A3H	CAN message event pointer 373	M_EVT373	R/W		x		Undefined
xxxxn4A4H	CAN message data length register 37	M_DLC37	R/W		x		Undefined
xxxxn4A5H	CAN message control register 37	M_DTRL37	R/W		x		Undefined
xxxxn4A6H	CAN message time stamp register 37	M_TIME37	R/W			x	Undefined
xxxxn4A8H	CAN message data register 370	M_DATA370	R/W		x		Undefined
xxxxn4A9H	CAN message data register 371	M_DATA371	R/W		x		Undefined
xxxxn4AAH	CAN message data register 372	M_DATA372	R/W		x		Undefined
xxxxn4ABH	CAN message data register 373	M_DATA373	R/W		x		Undefined
xxxxn4ACH	CAN message data register 374	M_DATA374	R/W		x		Undefined
xxxxn4ADH	CAN message data register 375	M_DATA375	R/W		x		Undefined
xxxxn4AEH	CAN message data register 376	M_DATA376	R/W		x		Undefined
xxxxn4AFH	CAN message data register 377	M_DATA377	R/W		x		Undefined
xxxxn4B0H	CAN message ID register L37	M_IDL37	R/W			x	Undefined
xxxxn4B2H	CAN message ID register H37	M_IDH37	R/W			x	Undefined
xxxxn4B4H	CAN message configuration register 37	M_CONF37	R/W		x		Undefined
xxxxn4B5H	CAN message status register 37	M_STAT37	R		x		Undefined
xxxxn4B6H	CAN status set/cancel register 37	SC_STAT37	W			x	0000H
xxxxn4C0H	CAN message event pointer 380	M_EVT380	R/W		x		Undefined
xxxxn4C1H	CAN message event pointer 381	M_EVT381	R/W		x		Undefined
xxxxn4C2H	CAN message event pointer 382	M_EVT382	R/W		x		Undefined
xxxxn4C3H	CAN message event pointer 383	M_EVT383	R/W		x		Undefined
xxxxn4C4H	CAN message data length register 38	M_DLC38	R/W		x		Undefined
xxxxn4C5H	CAN message control register 38	M_DTRL38	R/W		x		Undefined
xxxxn4C6H	CAN message time stamp register 38	M_TIME38	R/W			x	Undefined
xxxxn4C8H	CAN message data register 380	M_DATA380	R/W		x		Undefined
xxxxn4C9H	CAN message data register 381	M_DATA381	R/W		x		Undefined
xxxxn4CAH	CAN message data register 382	M_DATA382	R/W		x		Undefined
xxxxn4CBH	CAN message data register 383	M_DATA383	R/W		x		Undefined
xxxxn4CCH	CAN message data register 384	M_DATA384	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 19 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn4CDH	CAN message data register 385	M_DATA385	R/W		x		Undefined
xxxxn4CEH	CAN message data register 386	M_DATA386	R/W		x		Undefined
xxxxn4CFH	CAN message data register 387	M_DATA387	R/W		x		Undefined
xxxxn4D0H	CAN message ID register L38	M_IDL38	R/W			x	Undefined
xxxxn4D2H	CAN message ID register H38	M_IDH38	R/W			x	Undefined
xxxxn4D4H	CAN message configuration register 38	M_CONF38	R/W		x		Undefined
xxxxn4D5H	CAN message status register 38	M_STAT38	R		x		Undefined
xxxxn4D6H	CAN status set/cancel register 38	SC_STAT38	W			x	0000H
xxxxn4E0H	CAN message event pointer 390	M_EVT390	R/W		x		Undefined
xxxxn4E1H	CAN message event pointer 391	M_EVT391	R/W		x		Undefined
xxxxn4E2H	CAN message event pointer 392	M_EVT392	R/W		x		Undefined
xxxxn4E3H	CAN message event pointer 393	M_EVT393	R/W		x		Undefined
xxxxn4E4H	CAN message data length register 39	M_DLC39	R/W		x		Undefined
xxxxn4E5H	CAN message control register 39	M_CTRL39	R/W		x		Undefined
xxxxn4E6H	CAN message time stamp register 39	M_TIME39	R/W			x	Undefined
xxxxn4E8H	CAN message data register 390	M_DATA390	R/W		x		Undefined
xxxxn4E9H	CAN message data register 391	M_DATA391	R/W		x		Undefined
xxxxn4EAH	CAN message data register 392	M_DATA392	R/W		x		Undefined
xxxxn4EBH	CAN message data register 393	M_DATA393	R/W		x		Undefined
xxxxn4ECH	CAN message data register 394	M_DATA394	R/W		x		Undefined
xxxxn4EDH	CAN message data register 395	M_DATA395	R/W		x		Undefined
xxxxn4EEH	CAN message data register 396	M_DATA396	R/W		x		Undefined
xxxxn4EFH	CAN message data register 397	M_DATA397	R/W		x		Undefined
xxxxn4F0H	CAN message ID register L39	M_IDL39	R/W			x	Undefined
xxxxn4F2H	CAN message ID register H39	M_IDH39	R/W			x	Undefined
xxxxn4F4H	CAN message configuration register 39	M_CONF39	R/W		x		Undefined
xxxxn4F5H	CAN message status register 39	M_STAT39	R		x		Undefined
xxxxn4F6H	CAN status set/cancel register 39	SC_STAT39	W			x	0000H
xxxxn500H	CAN message event pointer 400	M_EVT400	R/W		x		Undefined
xxxxn501H	CAN message event pointer 401	M_EVT401	R/W		x		Undefined
xxxxn502H	CAN message event pointer 402	M_EVT402	R/W		x		Undefined
xxxxn503H	CAN message event pointer 403	M_EVT403	R/W		x		Undefined
xxxxn504H	CAN message data length register 40	M_DLC40	R/W		x		Undefined
xxxxn505H	CAN message control register 40	M_CTRL40	R/W		x		Undefined
xxxxn506H	CAN message time stamp register 40	M_TIME40	R/W			x	Undefined
xxxxn508H	CAN message data register 400	M_DATA400	R/W		x		Undefined
xxxxn509H	CAN message data register 401	M_DATA401	R/W		x		Undefined
xxxxn50AH	CAN message data register 402	M_DATA402	R/W		x		Undefined
xxxxn50BH	CAN message data register 403	M_DATA403	R/W		x		Undefined
xxxxn50CH	CAN message data register 404	M_DATA404	R/W		x		Undefined
xxxxn50DH	CAN message data register 405	M_DATA405	R/W		x		Undefined
xxxxn50EH	CAN message data register 406	M_DATA406	R/W		x		Undefined
xxxxn50FH	CAN message data register 407	M_DATA407	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 20 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn510H	CAN message ID register L40	M_IDL40	R/W			x	Undefined
xxxxn512H	CAN message ID register H40	M_IDH40	R/W			x	Undefined
xxxxn514H	CAN message configuration register 40	M_CONF40	R/W		x		Undefined
xxxxn515H	CAN message status register 40	M_STAT40	R		x		Undefined
xxxxn516H	CAN status set/cancel register 40	SC_STAT40	W			x	0000H
xxxxn520H	CAN message event pointer 410	M_EVT410	R/W		x		Undefined
xxxxn521H	CAN message event pointer 411	M_EVT411	R/W		x		Undefined
xxxxn522H	CAN message event pointer 412	M_EVT412	R/W		x		Undefined
xxxxn523H	CAN message event pointer 413	M_EVT413	R/W		x		Undefined
xxxxn524H	CAN message data length register 41	M_DLC41	R/W		x		Undefined
xxxxn525H	CAN message control register 41	M_CTRL41	R/W		x		Undefined
xxxxn526H	CAN message time stamp register 41	M_TIME41	R/W			x	Undefined
xxxxn528H	CAN message data register 410	M_DATA410	R/W		x		Undefined
xxxxn529H	CAN message data register 411	M_DATA411	R/W		x		Undefined
xxxxn52AH	CAN message data register 412	M_DATA412	R/W		x		Undefined
xxxxn52BH	CAN message data register 413	M_DATA413	R/W		x		Undefined
xxxxn52CH	CAN message data register 414	M_DATA414	R/W		x		Undefined
xxxxn52DH	CAN message data register 415	M_DATA415	R/W		x		Undefined
xxxxn52EH	CAN message data register 416	M_DATA416	R/W		x		Undefined
xxxxn52FH	CAN message data register 417	M_DATA417	R/W		x		Undefined
xxxxn530H	CAN message ID register L41	M_IDL41	R/W			x	Undefined
xxxxn532H	CAN message ID register H41	M_IDH41	R/W			x	Undefined
xxxxn534H	CAN message configuration register 41	M_CONF41	R/W		x		Undefined
xxxxn535H	CAN message status register 41	M_STAT41	R		x		Undefined
xxxxn536H	CAN status set/cancel register 41	SC_STAT41	W			x	0000H
xxxxn540H	CAN message event pointer420	M_EVT420	R/W		x		Undefined
xxxxn541H	CAN message event pointer 421	M_EVT421	R/W		x		Undefined
xxxxn542H	CAN message event pointer 422	M_EVT422	R/W		x		Undefined
xxxxn543H	CAN message event pointer423	M_EVT423	R/W		x		Undefined
xxxxn544H	CAN message data length register 42	M_DLC42	R/W		x		Undefined
xxxxn545H	CAN message control register 42	M_CTRL42	R/W		x		Undefined
xxxxn546H	CAN message time stamp register 42	M_TIME42	R/W			x	Undefined
xxxxn548H	CAN message data register 420	M_DATA420	R/W		x		Undefined
xxxxn549H	CAN message data register 421	M_DATA421	R/W		x		Undefined
xxxxn54AH	CAN message data register 422	M_DATA422	R/W		x		Undefined
xxxxn54BH	CAN message data register 423	M_DATA423	R/W		x		Undefined
xxxxn54CH	CAN message data register 424	M_DATA424	R/W		x		Undefined
xxxxn54DH	CAN message data register 425	M_DATA425	R/W		x		Undefined
xxxxn54EH	CAN message data register 426	M_DATA426	R/W		x		Undefined
xxxxn54FH	CAN message data register 427	M_DATA427	R/W		x		Undefined
xxxxn550H	CAN message ID register L42	M_IDL42	R/W			x	Undefined
xxxxn552H	CAN message ID register H42	M_IDH42	R/W			x	Undefined
xxxxn554H	CAN message configuration register 42	M_CONF42	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 21 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn555H	CAN message status register 42	M_STAT42	R		x		Undefined
xxxxn556H	CAN status set/cancel register 42	SC_STAT42	W			x	0000H
xxxxn560H	CAN message event pointer 430	M_EVT430	R/W		x		Undefined
xxxxn561H	CAN message event pointer431	M_EVT431	R/W		x		Undefined
xxxxn562H	CAN message event pointer 432	M_EVT432	R/W		x		Undefined
xxxxn563H	CAN message event pointer 433	M_EVT433	R/W		x		Undefined
xxxxn564H	CAN message data length register 43	M_DLC43	R/W		x		Undefined
xxxxn565H	CAN message control register 43	M_CTRL43	R/W		x		Undefined
xxxxn566H	CAN message time stamp register 43	M_TIME43	R/W			x	Undefined
xxxxn568H	CAN message data register 430	M_DATA430	R/W		x		Undefined
xxxxn569H	CAN message data register 431	M_DATA431	R/W		x		Undefined
xxxxn56AH	CAN message data register 432	M_DATA432	R/W		x		Undefined
xxxxn56BH	CAN message data register 433	M_DATA433	R/W		x		Undefined
xxxxn56CH	CAN message data register 434	M_DATA434	R/W		x		Undefined
xxxxn56DH	CAN message data register 435	M_DATA435	R/W		x		Undefined
xxxxn56EH	CAN message data register 436	M_DATA436	R/W		x		Undefined
xxxxn56FH	CAN message data register 437	M_DATA437	R/W		x		Undefined
xxxxn570H	CAN message ID register L43	M_IDL43	R/W			x	Undefined
xxxxn572H	CAN message ID register H43	M_IDH43	R/W			x	Undefined
xxxxn574H	CAN message configuration register 43	M_CONF43	R/W		x		Undefined
xxxxn575H	CAN message status register 43	M_STAT43	R		x		Undefined
xxxxn576H	CAN status set/cancel register 43	SC_STAT43	W			x	0000H
xxxxn580H	CAN message event pointer 440	M_EVT440	R/W		x		Undefined
xxxxn581H	CAN message event pointer 441	M_EVT441	R/W		x		Undefined
xxxxn582H	CAN message event pointer 442	M_EVT442	R/W		x		Undefined
xxxxn583H	CAN message event pointer 443	M_EVT43	R/W		x		Undefined
xxxxn584H	CAN message data length register 44	M_DLC44	R/W		x		Undefined
xxxxn585H	CAN message control register 44	M_CTRL44	R/W		x		Undefined
xxxxn586H	CAN message time stamp register 44	M_TIME44	R/W			x	Undefined
xxxxn588H	CAN message data register 440	M_DATA440	R/W		x		Undefined
xxxxn589H	CAN message data register 441	M_DATA441	R/W		x		Undefined
xxxxn58AH	CAN message data register 442	M_DATA442	R/W		x		Undefined
xxxxn58BH	CAN message data register 443	M_DATA443	R/W		x		Undefined
xxxxn58CH	CAN message data register 444	M_DATA444	R/W		x		Undefined
xxxxn58DH	CAN message data register 445	M_DATA445	R/W		x		Undefined
xxxxn58EH	CAN message data register 446	M_DATA446	R/W		x		Undefined
xxxxn58FH	CAN message data register 447	M_DATA447	R/W		x		Undefined
xxxxn590H	CAN message ID register L44	M_IDL44	R/W			x	Undefined
xxxxn592H	CAN message ID register H44	M_IDH44	R/W			x	Undefined
xxxxn594H	CAN message configuration register 44	M_CONF44	R/W		x		Undefined
xxxxn595H	CAN message status register 44	M_STAT44	R		x		Undefined
xxxxn596H	CAN status set/cancel register 44	SC_STAT44	W			x	0000H
xxxxn5A0H	CAN message event pointer 450	M_EVT450	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 22 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn5A1H	CAN message event pointer 451	M_EVT451	R/W		x		Undefined
xxxxn5A2H	CAN message event pointer 452	M_EVT452	R/W		x		Undefined
xxxxn5A3H	CAN message event pointer 453	M_EVT453	R/W		x		Undefined
xxxxn5A4H	CAN message data length register 45	M_DLC45	R/W		x		Undefined
xxxxn5A5H	CAN message control register 45	M_CTRL45	R/W		x		Undefined
xxxxn5A6H	CAN message time stamp register 45	M_TIME45	R/W			x	Undefined
xxxxn5A8H	CAN message data register 450	M_DATA450	R/W		x		Undefined
xxxxn5A9H	CAN message data register 451	M_DATA451	R/W		x		Undefined
xxxxn5AAH	CAN message data register 452	M_DATA452	R/W		x		Undefined
xxxxn5ABH	CAN message data register 453	M_DATA453	R/W		x		Undefined
xxxxn5ACH	CAN message data register 454	M_DATA454	R/W		x		Undefined
xxxxn5ADH	CAN message data register 455	M_DATA455	R/W		x		Undefined
xxxxn5AEH	CAN message data register 456	M_DATA456	R/W		x		Undefined
xxxxn5AFH	CAN message data register 457	M_DATA457	R/W		x		Undefined
xxxxn5B0H	CAN message ID register L45	M_IDL45	R/W			x	Undefined
xxxxn5B2H	CAN message ID register H45	M_IDH45	R/W			x	Undefined
xxxxn5B4H	CAN message configuration register 45	M_CONF45	R/W		x		Undefined
xxxxn5B5H	CAN message status register 45	M_STAT45	R		x		Undefined
xxxxn5B6H	CAN status set/cancel register 45	SC_STAT45	W			x	0000H
xxxxn5C0H	CAN message event pointer 460	M_EVT460	R/W		x		Undefined
xxxxn5C1H	CAN message event pointer 461	M_EVT461	R/W		x		Undefined
xxxxn5C2H	CAN message event pointer 462	M_EVT462	R/W		x		Undefined
xxxxn5C3H	CAN message event pointer 463	M_EVT463	R/W		x		Undefined
xxxxn5C4H	CAN message data length register 46	M_DLC46	R/W		x		Undefined
xxxxn5C5H	CAN message control register 46	M_CTRL46	R/W		x		Undefined
xxxxn5C6H	CAN message time stamp register 46	M_TIME46	R/W			x	Undefined
xxxxn5C8H	CAN message data register 460	M_DATA460	R/W		x		Undefined
xxxxn5C9H	CAN message data register 461	M_DATA461	R/W		x		Undefined
xxxxn5CAH	CAN message data register 462	M_DATA462	R/W		x		Undefined
xxxxn5CBH	CAN message data register 463	M_DATA463	R/W		x		Undefined
xxxxn5CCH	CAN message data register 464	M_DATA464	R/W		x		Undefined
xxxxn5CDH	CAN message data register 465	M_DATA465	R/W		x		Undefined
xxxxn5CEH	CAN message data register 466	M_DATA466	R/W		x		Undefined
xxxxn5CFH	CAN message data register 467	M_DATA467	R/W		x		Undefined
xxxxn5D0H	CAN message ID register L46	M_IDL46	R/W			x	Undefined
xxxxn5D2H	CAN message ID register H46	M_IDH46	R/W			x	Undefined
xxxxn5D4H	CAN message configuration register 46	M_CONF46	R/W		x		Undefined
xxxxn5D5H	CAN message status register 46	M_STAT46	R		x		Undefined
xxxxn5D6H	CAN status set/cancel register 46	SC_STAT46	W			x	0000H
xxxxn5E0H	CAN message event pointer 470	M_EVT470	R/W		x		Undefined
xxxxn5E1H	CAN message event pointer 471	M_EVT471	R/W		x		Undefined
xxxxn5E2H	CAN message event pointer 472	M_EVT472	R/W		x		Undefined
xxxxn5E3H	CAN message event pointer 473	M_EVT473	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 23 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn5E4H	CAN message data length register 47	M_DLC47	R/W		x		Undefined
Xxxxn5E5H	CAN message control register 47	M_CTRL47	R/W		x		Undefined
Xxxxn5E6H	CAN message time stamp register 47	M_TIME47	R/W			x	Undefined
xxxxn5E8H	CAN message data register 470	M_DATA470	R/W		x		Undefined
xxxxn5E9H	CAN message data register 471	M_DATA471	R/W		x		Undefined
xxxxn5EAH	CAN message data register 472	M_DATA472	R/W		x		Undefined
xxxxn5EBH	CAN message data register 473	M_DATA473	R/W		x		Undefined
xxxxn5ECH	CAN message data register 474	M_DATA474	R/W		x		Undefined
xxxxn5EDH	CAN message data register 475	M_DATA475	R/W		x		Undefined
xxxxn5EEH	CAN message data register 476	M_DATA476	R/W		x		Undefined
xxxxn5EFH	CAN message data register 477	M_DATA477	R/W		x		Undefined
xxxxn5F0H	CAN message ID register L47	M_IDL47	R/W			x	Undefined
xxxxn5F2H	CAN message ID register H47	M_IDH47	R/W			x	Undefined
xxxxn5F4H	CAN message configuration register 47	M_CONF47	R/W		x		Undefined
xxxxn5F5H	CAN message status register 47	M_STAT47	R		x		Undefined
xxxxn5F6H	CAN status set/cancel register 47	SC_STAT47	W			x	0000H
xxxxn600H	CAN message event pointer 480	M_EVT480	R/W		x		Undefined
xxxxn601H	CAN message event pointer 481	M_EVT481	R/W		x		Undefined
xxxxn602H	CAN message event pointer 482	M_EVT482	R/W		x		Undefined
xxxxn603H	CAN message event pointer 483	M_EVT483	R/W		x		Undefined
xxxxn604H	CAN message data length register 48	M_DLC48	R/W		x		Undefined
xxxxn605H	CAN message control register 48	M_CTRL48	R/W		x		Undefined
xxxxn606H	CAN message time stamp register 48	M_TIME48	R/W			x	Undefined
xxxxn608H	CAN message data register 480	M_DATA480	R/W		x		Undefined
xxxxn609H	CAN message data register 481	M_DATA481	R/W		x		Undefined
xxxxn60AH	CAN message data register 482	M_DATA482	R/W		x		Undefined
xxxxn60BH	CAN message data register 483	M_DATA483	R/W		x		Undefined
xxxxn60CH	CAN message data register 484	M_DATA484	R/W		x		Undefined
xxxxn60DH	CAN message data register 485	M_DATA485	R/W		x		Undefined
xxxxn60EH	CAN message data register 486	M_DATA486	R/W		x		Undefined
xxxxn60FH	CAN message data register 487	M_DATA487	R/W		x		Undefined
xxxxn610H	CAN message ID register L48	M_IDL48	R/W			x	Undefined
xxxxn612H	CAN message ID register H48	M_IDH48	R/W			x	Undefined
xxxxn614H	CAN message configuration register 48	M_CONF48	R/W		x		Undefined
xxxxn615H	CAN message status register 48	M_STAT48	R		x		Undefined
xxxxn616H	CAN status set/cancel register 48	SC_STAT48	W			x	0000H
xxxxn620H	CAN message event pointer 490	M_EVT490	R/W		x		Undefined
xxxxn621H	CAN message event pointer 491	M_EVT491	R/W		x		Undefined
xxxxn622H	CAN message event pointer 492	M_EVT492	R/W		x		Undefined
xxxxn623H	CAN message event pointer 493	M_EVT493	R/W		x		Undefined
xxxxn624H	CAN message data length register 49	M_DLC49	R/W		x		Undefined
xxxxn625H	CAN message control register 49	M_CTRL49	R/W		x		Undefined
xxxxn626H	CAN message time stamp register 49	M_TIME49	R/W			x	Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 24 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn628H	CAN message data register 490	M_DATA490	R/W		x		Undefined
xxxxn629H	CAN message data register 491	M_DATA491	R/W		x		Undefined
xxxxn62AH	CAN message data register 492	M_DATA492	R/W		x		Undefined
xxxxn62BH	CAN message data register 493	M_DATA493	R/W		x		Undefined
xxxxn62CH	CAN message data register 494	M_DATA494	R/W		x		Undefined
xxxxn62DH	CAN message data register 495	M_DATA495	R/W		x		Undefined
xxxxn62EH	CAN message data register 496	M_DATA496	R/W		x		Undefined
xxxxn62FH	CAN message data register 497	M_DATA497	R/W		x		Undefined
xxxxn630H	CAN message ID register L49	M_IDL49	R/W			x	Undefined
xxxxn632H	CAN message ID register H49	M_IDH49	R/W			x	Undefined
xxxxn634H	CAN message configuration register 49	M_CONF49	R/W		x		Undefined
xxxxn635H	CAN message status register 49	M_STAT49	R		x		Undefined
xxxxn636H	CAN status set/cancel register 49	SC_STAT49	W			x	0000H
xxxxn640H	CAN message event pointer 500	M_EVT500	R/W		x		Undefined
xxxxn641H	CAN message event pointer 501	M_EVT501	R/W		x		Undefined
xxxxn642H	CAN message event pointer 502	M_EVT502	R/W		x		Undefined
xxxxn643H	CAN message event pointer 503	M_EVT503	R/W		x		Undefined
xxxxn644H	CAN message data length register 508	M_DLC50	R/W		x		Undefined
xxxxn645H	CAN message control register 50	M_CTRL50	R/W		x		Undefined
xxxxn646H	CAN message time stamp register 50	M_TIME50	R/W			x	Undefined
xxxxn648H	CAN message data register 500	M_DATA500	R/W		x		Undefined
xxxxn649H	CAN message data register 501	M_DATA501	R/W		x		Undefined
xxxxn64AH	CAN message data register 502	M_DATA502	R/W		x		Undefined
xxxxn64BH	CAN message data register 503	M_DATA503	R/W		x		Undefined
xxxxn64CH	CAN message data register 504	M_DATA504	R/W		x		Undefined
xxxxn64DH	CAN message data register 505	M_DATA505	R/W		x		Undefined
xxxxn64EH	CAN message data register 506	M_DATA506	R/W		x		Undefined
xxxxn64FH	CAN message data register 507	M_DATA507	R/W		x		Undefined
xxxxn650H	CAN message ID register L50	M_IDL50	R/W			x	Undefined
xxxxn652H	CAN message ID register H50	M_IDH50	R/W			x	Undefined
xxxxn654H	CAN message configuration register 50	M_CONF50	R/W		x		Undefined
xxxxn655H	CAN message status register 50	M_STAT50	R		x		Undefined
xxxxn656H	CAN status set/cancel register 50	SC_STAT50	W			x	0000H
xxxxn660H	CAN message event pointer 510	M_EVT510	R/W		x		Undefined
xxxxn661H	CAN message event pointer 511	M_EVT511	R/W		x		Undefined
xxxxn662H	CAN message event pointer 512	M_EVT512	R/W		x		Undefined
xxxxn663H	CAN message event pointer 513	M_EVT513	R/W		x		Undefined
xxxxn664H	CAN message data length register 51	M_DLC51	R/W		x		Undefined
xxxxn665H	CAN message control register 51	M_CTRL51	R/W		x		Undefined
xxxxn666H	CAN message time stamp register 51	M_TIME51	R/W			x	Undefined
xxxxn668H	CAN message data register 510	M_DATA510	R/W		x		Undefined
xxxxn669H	CAN message data register 511	M_DATA511	R/W		x		Undefined
xxxxn66AH	CAN message data register 512	M_DATA512	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 25 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn66BH	CAN message data register 513	M_DATA513	R/W		x		Undefined
xxxxn66CH	CAN message data register 514	M_DATA514	R/W		x		Undefined
xxxxn66DH	CAN message data register 515	M_DATA515	R/W		x		Undefined
xxxxn66EH	CAN message data register 516	M_DATA516	R/W		x		Undefined
xxxxn66FH	CAN message data register 517	M_DATA517	R/W		x		Undefined
xxxxn670H	CAN message ID register L51	M_IDL51	R/W			x	Undefined
xxxxn672H	CAN message ID register H51	M_IDH51	R/W			x	Undefined
xxxxn674H	CAN message configuration register 51	M_CONF51	R/W		x		Undefined
xxxxn675H	CAN message status register 51	M_STAT50	R		x		Undefined
xxxxn676H	CAN status set/cancel register 51	SC_STAT51	W			x	0000H
xxxxn680H	CAN message event pointer 520	M_EVT520	R/W		x		Undefined
xxxxn681H	CAN message event pointer 521	M_EVT521	R/W		x		Undefined
xxxxn682H	CAN message event pointer 522	M_EVT522	R/W		x		Undefined
xxxxn683H	CAN message event pointer 523	M_EVT523	R/W		x		Undefined
xxxxn684H	CAN message data length register 52	M_DLC52	R/W		x		Undefined
xxxxn685H	CAN message control register 52	M_CTRL52	R/W		x		Undefined
xxxxn686H	CAN message time stamp register 52	M_TIME52	R/W			x	Undefined
xxxxn688H	CAN message data register 520	M_DATA520	R/W		x		Undefined
xxxxn689H	CAN message data register 521	M_DATA521	R/W		x		Undefined
xxxxn68AH	CAN message data register 512	M_DATA522	R/W		x		Undefined
xxxxn68BH	CAN message data register 523	M_DATA523	R/W		x		Undefined
xxxxn68CH	CAN message data register 524	M_DATA524	R/W		x		Undefined
xxxxn68DH	CAN message data register 525	M_DATA525	R/W		x		Undefined
xxxxn68EH	CAN message data register 526	M_DATA526	R/W		x		Undefined
xxxxn68FH	CAN message data register 527	M_DATA527	R/W		x		Undefined
xxxxn690H	CAN message ID register L52	M_IDL52	R/W			x	Undefined
xxxxn692H	CAN message ID register H52	M_IDH52	R/W			x	Undefined
xxxxn694H	CAN message configuration register 52	M_CONF52	R/W		x		Undefined
xxxxn695H	CAN message status register 52	M_STAT52	R		x		Undefined
xxxxn696H	CAN status set/cancel register 52	SC_STAT52	W			x	0000H
xxxxn6A0H	CAN message event pointer 530	M_EVT530	R/W		x		Undefined
xxxxn6A1H	CAN message event pointer 531	M_EVT531	R/W		x		Undefined
xxxxn62AH	CAN message event pointer 532	M_EVT532	R/W		x		Undefined
xxxxn6A3H	CAN message event pointer 533	M_EVT533	R/W		x		Undefined
xxxxn6A4H	CAN message data length register 53	M_DLC53	R/W		x		Undefined
xxxxn6A5H	CAN message control register 53	M_CTRL53	R/W		x		Undefined
xxxxn6A6H	CAN message time stamp register 53	M_TIME53	R/W			x	Undefined
xxxxn6A8H	CAN message data register 530	M_DATA530	R/W		x		Undefined
xxxxn6A9H	CAN message data register 531	M_DATA531	R/W		x		Undefined
xxxxn6AAH	CAN message data register 532	M_DATA532	R/W		x		Undefined
xxxxn6ABH	CAN message data register 533	M_DATA533	R/W		x		Undefined
xxxxn6ACH	CAN message data register 534	M_DATA534	R/W		x		Undefined
xxxxn6ADH	CAN message data register 535	M_DATA535	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 26 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn6AEH	CAN message data register 536	M_DATA536	R/W		x		Undefined
xxxxn6AFH	CAN message data register 537	M_DATA537	R/W		x		Undefined
xxxxn6B0H	CAN message ID register L53	M_IDL53	R/W			x	Undefined
xxxxn6B2H	CAN message ID register H53	M_IDH53	R/W			x	Undefined
xxxxn6B4H	CAN message configuration register 53	M_CONF53	R/W		x		Undefined
xxxxn6B5H	CAN message status register 53	M_STAT53	R		x		Undefined
xxxxn6B6H	CAN status set/cancel register 53	SC_STAT53	W			x	0000H
xxxxn6C0H	CAN message event pointer 540	M_EVT540	R/W		x		Undefined
xxxxn6C1H	CAN message event pointer 541	M_EVT541	R/W		x		Undefined
xxxxn6C2H	CAN message event pointer 542	M_EVT542	R/W		x		Undefined
xxxxn6C3H	CAN message event pointer 543	M_EVT543	R/W		x		Undefined
xxxxn6C4H	CAN message data length register 54	M_DLC54	R/W		x		Undefined
xxxxn6C5H	CAN message control register 54	M_CTRL54	R/W		x		Undefined
xxxxn6C6H	CAN message time stamp register 54	M_TIME54	R/W			x	Undefined
xxxxn6C8H	CAN message data register 540	M_DATA540	R/W		x		Undefined
xxxxn6C9H	CAN message data register 541	M_DATA541	R/W		x		Undefined
xxxxn6CAH	CAN message data register 542	M_DATA542	R/W		x		Undefined
xxxxn6CBH	CAN message data register 543	M_DATA543	R/W		x		Undefined
xxxxn6CCH	CAN message data register 544	M_DATA544	R/W		x		Undefined
xxxxn6CDH	CAN message data register 545	M_DATA545	R/W		x		Undefined
xxxxn6CEH	CAN message data register 546	M_DATA546	R/W		x		Undefined
xxxxn6CFH	CAN message data register 547	M_DATA547	R/W		x		Undefined
xxxxn6D0H	CAN message ID register L54	M_IDL54	R/W			x	Undefined
xxxxn6D2H	CAN message ID register H54	M_IDH54	R/W			x	Undefined
xxxxn6D4H	CAN message configuration register 54	M_CONF54	R/W		x		Undefined
xxxxn6D5H	CAN message status register 54	M_STAT54	R		x		Undefined
xxxxn6D6H	CAN status set/cancel register 54	SC_STAT54	W			x	0000H
xxxxn6E0H	CAN message event pointer 550	M_EVT550	R/W		x		Undefined
xxxxn6E1H	CAN message event pointer 551	M_EVT551	R/W		x		Undefined
xxxxn6EH	CAN message event pointer 552	M_EVT552	R/W		x		Undefined
xxxxn6E3H	CAN message event pointer 553	M_EVT553	R/W		x		Undefined
xxxxn6E4H	CAN message data length register 55	M_DLC55	R/W		x		Undefined
xxxxn6E5H	CAN message control register 55	M_CTRL55	R/W		x		Undefined
xxxxn6E6H	CAN message time stamp register 55	M_TIME55	R/W			x	Undefined
xxxxn6E8H	CAN message data register 550	M_DATA550	R/W		x		Undefined
xxxxn6E9H	CAN message data register 551	M_DATA551	R/W		x		Undefined
xxxxn6EAH	CAN message data register 552	M_DATA552	R/W		x		Undefined
xxxxn6EBH	CAN message data register 553	M_DATA553	R/W		x		Undefined
xxxxn6ECH	CAN message data register 554	M_DATA554	R/W		x		Undefined
xxxxn6EDH	CAN message data register 555	M_DATA555	R/W		x		Undefined
xxxxn6EEH	CAN message data register 556	M_DATA556	R/W		x		Undefined
xxxxn6EFH	CAN message data register 557	M_DATA557	R/W		x		Undefined
xxxxn6F0H	CAN message ID register L55	M_IDL55	R/W			x	Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 27 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn6F2H	CAN message ID register H55	M_IDH55	R/W			x	Undefined
xxxxn6F4H	CAN message configuration register 55	M_CONF55	R/W		x		Undefined
xxxxn6F5H	CAN message status register 55	M_STAT55	R		x		Undefined
xxxxn6F6H	CAN status set/cancel register 55	SC_STAT55	W			x	0000H
xxxxn700H	CAN message event pointer 560	M_EVT560	R/W		x		Undefined
xxxxn701H	CAN message event pointer 561	M_EVT561	R/W		x		Undefined
xxxxn702H	CAN message event pointer 562	M_EVT562	R/W		x		Undefined
xxxxn703H	CAN message event pointer 563	M_EVT563	R/W		x		Undefined
xxxxn704H	CAN message data length register 56	M_DLC56	R/W		x		Undefined
xxxxn705H	CAN message control register 56	M_CTRL56	R/W		x		Undefined
xxxxn706H	CAN message time stamp register 56	M_TIME56	R/W			x	Undefined
xxxxn708H	CAN message data register 560	M_DATA560	R/W		x		Undefined
xxxxn709H	CAN message data register 561	M_DATA561	R/W		x		Undefined
xxxxn70AH	CAN message data register 562	M_DATA562	R/W		x		Undefined
xxxxn70BH	CAN message data register 563	M_DATA563	R/W		x		Undefined
xxxxn70CH	CAN message data register 564	M_DATA564	R/W		x		Undefined
xxxxn70DH	CAN message data register 565	M_DATA565	R/W		x		Undefined
xxxxn70EH	CAN message data register 566	M_DATA566	R/W		x		Undefined
xxxxn70FH	CAN message data register 567	M_DATA567	R/W		x		Undefined
xxxxn710H	CAN message ID register L56	M_IDL56	R/W			x	Undefined
xxxxn712H	CAN message ID register H56	M_IDH56	R/W			x	Undefined
xxxxn714H	CAN message configuration register 56	M_CONF56	R/W		x		Undefined
xxxxn715H	CAN message status register 56	M_STAT56	R		x		Undefined
xxxxn716H	CAN status set/cancel register 56	SC_STAT56	W			x	0000H
xxxxn720H	CAN message event pointer 570	M_EVT570	R/W		x		Undefined
xxxxn721H	CAN message event pointer 571	M_EVT571	R/W		x		Undefined
xxxxn722H	CAN message event pointer 572	M_EVT572	R/W		x		Undefined
xxxxn723H	CAN message event pointer 573	M_EVT573	R/W		x		Undefined
xxxxn724H	CAN message data length register 57	M_DLC57	R/W		x		Undefined
xxxxn725H	CAN message control register 57	M_CTRL57	R/W		x		Undefined
xxxxn726H	CAN message time stamp register 57	M_TIME57	R/W			x	Undefined
xxxxn728H	CAN message data register 570	M_DATA570	R/W		x		Undefined
xxxxn729H	CAN message data register 571	M_DATA571	R/W		x		Undefined
xxxxn72AH	CAN message data register 572	M_DATA572	R/W		x		Undefined
xxxxn72BH	CAN message data register 573	M_DATA573	R/W		x		Undefined
xxxxn72CH	CAN message data register 574	M_DATA574	R/W		x		Undefined
xxxxn72DH	CAN message data register 575	M_DATA575	R/W		x		Undefined
xxxxn72EH	CAN message data register 576	M_DATA576	R/W		x		Undefined
xxxxn72FH	CAN message data register 577	M_DATA577	R/W		x		Undefined
xxxxn730H	CAN message ID register L57	M_IDL57	R/W			x	Undefined
xxxxn732H	CAN message ID register H57	M_IDH57	R/W			x	Undefined
xxxxn734H	CAN message configuration register 57	M_CONF57	R/W		x		Undefined
xxxxn735H	CAN message status register 57	M_STAT57	R		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 28 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn736H	CAN status set/cancel register 57	SC_STAT57	W			x	0000H
xxxxn740H	CAN message event pointer 580	M_EVT580	R/W		x		Undefined
xxxxn741H	CAN message event pointer581	M_EVT581	R/W		x		Undefined
xxxxn742H	CAN message event pointer 582	M_EVT582	R/W		x		Undefined
xxxxn743H	CAN message event pointer 583	M_EVT583	R/W		x		Undefined
xxxxn744H	CAN message data length register 58	M_DLC58	R/W		x		Undefined
xxxxn745H	CAN message control register 58	M_CTRL58	R/W		x		Undefined
xxxxn746H	CAN message time stamp register 58	M_TIME58	R/W			x	Undefined
xxxxn748H	CAN message data register 580	M_DATA580	R/W		x		Undefined
xxxxn749H	CAN message data register 581	M_DATA581	R/W		x		Undefined
xxxxn74AH	CAN message data register 582	M_DATA582	R/W		x		Undefined
xxxxn74BH	CAN message data register 583	M_DATA583	R/W		x		Undefined
xxxxn74CH	CAN message data register 584	M_DATA584	R/W		x		Undefined
xxxxn74DH	CAN message data register 585	M_DATA585	R/W		x		Undefined
xxxxn74EH	CAN message data register 586	M_DATA586	R/W		x		Undefined
xxxxn74FH	CAN message data register 587	M_DATA587	R/W		x		Undefined
xxxxn750H	CAN message ID register L58	M_IDL58	R/W			x	Undefined
xxxxn752H	CAN message ID register H58	M_IDH58	R/W			x	Undefined
xxxxn754H	CAN message configuration register 58	M_CONF58	R/W		x		Undefined
xxxxn755H	CAN message status register 58	M_STAT58	R		x		Undefined
xxxxn756H	CAN status set/cancel register 58	SC_STAT58	W			x	0000H
xxxxn760H	CAN message event pointer 590	M_EVT590	R/W		x		Undefined
xxxxn761H	CAN message event pointer 591	M_EVT591	R/W		x		Undefined
xxxxn762H	CAN message event pointer 592	M_EVT592	R/W		x		Undefined
xxxxn763H	CAN message event pointer 593	M_EVT593	R/W		x		Undefined
xxxxn764H	CAN message data length register 59	M_DLC59	R/W		x		Undefined
xxxxn765H	CAN message control register 59	M_CTRL59	R/W		x		Undefined
xxxxn766H	CAN message time stamp register 59	M_TIME59	R/W			x	Undefined
xxxxn768H	CAN message data register 590	M_DATA590	R/W		x		Undefined
xxxxn769H	CAN message data register 591	M_DATA591	R/W		x		Undefined
xxxxn76AH	CAN message data register 592	M_DATA592	R/W		x		Undefined
xxxxn76BH	CAN message data register 593	M_DATA593	R/W		x		Undefined
xxxxn76CH	CAN message data register 594	M_DATA594	R/W		x		Undefined
xxxxn76DH	CAN message data register 595	M_DATA595	R/W		x		Undefined
xxxxn76EH	CAN message data register 596	M_DATA596	R/W		x		Undefined
xxxxn76FH	CAN message data register 597	M_DATA597	R/W		x		Undefined
xxxxn770H	CAN message ID register L59	M_IDL59	R/W			x	Undefined
xxxxn772H	CAN message ID register H59	M_IDH59	R/W			x	Undefined
xxxxn774H	CAN message configuration register 59	M_CONF59	R/W		x		Undefined
xxxxn775H	CAN message status register 59	M_STAT59	R		x		Undefined
xxxxn776H	CAN status set/cancel register 59	SC_STAT59	W			x	0000H
xxxxn780H	CAN message event pointer 600	M_EVT600	R/W		x		Undefined
xxxxn781H	CAN message event pointer 601	M_EVT601	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 29 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn782H	CAN message event pointer 602	M_EVT602	R/W		x		Undefined
xxxxn783H	CAN message event pointer 603	M_EVT603	R/W		x		Undefined
xxxxn784H	CAN message data length register 60	M_DLC60	R/W		x		Undefined
xxxxn785H	CAN message control register 60	M_CTRL60	R/W		x		Undefined
xxxxn786H	CAN message time stamp register 60	M_TIME60	R/W			x	Undefined
xxxxn788H	CAN message data register 600	M_DATA600	R/W		x		Undefined
xxxxn789H	CAN message data register 601	M_DATA601	R/W		x		Undefined
xxxxn78AH	CAN message data register 602	M_DATA602	R/W		x		Undefined
xxxxn78BH	CAN message data register 603	M_DATA603	R/W		x		Undefined
xxxxn78CH	CAN message data register 604	M_DATA604	R/W		x		Undefined
xxxxn78DH	CAN message data register 605	M_DATA605	R/W		x		Undefined
xxxxn78EH	CAN message data register 606	M_DATA606	R/W		x		Undefined
xxxxn78FH	CAN message data register 607	M_DATA607	R/W		x		Undefined
xxxxn790H	CAN message ID register L60	M_IDL60	R/W			x	Undefined
xxxxn792H	CAN message ID register H60	M_IDH60	R/W			x	Undefined
xxxxn794H	CAN message configuration register 60	M_CONF60	R/W		x		Undefined
xxxxn795H	CAN message status register 60	M_STAT60	R		x		Undefined
xxxxn796H	CAN status set/cancel register 60	SC_STAT60	W			x	0000H
xxxxn7A0H	CAN message event pointer 610	M_EVT610	R/W		x		Undefined
xxxxn7A1H	CAN message event pointer 611	M_EVT611	R/W		x		Undefined
xxxxn7A2H	CAN message event pointer 612	M_EVT612	R/W		x		Undefined
xxxxn7A3H	CAN message event pointer 613	M_EVT613	R/W		x		Undefined
xxxxn7A4H	CAN message data length register 61	M_DLC61	R/W		x		Undefined
xxxxn7A5H	CAN message control register 61	M_CTRL61	R/W		x		Undefined
xxxxn7A6H	CAN message time stamp register 61	M_TIME61	R/W			x	Undefined
xxxxn7A8H	CAN message data register 610	M_DATA610	R/W		x		Undefined
xxxxn7A9H	CAN message data register 611	M_DATA611	R/W		x		Undefined
xxxxn7AAH	CAN message data register 612	M_DATA612	R/W		x		Undefined
xxxxn7ABH	CAN message data register 613	M_DATA613	R/W		x		Undefined
xxxxn7ACH	CAN message data register 614	M_DATA614	R/W		x		Undefined
xxxxn7ADH	CAN message data register 615	M_DATA615	R/W		x		Undefined
xxxxn7AEH	CAN message data register 616	M_DATA616	R/W		x		Undefined
xxxxn7AFH	CAN message data register 617	M_DATA617	R/W		x		Undefined
xxxxn7B0H	CAN message ID register L61	M_IDL61	R/W			x	Undefined
xxxxn7B2H	CAN message ID register H61	M_IDH61	R/W			x	Undefined
xxxxn7B4H	CAN message configuration register 61	M_CONF61	R/W		x		Undefined
xxxxn7B5H	CAN message status register 61	M_STAT61	R		x		Undefined
xxxxn7B6H	CAN status set/cancel register 61	SC_STAT61	W			x	0000H
xxxxn7C0H	CAN message event pointer 620	M_EVT620	R/W		x		Undefined
xxxxn7C1H	CAN message event pointer 621	M_EVT621	R/W		x		Undefined
xxxxn7C2H	CAN message event pointer 622	M_EVT622	R/W		x		Undefined
xxxxn7C3H	CAN message event pointer 623	M_EVT623	R/W		x		Undefined
xxxxn7C4H	CAN message data length register 62	M_DLC62	R/W		x		Undefined

Table 3-5: List of programmable peripheral I/O registers (Sheet 30 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn7C5H	CAN message control register 62	M_CTRL62	R/W		x		Undefined
xxxxn7C6H	CAN message time stamp register 62	M_TIME62	R/W			x	Undefined
xxxxn7C8H	CAN message data register 620	M_DATA620	R/W		x		Undefined
xxxxn7C9H	CAN message data register 621	M_DATA621	R/W		x		Undefined
xxxxn7CAH	CAN message data register 622	M_DATA622	R/W		x		Undefined
xxxxn7CBH	CAN message data register 623	M_DATA623	R/W		x		Undefined
xxxxn7CCH	CAN message data register 624	M_DATA624	R/W		x		Undefined
xxxxn7CDH	CAN message data register 625	M_DATA625	R/W		x		Undefined
xxxxn7CEH	CAN message data register 626	M_DATA626	R/W		x		Undefined
xxxxn7CFH	CAN message data register 627	M_DATA627	R/W		x		Undefined
xxxxn7D0H	CAN message ID register L62	M_IDL62	R/W			x	Undefined
xxxxn7D2H	CAN message ID register H62	M_IDH62	R/W			x	Undefined
xxxxn7D4H	CAN message configuration register 62	M_CONF62	R/W		x		Undefined
xxxxn7D5H	CAN message status register 62	M_STAT62	R		x		Undefined
xxxxn7D6H	CAN status set/cancel register 62	SC_STAT62	W			x	0000H
xxxxn7E0H	CAN message event pointer 630	M_EVT630	R/W		x		Undefined
xxxxn7E1H	CAN message event pointer 631	M_EVT631	R/W		x		Undefined
xxxxn7E2H	CAN message event pointer 632	M_EVT632	R/W		x		Undefined
xxxxn7E3H	CAN message event pointer 633	M_EVT633	R/W		x		Undefined
xxxxn7E4H	CAN message data length register 631	M_DLC63	R/W		x		Undefined
xxxxn7E5H	CAN message control register 63	M_CTRL63	R/W		x		Undefined
xxxxn7E6H	CAN message time stamp register 63	M_TIME63	R/W			x	Undefined
xxxxn7E8H	CAN message data register 630	M_DATA630	R/W		x		Undefined
xxxxn7E9H	CAN message data register 631	M_DATA631	R/W		x		Undefined
xxxxn7EAH	CAN message data register 632	M_DATA632	R/W		x		Undefined
xxxxn7EBH	CAN message data register 633	M_DATA633	R/W		x		Undefined
xxxxn7ECH	CAN message data register 634	M_DATA634	R/W		x		Undefined
xxxxn7EDH	CAN message data register 635	M_DATA635	R/W		x		Undefined
xxxxn7EEH	CAN message data register 636	M_DATA636	R/W		x		Undefined
xxxxn7EFH	CAN message data register 637	M_DATA637	R/W		x		Undefined
xxxxn7FCH	CAN message ID register L63	M_IDL63	R/W			x	Undefined
xxxxn7F2H	CAN message ID register H63	M_IDH63	R/W			x	Undefined
xxxxn7F4H	CAN message configuration register 63	M_CONF63	R/W		x		Undefined
xxxxn7F5H	CAN message status register 63	M_STAT63	R		x		Undefined
xxxxn7F6H	CAN status set/cancel register 63	SC_STAT63	W			x	0000H
xxxxn800H	CAN interrupt pending register	CCINTP	R		x	x	0000H
xxxxn802H	CAN global interrupt pending register	CGINTP	R/W		x	x	00H
xxxxn804H	CAN1 local interrupt pending register	C1INTP	R/W		x	x	00H
xxxxn806H	CAN2 local interrupt pending register	C2INTP	R/W		x	x	00H
xxxxn808H	CAN3 local interrupt pending register	C3INTP	R/W		x	x	00H
xxxxn80CH	CAN stop register	CSTOP	R/W		x	x	0000H
xxxxn810H	CAN global status register **Note**	CGST	R/W	x	x	x	0100H

Table 3-5: List of programmable peripheral I/O registers (Sheet 31 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn812H	CAN global interrupt enable register **Note**	CGIE	R/W	x	x	x	0A00H
xxxxn814H	CAN main clock select register	CGCS	R/W	x	x	x	7F05H
xxxxn816H	CAN timer event enable register	CGTEN	R/W	x	x	x	0000H
xxxxn818H	CAN time stop count register	CGTSC	R	x	x	x	0000H
xxxxn81AH	CAN message find start register	CGMSS	W			x	0000H
xxxxn81AH	CAN message find result register	CGMSR	R			x	0000H
xxxxn81CH	CAN test bus register	CTBR	R/W		x	x	0000H
xxxxn840H	CAN1 address mask register L0	C1MASKL0	R/W		x	x	Undefined
xxxxn842H	CAN1 address mask register H0	C1MASKH0	R/W		x	x	Undefined
xxxxn844H	CAN1 address mask register L1	C1MASKL1	R/W		x	x	Undefined
xxxxn846H	CAN1 address mask register H1	C1MASKH1	R/W		x	x	Undefined
xxxxn848H	CAN1 address mask register L2	C1MASKL2	R/W		x	x	Undefined
xxxxn84AH	CAN1 address mask register H2	C1MASKH2	R/W		x	x	Undefined
xxxxn84CH	CAN1 address mask register L3	C1MASKL3	R/W		x	x	Undefined
xxxxn84EH	CAN1 address mask register H3	C1MASKH3	R/W		x	x	Undefined
xxxxn850H	CAN1 control register	C1CTRL	R/W		x	x	0101H
xxxxn852H	CAN1 definition register **Note**	C1DEF	R/W		x	x	0000H
xxxxn854H	CAN1 information register	C1LAST	R		x	x	00FFH
xxxxn856H	CAN1 error counter register	C1ERC	R		x	x	0000H
xxxxn858H	CAN1 interrupt enable register **Note**	C1IE	R/W		x	x	0000H
xxxxn85AH	CAN1 bus active register	C1BA	R		x	x	00FFH
xxxxn85CH	CAN1 bit rate prescaler register	C1BRP	R/W		x	x	0000H
xxxxn85DH	CAN1 bus diagnostic information register	C1DINF	R		x	x	0000H
xxxxn85EH	CAN1 synchronization control register	C1SYNC	R/W		x	x	0218H
xxxxn880H	CAN2 address mask register L0	C2MASKL0	R/W		x	x	Undefined
xxxxn882H	CAN2 address mask register H0	C2MASKH0	R/W		x	x	Undefined
xxxxn884H	CAN2 address mask register L1	C2MASKL1	R/W		x	x	Undefined
xxxxn886H	CAN2 address mask register H1	C2MASKH1	R/W		x	x	Undefined
xxxxn888H	CAN2 address mask register L2	C2MASKL2	R/W		x	x	Undefined
xxxxn88AH	CAN2 address mask register H2	C2MASKH2	R/W		x	x	Undefined
xxxxn88CH	CAN2 address mask register L3	C2MASKL3	R/W		x	x	Undefined
xxxxn88EH	CAN2 address mask register H3	C2MASKH3	R/W		x	x	Undefined
xxxxn890H	CAN2 control register	C2CTRL	R/W		x	x	0101H
xxxxn892H	CAN2 definition register **Note**	C2DEF	R/W		x	x	0000H
xxxxn894H	CAN2 information register	C2LAST	R		x	x	00FFH
xxxxn896H	CAN2 error counter register	C2ERC	R		x	x	0000H
xxxxn898H	CAN2 interrupt enable register **Note**	C2IE	R/W		x	x	0000H
xxxxn89AH	CAN2 bus active register	C2BA	R		x	x	00FFH
xxxxn89CH	CAN2 bit rate prescaler register	C2BRP	R/W		x	x	0000H
xxxxn89DH	CAN2 bus diagnostic information register	C2DINF	R		x	x	0000H

Table 3-5: List of programmable peripheral I/O registers (Sheet 32 of 32)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1 bit	8 bits	16 bits	
xxxxn89EH	CAN2 synchronization control register	C2SYNC	R/W		x	x	0218H
xxxxn8C0H	CAN1 address mask register L0	C3MASKL0	R/W		x	x	Undefined
xxxxn8C2H	CAN1 address mask register H0	C3MASKH0	R/W		x	x	Undefined
xxxxn8C4H	CAN1 address mask register L1	C3MASKL1	R/W		x	x	Undefined
xxxxn8C6H	CAN3 address mask register H1	C3MASKH1	R/W		x	x	Undefined
xxxxn8C8H	CAN3 address mask register L2	C3MASKL2	R/W		x	x	Undefined
xxxxn8CAH	CAN3 address mask register H2	C3MASKH2	R/W		x	x	Undefined
xxxxn8CCH	CAN3 address mask register L3	C3MASKL3	R/W		x	x	Undefined
xxxxn8CEH	CAN3 address mask register H3	C3MASKH3	R/W		x	x	Undefined
xxxxn8D0H	CAN3 control register	C3CTRL	R/W		x	x	0101H
xxxxn8D2H	CAN3 definition register Note	C3DEF	R/W		x	x	0000H
xxxxn8D4H	CAN3 information register	C3LAST	R		x	x	00FFH
xxxxn8D6H	CAN3 error counter register	C3ERC	R		x	x	0000H
xxxxn8D8H	CAN3 interrupt enable register Note	C3IE	R/W		x	x	0000H
xxxxn8DAH	CAN3 bus active register	C3BA	R		x	x	00FFH
xxxxn8DCH	CAN3 bit rate prescaler register	C3BRP	R/W		x	x	0000H
xxxxn8DDH	CAN3 bus diagnostic information register	C3DINF	R		x	x	0000H
xxxxn8DEH	CAN3 synchronization control register	C3SYNC	R/W		x	x	0218H
xxxxnA10H	ELISA timer event pointer register 0	TEP0	R/W		x		00H
xxxxnA11H	ELISA timer event pointer register 1	TEP1	R/W		x		00H
xxxxnA13H	ELISA timer event pointer register 3	TEP3	R/W		x		00H
xxxxnA14H	ELISA script event pointer register	SEPCC	R/W		x	x	0000H
xxxxnA16H	ELISA event processing status	EEPS	R		x	x	0000H
xxxxnA18H	ELISA status register	ELSR	R/W		x	x	0000H
xxxxnA1AH	ELISA last processed command	ELC	R		x	x	0000H
xxxxnA1BH	ELISA temporary buffer low	ETBL	R/W		x	x	0000H
xxxxnA1CH	ELISA temporary buffer high	ETBH	R/W		x	x	0000H

Note: This register can be accessed in 16-bit or 8-bit units during read and in 16-bit units during write.

Remark: n = xx00b

3.5 Specific Registers

Specific registers are registers that are protected from being written with illegal data due to erroneous program execution, etc. The write access of these specific registers is executed in a specific sequence, and if abnormal store operations occur, it is notified by the peripheral status register (PHS). The V850E/CA1 has three specific registers, clock control register (CKC), the power save control register (PSC) and the power save mode register (PSM). For details of the CKC register, refer to chapter 8.4.1 “Clock Control Register (CKC)” on page 227, for details of the PSC register, refer to chapter 8.6.1 “Power Save Control Register (PSC)” on page 240 and for details of the PSM register refer to chapter 8.6.2 “Power Save Mode Register (PSM)” on page 242.

The access sequence to the specified registers is shown below.

The following sequence shows the data setting of the specific registers.

- Store instruction (ST/SST instruction)
- Bit operation instruction (SET1/CLR1/NOT1 instruction)

Examples **1.**

```
<1>   MOV        0x04,r10
<2>   ST.B       r10,PRCMD[r0]
<3>   ST.B       r10,PSC[r0]
<4>   NOP        dummy instruction (5 times NOP required)
::
```

No special sequence is required when reading the specific registers.

Remarks:

1. A store instruction to a command register will not be received with an interrupt.
This presupposes that this is done with the continuous store instructions in <1> and <2> above in the program. If another instruction is placed between <1> and <2>, when an interrupt is received by that instruction, the above sequence may not be established, and cause a malfunction, so caution is necessary.

2. The data written in the PRCMD register is dummy data, but use the same general purpose register for writing to the PRCMD register (<2> in the example above) as was used in setting data in the specified register (<3> in the example above). Addressing is the same in the case where a general purpose register is used.

3. In a store instruction to the PSC register for setting it in the software STOP mode or IDLE mode, it is necessary to insert 1 or more NOP instructions just after. When clearing each power save mode by interrupt, or when resetting after executing interrupt processing, start executing from the next instruction without executing 1 instruction just after the store instruction.

3.5.1 Command Register (PRCMD)

This command register (PRCMD) is to protect the registers that may have a significant influence on the application system (PSC, PSM) from an inadvertent write access, so that the system does not stop in case of a program hang-up.
This register can only be written in 8-bit units (undefined data is used when this register is read).
Only the first write access to a specific on-chip register (hereafter referred to as a “specific register”) after data has been written to the PRCMD register is valid.
In this way, the value of the specific register can be rewritten only in a specified sequence, and an illegal write access is inhibited.

	7	6	5	4	3	2	1	0	Address	R/W	At Reset
PRCMD	REG7	REG6	REG5	REG4	REG3	REG2	REG1	REG0	FFFFF1FCH	R/W	xxH

REG7-0: registration code (**any** 8-bit data)

Remark: The registers must be written with store instruction execution by CPU. DMA transfer is prohibited.

3.5.2 Peripheral Command Register (PHCMD)

This command register (PHCMD) is to protect the registers that may have a significant influence on the application system (CKC) from an inadvertent write access, so that the system does not stop in case of a program hang-up.
This register can be only written in 8-bit units (undefined data is used when this register is read).

Only the first write access to a specific on-chip register (hereafter referred to as a “specific register”) after data has been written to the PHCMD register is valid.
In this way, the value of the specific register can be rewritten only in a specified sequence, and an illegal write access is inhibited.

	7	6	5	4	3	2	1	0	Address	R/W	At Reset
PHCMD	REG7	REG6	REG5	REG4	REG3	REG2	REG1	REG0	FFFFF800H	R/W	xxH

REG7-0: registration code (**any** 8-bit data)

Remark: The registers must be written with store instruction execution by CPU. DMA transfer is prohibited.

If an illegal store operation takes place, it can be checked by the PRERR flag of the peripheral status register (PHS).

Remark: Write to this registers by DMA transfer is prohibited!

3.5.3 Peripheral Status Register (PHS)

The flag PRERR in the peripheral status register PHS indicates protection error occurrence.
This register can be read/written in 8-bit units or bit-wise.

	7	6	5	4	3	2	1	0	Address	R/W	At Reset
PHS	0	0	0	0	0	0	0	PRERR	FFFFF802H	R/W	00H

Protection error detection:
If an incorrect write operation in a sequence without accessing the command register is performed to a protected internal register, the register is not written to, causing a protection error.
Writing "0" to the PRERR flag after the value is checked clears the error.

Operation conditions of PRERR flag:

Set condition:

<1> If the most recent store instruction for peripheral I/O register operation is not an operation to write the PHCMD register and if data is written to the specific register

<2> If the first store instruction operation after data has been written to the PHCMD register is to memory or peripheral I/Os other than those of a specified register

Reset condition:

<1> When "0" is written to the PRERR flag of the PHS register

<2> On system reset

3.5.4 Internal peripheral function wait control register VSWC

This register inserts wait states to the internal access of peripheral SFRs.
This register can be read or written in 1-bit and 8-bit units.

	7	6	5	4	3	2	1	0	Address	R/W	Reset Value
VSWC	0	SUWL2	SUWL1	SUWL0	0	VSWL2	VSWL1	VSWL0	FFFFF06EH	R/W	77H
	0	1	1	1	0	1	1	1			

<table>
<tr><th>Bit Name</th><th>Description</th></tr>
<tr><td>SUWL2,
SUWL1,
SUWL0</td><td>Setup wait for internal peripheral bus length
<table>
<tr><th>SUWL2</th><th>SUWL1</th><th>SUWL0</th><th>Number of data wait states (n = 7 - 0)</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 system clock</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2 system clock</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3 system clock</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4 system clock</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5 system clock</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6 system clock</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7 system clock (default)</td></tr>
</table></td></tr>
<tr><td>VSWL2,
VSWL1,
VSWL0</td><td>internal peripheral bus wait length
<table>
<tr><th>VSWL2</th><th>VSWL1</th><th>VSWL0</th><th>Number of data wait states (n = 7 - 0)</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 system clock</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2 system clock</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3 system clock</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4 system clock</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5 system clock</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6 system clock</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7 system clock (default)</td></tr>
</table></td></tr>
</table>

Caution: Ensure that at least 1 wait SUWLx and 2 wait VSWLx are set.

Chapter 4 Bus Control Function

The V850E/CA1 / ATOMIC is provided with an external bus interface function by which external memories such as ROM and RAM, and I/O can be connected.

4.1 Features

- 16-bit/8-bit data bus sizing function
- 8 chip areas select function
  - 3 chip area select signals externally available ($\overline{\text{CS2}}$ to $\overline{\text{CS4}}$)
- Wait function
  - Programmable wait function, capable of inserting up to 7 wait states for each memory block
  - External wait function through $\overline{\text{WAIT}}$ pin
- Idle state insertion function
- Bus mastership arbitration function
- Bus hold function
- External device connection can be enabled via bus control/port alternate function pins

4.2 Bus Control Pins

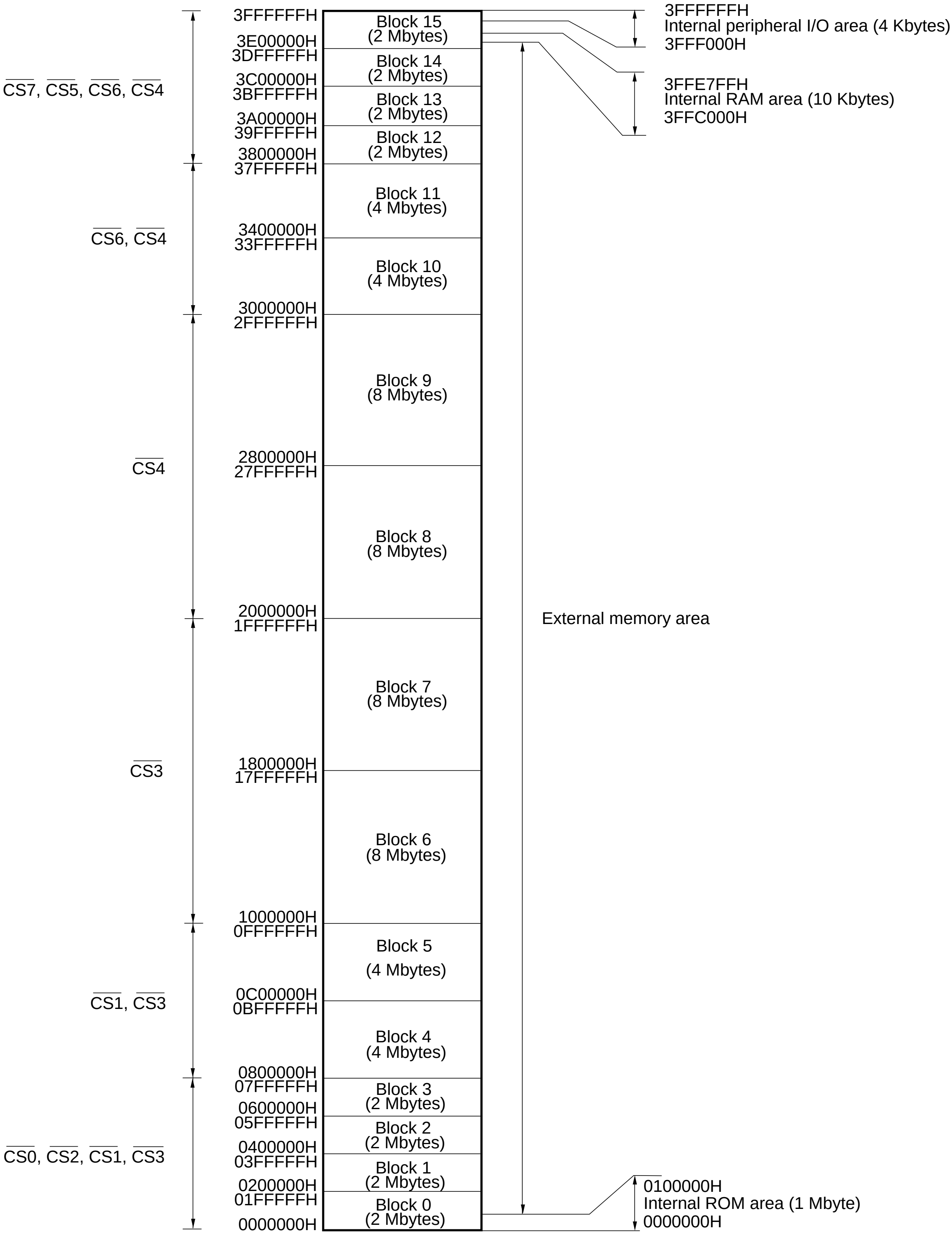
The following pins are used for connecting to external devices.

Bus Control Pin (Function when in Control Mode)	Function when in Port Mode	Register for Port/Control Mode Switching
Address data Data bus (D0 to D15)	PDL0 to PDL15 (Port DL)	PMCDL
Address bus (A0 to A15)	PAL0 to PAL15 (Port AL)	PMCAL
Address bus (A16 to A23)	PAH0 to PAH7 (Port AH)	PMCAH
Chip select ($\overline{\text{CS2}}$ to $\overline{\text{CS4}}$)	PCS2 to PCS4 (Port CS)	PMCCS
Read/write control ($\overline{\text{LWR}}$, $\overline{\text{UWR}}$, $\overline{\text{RD}}$)	PCT0, PCT1, PCT4 (Port CT)	PMCCT
External wait control ($\overline{\text{WAIT}}$)	PCM0 (Port CM)	PMCCM
Internal system clock (CLKOUT)	PCM1 (Port CM)	

4.3 Memory Block Function

The 64 MB memory space is divided into memory blocks of 2 MB, 4 MB, and 8 MB units.

Figure 4-1: Memory Block Function



4.3.1 Chip Select Control Function

The 64 MB memory area can be divided into 2 MB, 4 MB and 8 MB memory blocks by the chip area selection control registers 0 and 1 (CSC0, CSC1) to control the chip select signals.
The memory area can be effectively used by dividing the memory area into memory blocks using the chip select control function. The priority order is described below.

(1) Chip area selection control registers 0, 1 (CSC0, CSC1)

These registers can be read/written in 16-bit units. Valid by setting each bit (to 1).
If different chip area select signals are set to the same block, the priority order is controlled as follows.

CSC0: Peripheral I/O area > $\overline{\text{CS0}}$ > $\overline{\text{CS2}}$ > $\overline{\text{CS1}}$ > $\overline{\text{CS3}}$ **Note**
CSC1: Peripheral I/O area > $\overline{\text{CS7}}$ > $\overline{\text{CS5}}$ > $\overline{\text{CS6}}$ > $\overline{\text{CS4}}$ **Note**

If both the CS0n and CS2n bits of the CSC0 register are set to 0, $\overline{\text{CS1}}$ becomes active to the corresponding block (n = 0 to 3).
Similarly, if both the CS5n and CS7n bits of the CSC1 register are set to 0, $\overline{\text{CS6}}$ becomes active to the corresponding block (n = 0 to 3).

Note: Not all the chip area select signals are externally available on output pins. Even so, enabling chip area select signals other than $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$, the setting for the corresponding memory blocks will be effective too, regardless of an external chip select output pin.

Figure 4-2: Chip Area Select Control Registers 0, 1 (1/2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CSC0	CS33	CS32	CS31	CS30	CS23	CS22	CS21	CS20	CS13	CS12	CS11	CS10	CS03	CS02	CS01	CS00	3FFFFF060H	2C11H
	$\overline{\text{CS3}}$				$\overline{\text{CS2}}$				$\overline{\text{CS1}}$				$\overline{\text{CS0}}$					

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CSC1	CS43	CS42	CS41	CS40	CS53	CS52	CS51	CS50	CS63	CS62	CS61	CS60	CS73	CS72	CS71	CS70	3FFFFF062H	2C11H
	$\overline{\text{CS4}}$				$\overline{\text{CS5}}$				$\overline{\text{CS6}}$				$\overline{\text{CS7}}$					

Figure 4-2: Chip Area Select Control Registers 0, 1 (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15 to 0</td><td>CSn0 to
CSn3
(n = 0 to 7)</td><td>Chip Select
Enables chip select.
<table>
<tr><th>CSnm</th><th>CS Operation</th></tr>
<tr><td>CS00</td><td>$\overline{\text{CS0}}$ active during block 0 access</td></tr>
<tr><td>CS01</td><td>$\overline{\text{CS0}}$ active during block 1 access.</td></tr>
<tr><td>CS02</td><td>$\overline{\text{CS0}}$ active during block 2 access.</td></tr>
<tr><td>CS03</td><td>$\overline{\text{CS0}}$ active during block 3 access.</td></tr>
<tr><td>CS10</td><td>$\overline{\text{CS1}}$ active during block 0 or 1 access.</td></tr>
<tr><td>CS11</td><td>$\overline{\text{CS1}}$ active during block 2 or 3 access.</td></tr>
<tr><td>CS12</td><td>$\overline{\text{CS1}}$ active during block 4 access.</td></tr>
<tr><td>CS13</td><td>$\overline{\text{CS1}}$ active during block 5 access.</td></tr>
<tr><td>CS20</td><td>$\overline{\text{CS2}}$ active during block 0 access.</td></tr>
<tr><td>CS21</td><td>$\overline{\text{CS2}}$ active during block 1 access.</td></tr>
<tr><td>CS22</td><td>$\overline{\text{CS2}}$ active during block 2 access.</td></tr>
<tr><td>CS23</td><td>$\overline{\text{CS2}}$ active during block 3 access.</td></tr>
<tr><td>CS30</td><td>$\overline{\text{CS3}}$ active during block 0, 1, 2, or 3 access.</td></tr>
<tr><td>CS31</td><td>$\overline{\text{CS3}}$ active during block 4 or 5 access.</td></tr>
<tr><td>CS32</td><td>$\overline{\text{CS3}}$ active during block 6 access.</td></tr>
<tr><td>CS33</td><td>$\overline{\text{CS3}}$ active during block 7 access.</td></tr>
<tr><td>CS40</td><td>$\overline{\text{CS4}}$ active during block 12, 13, 14, or 15 access.</td></tr>
<tr><td>CS41</td><td>$\overline{\text{CS4}}$ active during block 10 or 11 access.</td></tr>
<tr><td>CS42</td><td>$\overline{\text{CS4}}$ active during block 9 access.</td></tr>
<tr><td>CS43</td><td>$\overline{\text{CS4}}$ active during block 8 access.</td></tr>
<tr><td>CS50</td><td>$\overline{\text{CS5}}$ active during block 15 access.</td></tr>
<tr><td>CS51</td><td>$\overline{\text{CS5}}$ active during block 14 access.</td></tr>
<tr><td>CS52</td><td>$\overline{\text{CS5}}$ active during block 13 access.</td></tr>
<tr><td>CS53</td><td>$\overline{\text{CS5}}$ active during block 12 access.</td></tr>
<tr><td>CS60</td><td>$\overline{\text{CS6}}$ active during block 14 or 15 access.</td></tr>
<tr><td>CS61</td><td>$\overline{\text{CS6}}$ active during block 12 or 13 access.</td></tr>
<tr><td>CS62</td><td>$\overline{\text{CS6}}$ active during block 11 access.</td></tr>
<tr><td>CS63</td><td>$\overline{\text{CS6}}$ active during block 10 access.</td></tr>
<tr><td>CS70</td><td>$\overline{\text{CS7}}$ active during block 15 access.</td></tr>
<tr><td>CS71</td><td>$\overline{\text{CS7}}$ active during block 14 access.</td></tr>
<tr><td>CS72</td><td>$\overline{\text{CS7}}$ active during block 13 access.</td></tr>
<tr><td>CS73</td><td>$\overline{\text{CS7}}$ active during block 12 access.</td></tr>
</table>
</td></tr>
</table>

4.4 Bus Cycle Type Control Function

In the V850E/CA1 / ATOMIC, the following external devices can be connected directly to each memory block.

- SRAM, external ROM, external I/O
- Page ROM

Connected external devices are specified by the bus cycle type configuration registers 0, 1 (BCT0, BCT1).

4.4.1 Bus cycle type configuration

(1) Bus cycle configuration registers 0, 1 (BCT0, BCT1)

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
BCT0	ME3	0	0	BT300	ME2	0	0	BT200	ME1	0	0	BT100	ME0	0	0	BT00	FFFFF480H	8888H
	$\overline{CS3}$				$\overline{CS2}$				$\overline{CS1}$				$\overline{CS0}$					

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
BCT1	ME7	0	0	BT700	ME6	0	0	BT600	ME5	0	0	BT500	ME4	0	0	BT400	FFFFF482H	8888H
	$\overline{CS7}$				$\overline{CS6}$				$\overline{CS5}$				$\overline{CS4}$					

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15, 11, 7, 3
(BCT0),
15, 11, 7, 3
(BCT1)</td><td>MEn
(n = 0 to 7)</td><td>Memory Controller Enable
Sets memory controller operation enable for each chip select signal $\overline{CSn}$.
<table>
<tr><th>MEn</th><th>Memory Controller Operation Enable</th></tr>
<tr><td>0</td><td>Operation disable</td></tr>
<tr><td>1</td><td>Operation enable</td></tr>
</table></td></tr>
<tr><td>12, 8, 4, 0
(BCT0),
(BCT1)</td><td>BTn0
(n = 0 to 7)</td><td>Bus Cycle Type
Specifies the device to be connected to the $\overline{CSn}$ signal.
<table>
<tr><th>BTn0</th><th>External Device Connected Directly to $\overline{CSn}$ signal</th></tr>
<tr><td>0</td><td>SRAM, external I/O</td></tr>
<tr><td>1</td><td>Page ROM</td></tr>
</table></td></tr>
</table>

Cautions: 1. Write to the BCT0 and BCT1 registers after reset, and then do not change the set value. Also, do not access an external memory area other than that for this initialization routine until initial setting of the BCT0 and BCT1 registers is finished. However, it is possible to access external memory areas whose initialization has been finished.

2. The Bits marked as 0 are reserved. It have to leave to 0.

4.5 Bus Access

4.5.1 Number of access clocks

The number of basic clocks necessary for accessing each resource is as follows.

Table 4-1: Number of Bus Access Clocks

Resources (Bus width) Bus Cycle Configuration		Internal ROM (32 bits)	Internal RAM (32 bits)	Peripheral I/O (16 bits)	External memory (16 bits)
Instruction fetch	Normal access	$1^{\text{Note 1}}$	$1^{\text{Note 1}}$	-	$2^{\text{Note 2}}$
	Branch	2	1	-	$2^{\text{Note 2}}$
Operand data access		5	1	$3^{\text{Note 2}}$	$2^{\text{Note 2}}$

Notes: **1.** The instruction fetch becomes 2 clocks, in case of contention with data access.
2. This is the minimum value.

4.5.2 Bus sizing function

The bus sizing function controls data bus width for each CS area. The data bus width is specified by using the bus size configuration register (BSC).

(1) Bus size configuration register (BSC)

This register 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
BSC	0	BS70	0	BS60	0	BS50	0	BS40	0	BS30	0	BS20	0	BS10	0	BS00	FFFFF066H	5555H
	$\overline{\text{CS7}}$		$\overline{\text{CS6}}$		$\overline{\text{CS5}}$		$\overline{\text{CS4}}$		$\overline{\text{CS3}}$		$\overline{\text{CS2}}$		$\overline{\text{CS1}}$		$\overline{\text{CS0}}$			

Bit Position	Bit Name	Function
15 to 0	BSn1, BSn0 (n = 0 to 7)	Data Bus Width Sets the data bus width of CSn area.

BSn0	Data Bus Width of CSn area
0	8 bits
1	16 bits

Cautions: 1. Write to the BSC register after reset, and then do not change the set value. Also, do not access an external memory area other than that for this initialization routine until initial setting of the BSC register is finished. However, it is possible to access external memory areas whose initialization has been finished.

2. When the data bus width is specified as 8 bits, only the $\overline{\text{LWR}}$ signal becomes active.

4.5.3 Endian control function

The endian control function can be used to set processing of word data in memory either by the Big Endian method or the Little Endian method for each CS area selected with the chip select signal ($\overline{\text{CS0}}$ to $\overline{\text{CS7}}$). Switching of the endian method is specified with the endian configuration register (BEC).

(1) Endian configuration register (BEC)

This register 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
BEC	0	BE70	0	BE60	0	BE50	0	BE40	0	BE30	0	BE20	0	BE10	0	BE00	FFFFF068H	0000H
	$\overline{\text{CS7}}$		$\overline{\text{CS6}}$		$\overline{\text{CS5}}$		$\overline{\text{CS4}}$		$\overline{\text{CS3}}$		$\overline{\text{CS2}}$		$\overline{\text{CS1}}$		$\overline{\text{CS0}}$			

Bit Position	Bit Name	Function
14, 12, 10, 8, 6, 4, 2, 0	BEn0 (n = 0 to 7)	Big Endian Specifies the endian method. BEn0 | Endian Control 0 | Little Endian method 1 | Big Endian method

Cautions: 1. Bits 15, 13, 11, 9, 7, 5, 3, and 1 of the BEC register must be cleared (0). If these bits are set to 1, the operation is not guaranteed.

2. Set the CSn area specified as the programmable peripheral I/O area to Little Endian format (n = 0 to 7).

3. In the following areas, the data processing method is fixed to Little Endian method. Any setting of Big Endian method for these areas according to the BEC register is invalid.

- On-chip peripheral I/O area
- Internal ROM area
- Internal RAM area
- Fetch area of external memory

Figure 4-3: Big Endian Addresses within Word

31 24	23 16	17 8	7 0
0008H	0009H	000AH	000BH
0004H	0005H	0006H	0007H
0000H	0001H	0002H	0003H

Figure 4-4: Little Endian Addresses within Word

31 24	23 16	17 8	7 0
000BH	000AH	0009H	0008H
0007H	0006H	0005H	0004H
0003H	0002H	0001H	0000H

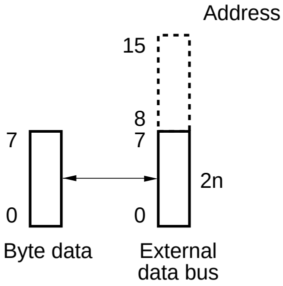
4.5.4 Bus width

The V850E/CA1 / ATOMIC accesses peripheral I/O and external memory in 8-bit, 16-bit, or 32-bit units. The following shows the operation for each type of access. Access all data in order starting from the lower order side.

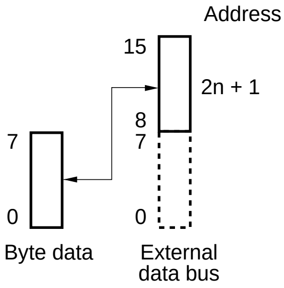
(1) Byte access (8 bits)

(a) When the data bus width is 16 bits (Little Endian)

<1> Access to even address (2n)

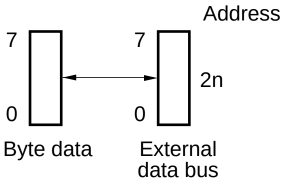


<2> Access to odd address (2n + 1)

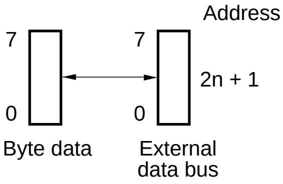


(b) When the data bus width is 8 bits (Little Endian)

<1> Access to even address (2n)

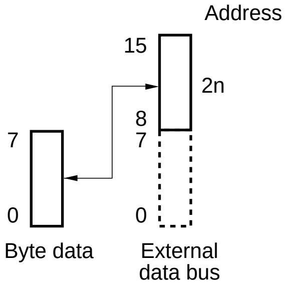


<2> Access to odd address (2n + 1)

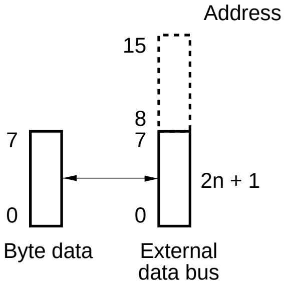


(c) When the data bus width is 16 bits (Big Endian)

<1> Access to even address (2n)

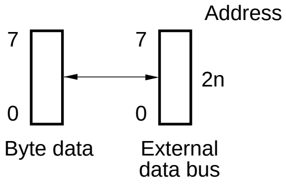


<2> Access to odd address (2n + 1)

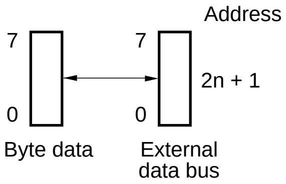


(d) When the data bus width is 8 bits (Big Endian)

<1> Access to even address (2n)



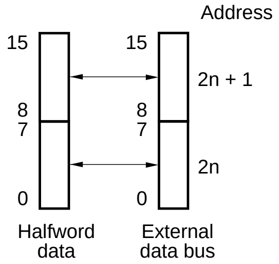
<2> Access to odd address (2n + 1)



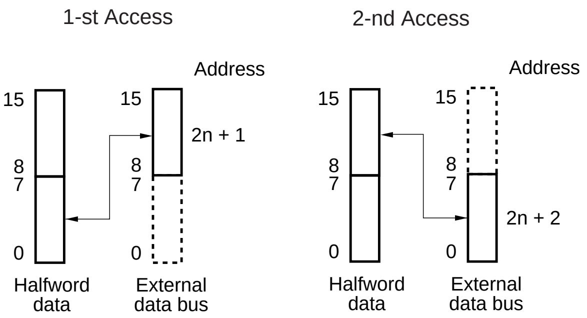
(2) Halfword access (16 bits)

(a) When the bus width is 16 bits (Little Endian)

<1> Access to even address (2n)

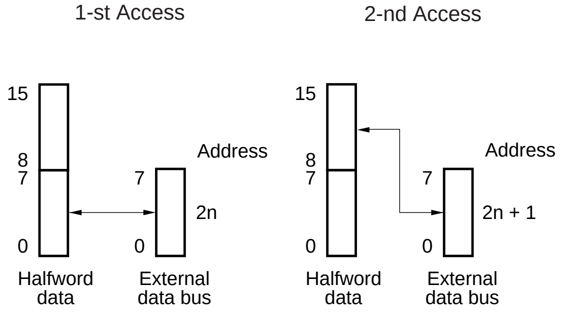


<2> Access to odd address (2n + 1)

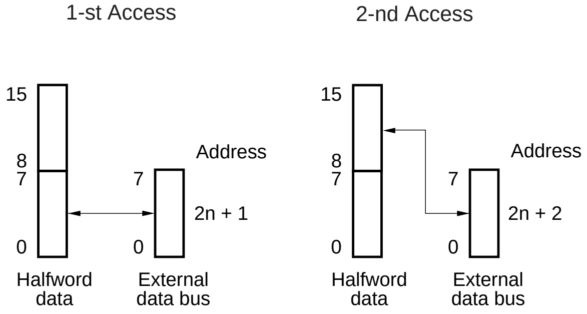


(b) When the data bus width is 8 bits (Little Endian)

<1> Access to even address (2n)

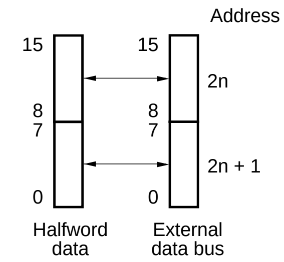


<2> Access to odd address (2n + 1)

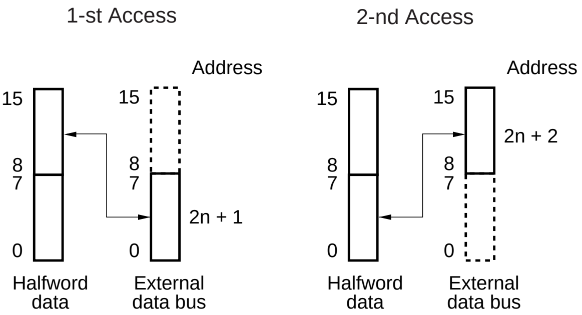


(c) When the data bus width is 16 bits (Big Endian)

<1> Access to even address (2n)

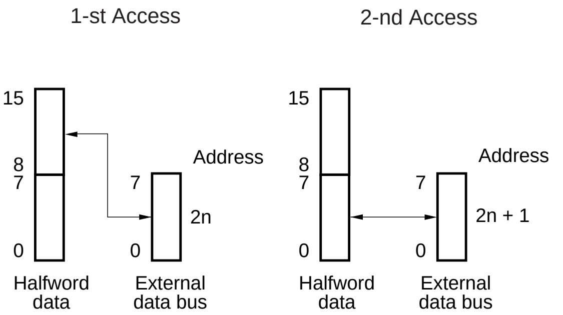


<2> Access to odd address (2n + 1)

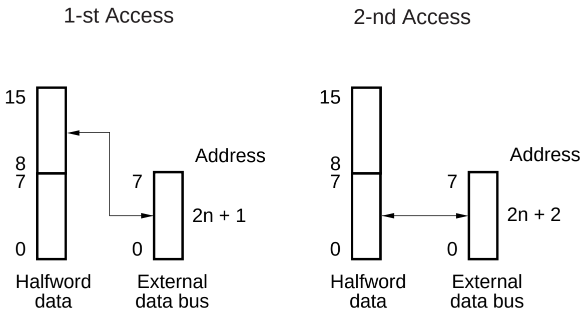


(d) When the data bus width is 8 bits (Big Endian)

<1> Access to even address (2n)



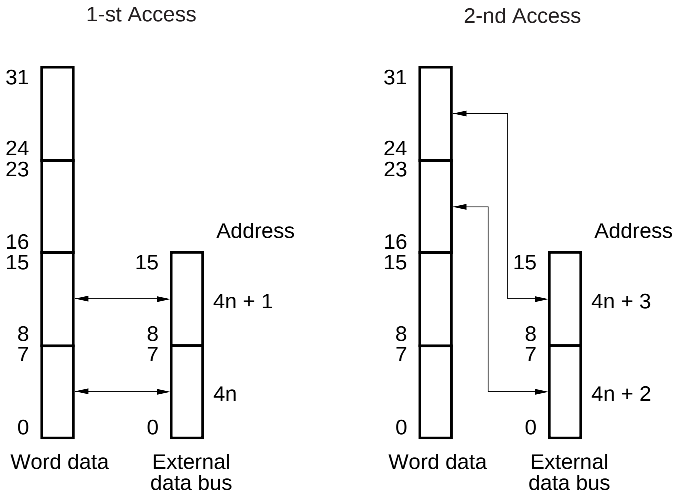
<2> Access to odd address (2n + 1)



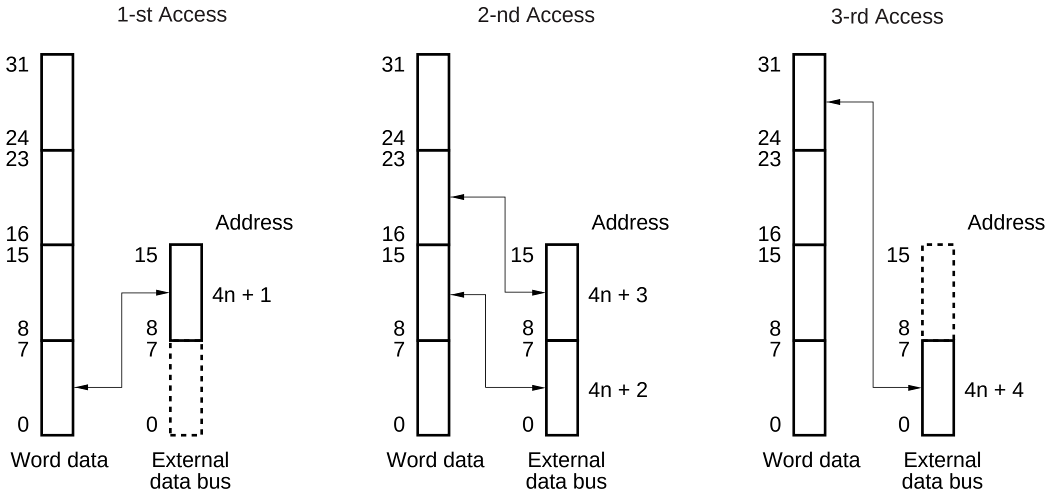
(3) Word access (32 bits)

(a) When the bus width is 16 bits (Little Endian)

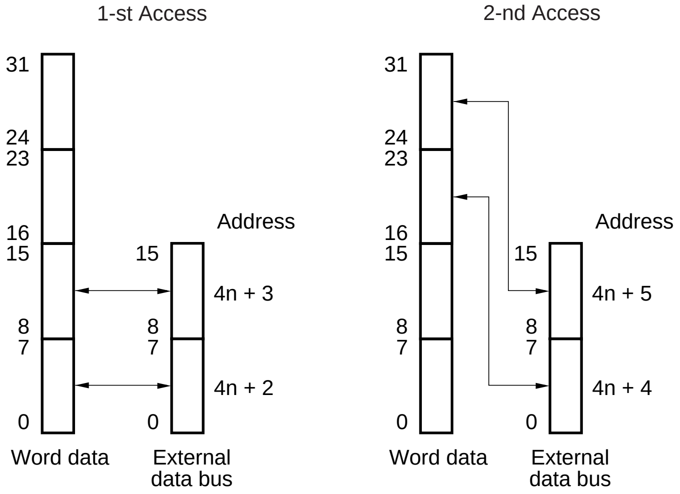
<1> Access to address 4n



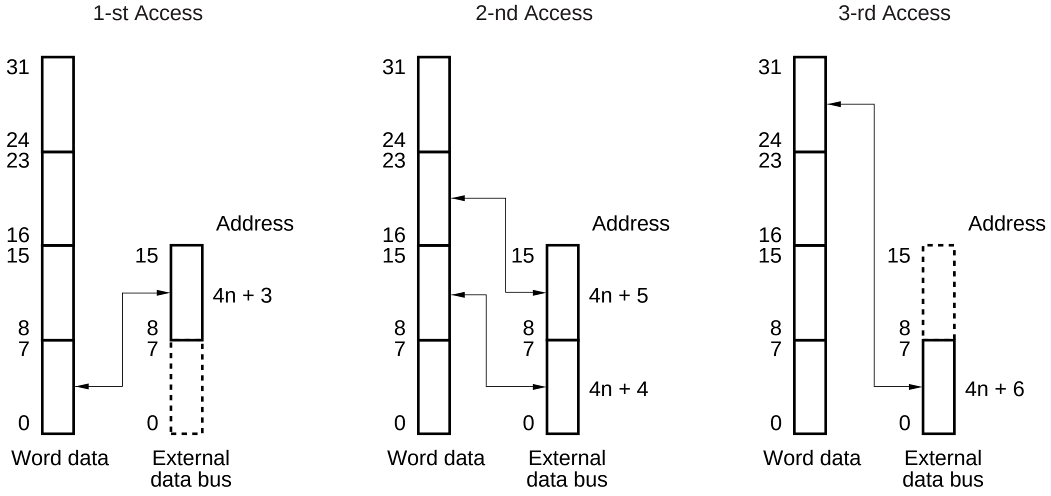
<2> Access to address 4n + 1



<3> Access to address 4n + 2

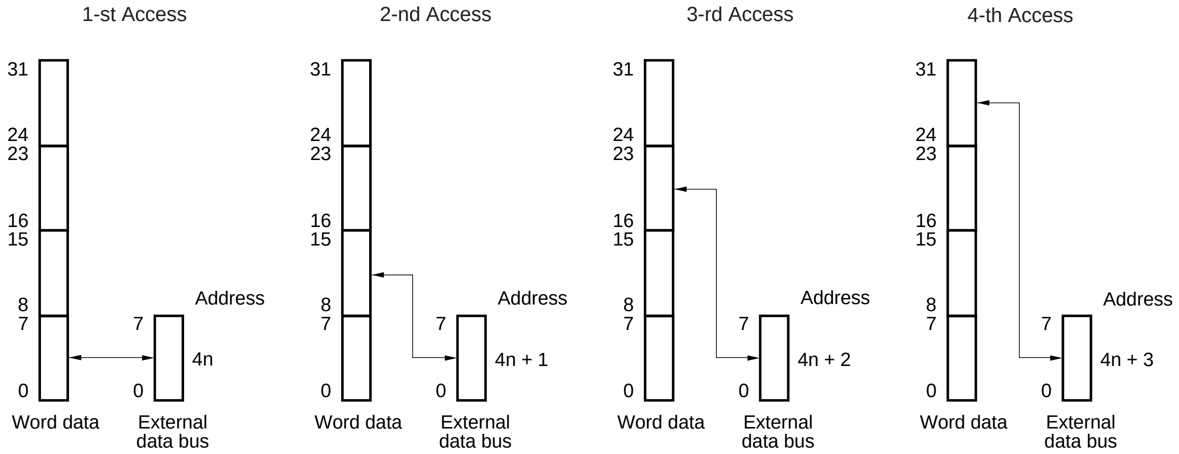


<4> Access to address 4n + 3

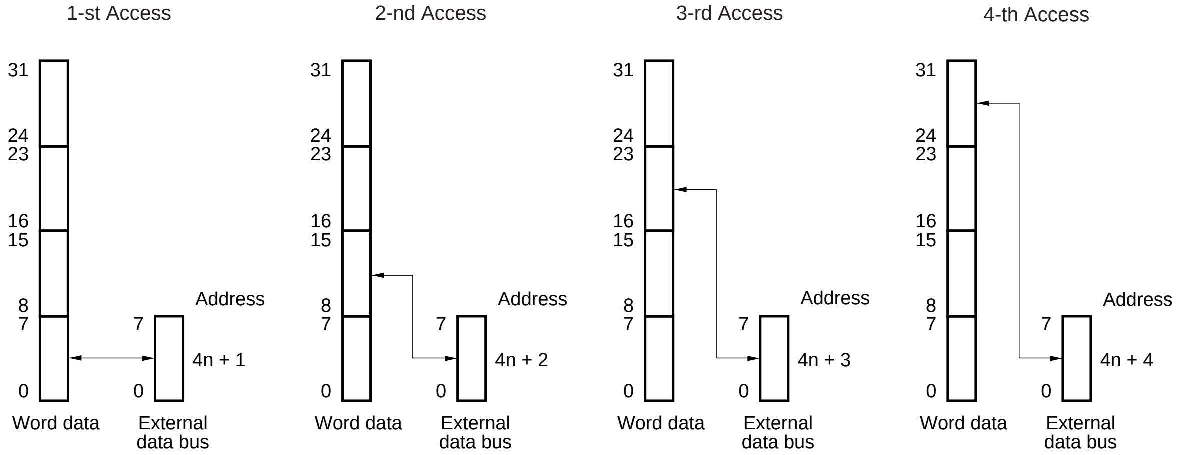


(b) When the bus width is 8 bits (Little Endian)

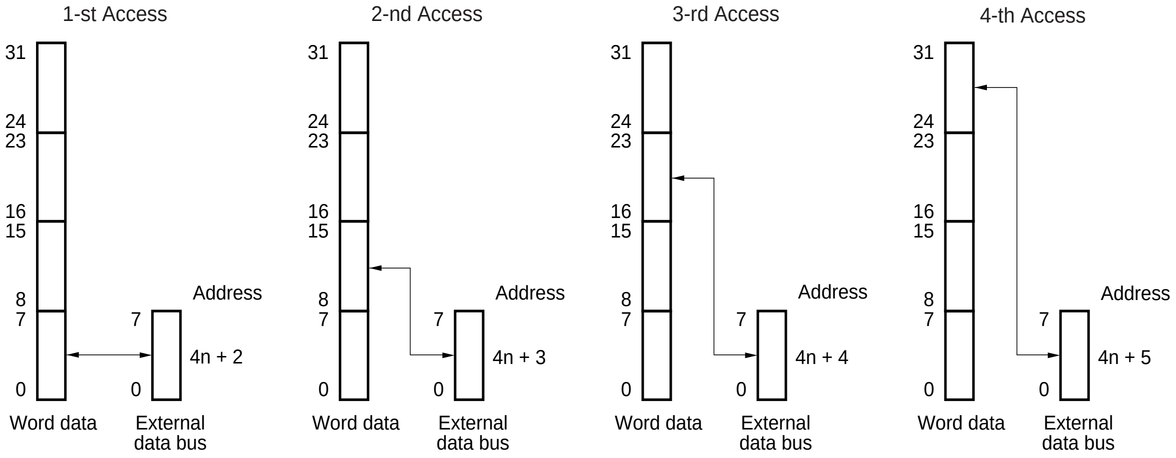
<1> Access to address 4n



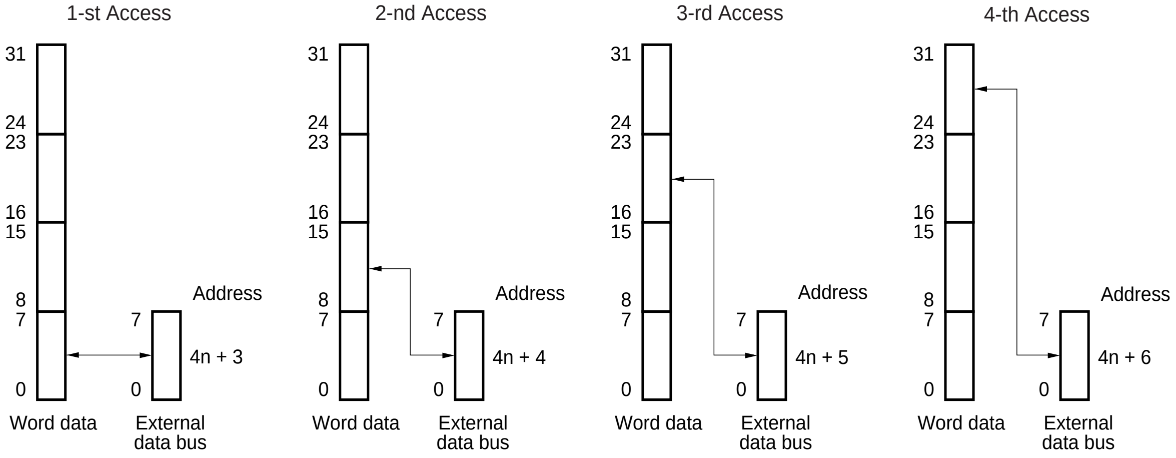
<2> Access to address 4n + 1



<3> Access to address 4n + 2

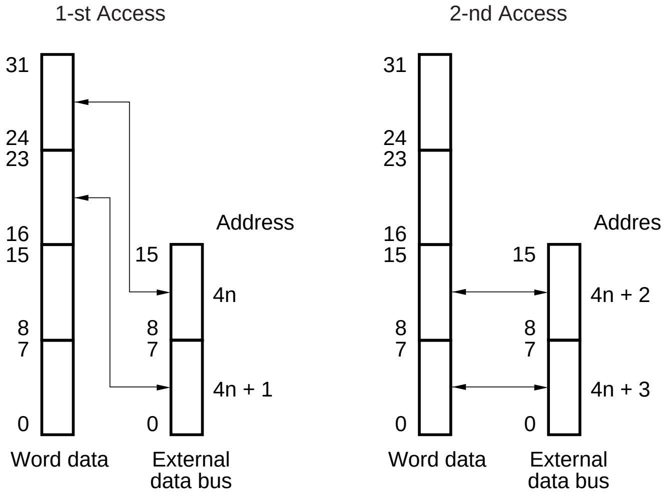


<4> Access to address 4n + 3

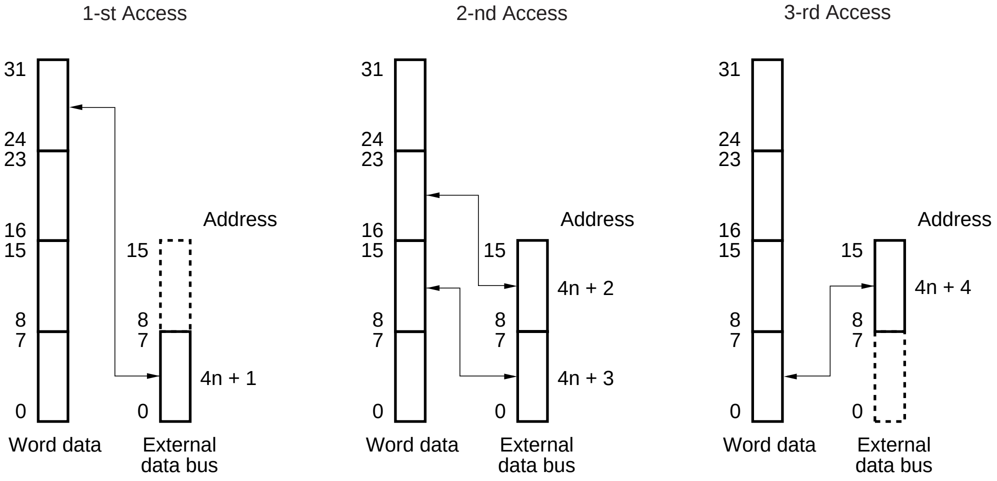


(c) When the data bus width is 16 bits (Big Endian)

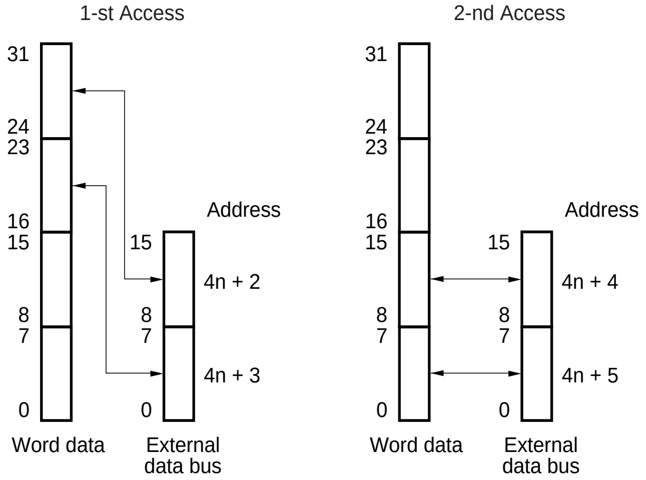
<1> Access to address 4n



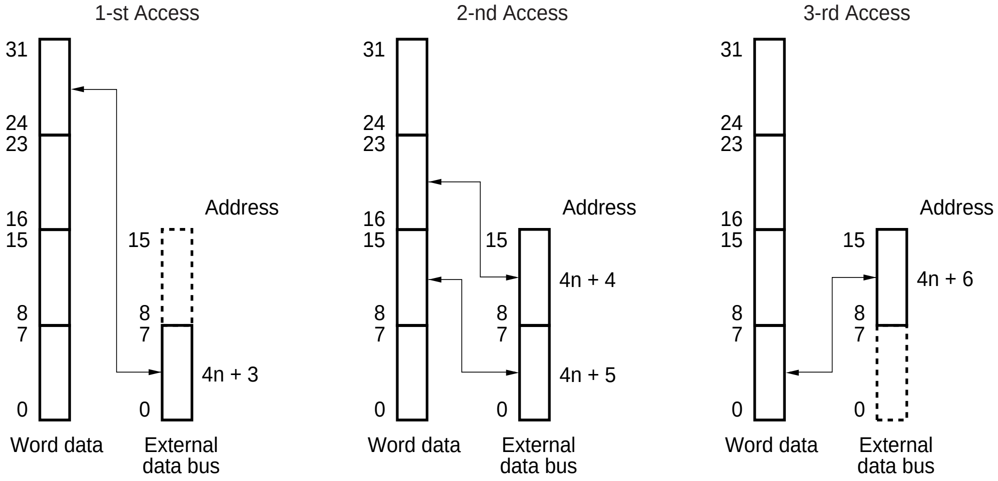
<2> Access to address 4n + 1



<3> Access to address 4n + 2

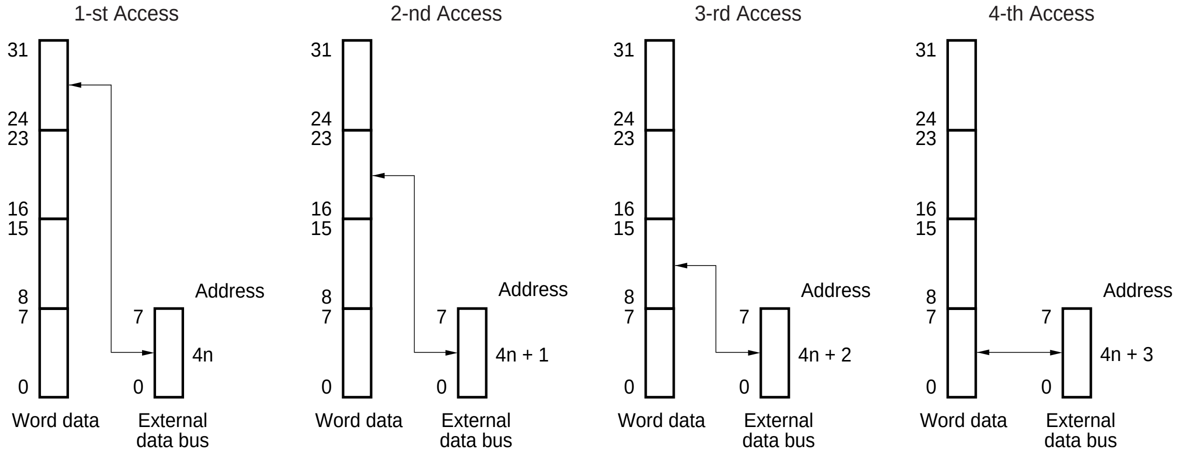


<4> Access to address 4n + 3

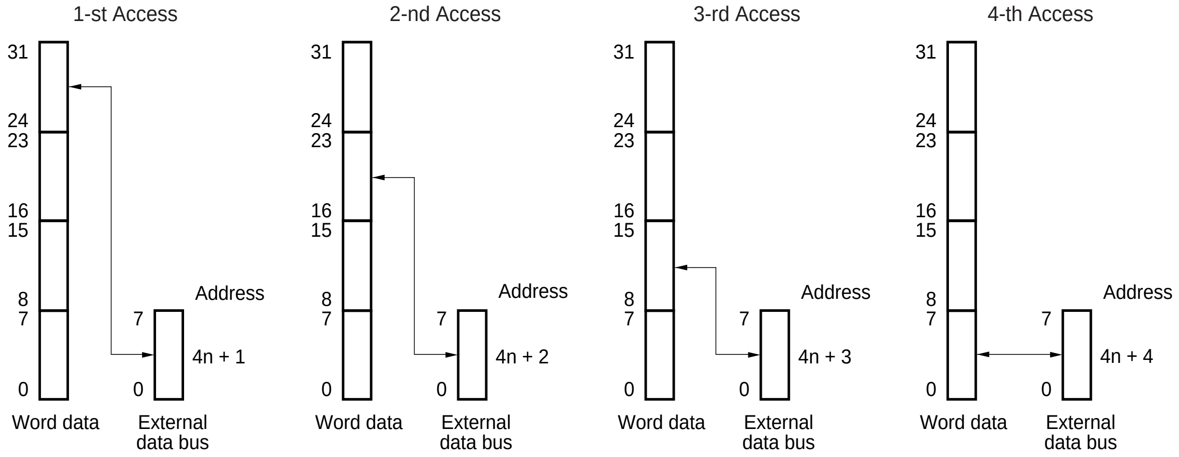


(d) When the data bus width is 8 bits (Big Endian)

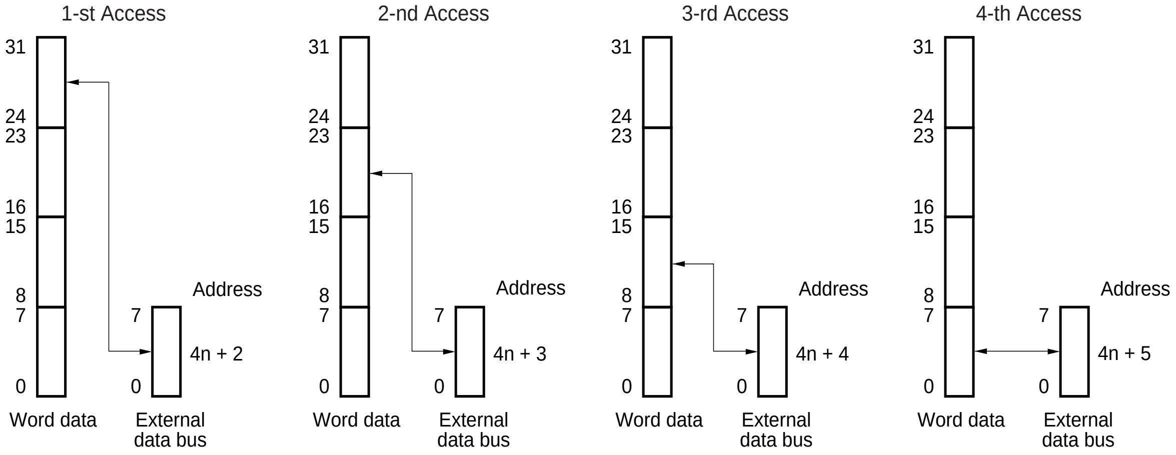
<1> Access to address 4n



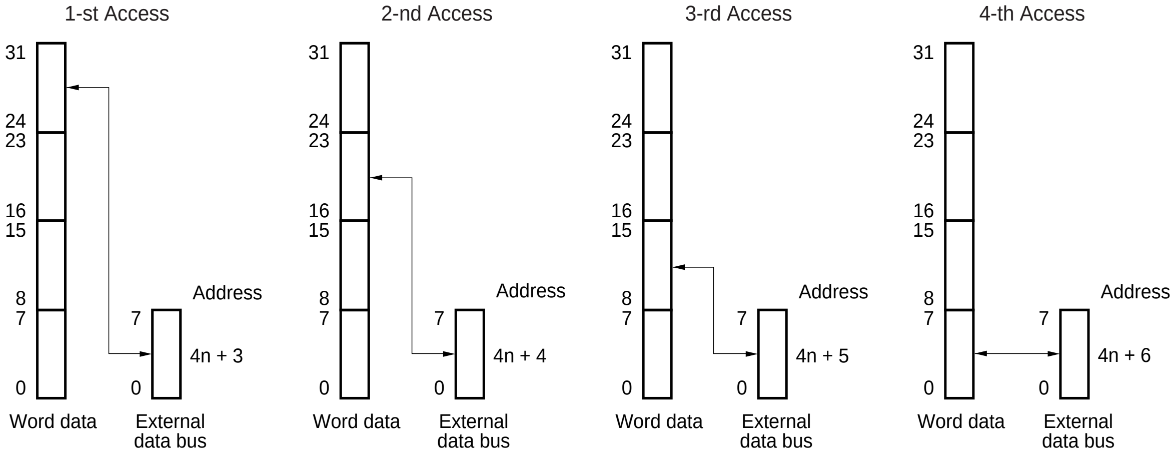
<2> Access to address 4n + 1



<3> Access to address 4n + 2



<4> Access to address 4n + 3



4.6 Wait Function

4.6.1 Programmable wait function

(1) Data wait control registers 0, 1 (DWC0, DWC1)

With the purpose of realizing easy interfacing with low-speed memory or with I/Os, it is possible to insert up to 7 data wait states with respect to the starting bus cycle for each CS area.
The number of wait states can be specified by data wait control registers 0 and 1 (DWC0, DWC1) in programming. Just after system reset, all blocks have 7 data wait states inserted.

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
DWC0	0	DW32	DW31	DW30	0	DW22	DW21	DW20	0	DW12	DW11	DW10	0	DW02	DW01	DW00	FFFFF484H	7777H
		$\overline{CS3}$				$\overline{CS2}$				$\overline{CS1}$				$\overline{CS0}$				

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DWC1	0	DW72	DW71	DW70	0	DW62	DW61	DW60	0	DW52	DW51	DW50	0	DW42	DW41	DW40	FFFFF486H	7777H
		$\overline{CS7}$				$\overline{CS6}$				$\overline{CS5}$				$\overline{CS4}$				

Bit Position	Bit Name	Function			
14 to 12, 10 to 8, 6 to 4, 2 to 0	DWn2 to DWn0 (n = 0 to 7)	Data Wait Specifies the number of wait states inserted in the $\overline{CSn}$ area.			
		DWn2	DWn1	DWn0	Number of Wait States Inserted in $\overline{CSn}$ Space
		0	0	0	No wait states inserted
		0	0	1	1
		0	1	0	2
		0	1	1	3
		1	0	0	4
		1	0	1	5
		1	1	0	6
		1	1	1	7

Cautions: 1. The internal ROM area and internal RAM area are not subject to programmable waits and ordinarily no wait access is carried out. The internal peripheral I/O area is also not subject to programmable wait states, with wait control performed only by each peripheral function.

2. In the following cases, the settings of registers DWC0 and DWC1 are invalid (wait control is performed by each memory controller).
- Page ROM on-page access

3. Write to the DWC0 and DWC1 registers after reset, and then do not change the set values. Also, do not access an external memory area other than that for this initialization routine until initial setting of the DWC0 and DWC1 registers is finished. However, it is possible to access external memory areas whose initialization has been finished.

(2) Address setup wait control register (ASC)

The V850E/CA1 / ATOMIC allows insertion of address setup wait states before the T1 cycle of the SRAM or page ROM cycle.
The number of address setup wait states can be set with the ASC register for each CS area.

This register 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
ASC	AC71	AC70	AC61	AC60	AC51	AC50	AC41	AC40	AC31	AC30	AC21	AC20	AC11	AC10	AC01	AC00	FFFFF48AH	FFFFH
	$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$			

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15 to 0</td><td>ACn1,
ACn0
(n = 0 to 7)</td><td>Address Cycle
Specifies the number of address setup wait states inserted before the T1 cycle of SRAM/page ROM cycle for each CS area.
<table>
<tr><th>ACn1</th><th>ACn0</th><th>Number of Wait States</th></tr>
<tr><td>0</td><td>0</td><td>Not inserted</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>2</td></tr>
<tr><td>1</td><td>1</td><td>3</td></tr>
</table>
</td></tr>
</table>

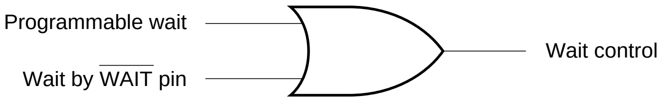
Remark: During address setup wait, the external wait function is disabled by the $\overline{\text{WAIT}}$ pin.

4.6.2 External wait function

When an extremely slow device, I/O, or asynchronous system is connected, any number of wait states can be inserted in a bus cycle by the external wait pin ($\overline{\text{WAIT}}$) to synchronize with the external device. Just as with programmable waits, access to internal ROM, internal RAM, and internal peripheral I/O areas cannot be controlled by external waits.
Input of the external $\overline{\text{WAIT}}$ signal can be done asynchronously to CLKOUT and is sampled at the rising edge of the clock in the T1 and TW states of a bus cycle. If the setup/hold time at sampling timing is not satisfied, the wait state may or may not be inserted in the next state.

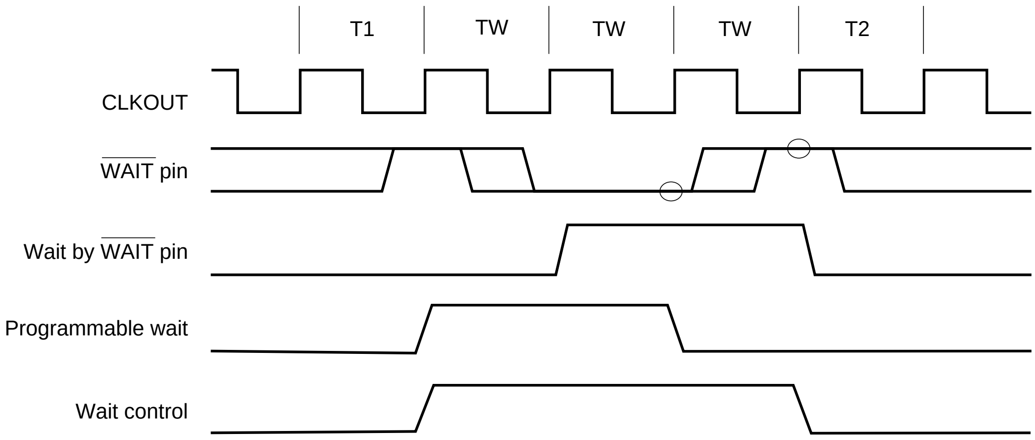
4.6.3 Relationship between programmable wait and external wait

A wait cycle is inserted as the result of an OR operation between the wait cycle specified by the set value of the programmable wait and the wait cycle controlled by the $\overline{\text{WAIT}}$ pin. In other words, the number of wait cycles is determined by the side with the greatest number of cycles.



For example, if the programmable wait and the timing of the $\overline{\text{WAIT}}$ pin signal are as illustrated below, three wait states will be inserted in the bus cycle.

Figure 4-5: Example of Wait Insertion


T1
TW
TW
TW
T2
CLKOUT
$\overline{\text{WAIT}}$ pin
Wait by $\overline{\text{WAIT}}$ pin
Programmable wait
Wait control

Remark: The circle ❍ indicates the sampling timing.

4.7 Idle State Insertion Function

To facilitate interfacing with low-speed memory devices, an idle state (TI) can be inserted into the current bus cycle after the T2 state to meet the data output float delay time (tdf) on memory read access for each CS space. The bus cycle following the T2 state starts after the idle state is inserted.

An idle state is inserted after read/write cycles for SRAM, external I/O, or external ROM.

In the following cases, an idle state is inserted in the timing.

- after read/write cycles for SRAM, external I/O, or external ROM

The idle state insertion setting can be specified by program using the bus cycle control register (BCC). Immediately after the system reset, idle state insertion is automatically programmed for all memory blocks.

(1) Bus cycle control register (BCC)

This register 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
BCC	BC71	BC70	BC61	BC60	BC51	BC50	BC41	BC40	BC31	BC30	BC21	BC20	BC11	BC10	BC01	BC00	FFFFF488H	FFFFH
	$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$			

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15 to 0</td><td>BCn1,
BCn0
(n = 0 to 7)</td><td>Data Cycle
Specifies the insertion of an idle state when accessing corresponding $\overline{CSn}$ area.
<table>
<tr><th>BCn1</th><th>BCn0</th><th>Idle State in $\overline{CSn}$ Area</th></tr>
<tr><td>0</td><td>0</td><td>Not inserted</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>2</td></tr>
<tr><td>1</td><td>1</td><td>3</td></tr>
</table></td></tr>
</table>

Cautions: 1. The internal ROM area, internal RAM area, and internal peripheral I/O area are not subject to insertion of an idle state.

2. Write to the BCC register after reset, and then do not change the set value. Also, do not access an external memory area other than that for this initialization routine until initial setting of the BCC register is finished. However, it is possible to access external memory areas whose initialization has been finished.

4.8 Bus Priority Order

There are three external bus cycles: DMA cycle, operand data access, and instruction fetch.
As for the priority order, the highest priority has the DMA cycle, instruction fetch, and operand data access, in this order.
An instruction fetch may be inserted between read access and write access during read modify write access.
Also, an instruction fetch may be inserted between bus access and bus access during CPU bus clock.

Table 4-2: Bus Priority Order

Priority Order	External Bus Cycle	Bus Master
High	DMA cycle	DMA controller
↕	Operand data access	CPU
Low	Instruction fetch	CPU

4.9 Boundary Operation Conditions

4.9.1 Program space

(1) Branching to the peripheral I/O area or successive fetch from the internal RAM area to the internal peripheral I/O area is inhibited. In terms of hardware, fetching the NOP op code continues, and fetching from the external memory is not performed.

(2) If a branch instruction exists at the upper limit of the internal RAM area, a pre-fetch operation (invalid fetch) that straddles over the internal peripheral I/O area does not occur when instruction fetch is performed.

4.9.2 Data space

The V850E/CA1 / ATOMIC is provided with an address misalign function.
Through this function, regardless of the data format (word data, halfword data, or byte data), data can be placed in all addresses. However, in the case of word data and halfword data, if data are not subjected to boundary alignment, the bus cycle will be generated a minimum of 2 times and bus efficiency will drop.

(1) In the case of halfword length data access

When the address's LSB bit is 1, the byte length bus cycle will be generated 2 times.

(2) In the case of word length data access

(a) When the address's LSB is 1, bus cycles will be generated in the order of byte length bus cycle, halfword length bus cycle, and word length bus cycle.

(b) When the address's lowest 2 bits are 10, the halfword length bus cycle will be generated 2 times.

[MEMO]

Chapter 5 Memory Access Control Function

5.1 SRAM, External ROM, External I/O Interface

5.1.1 Features

- Access to SRAM takes a minimum of 2 states.

- Up to 7 states of programmable data waits can be inserted through setting of the DWC0 and DWC1 registers.

- Data wait can be controlled with input pin ($\overline{\text{WAIT}}$).

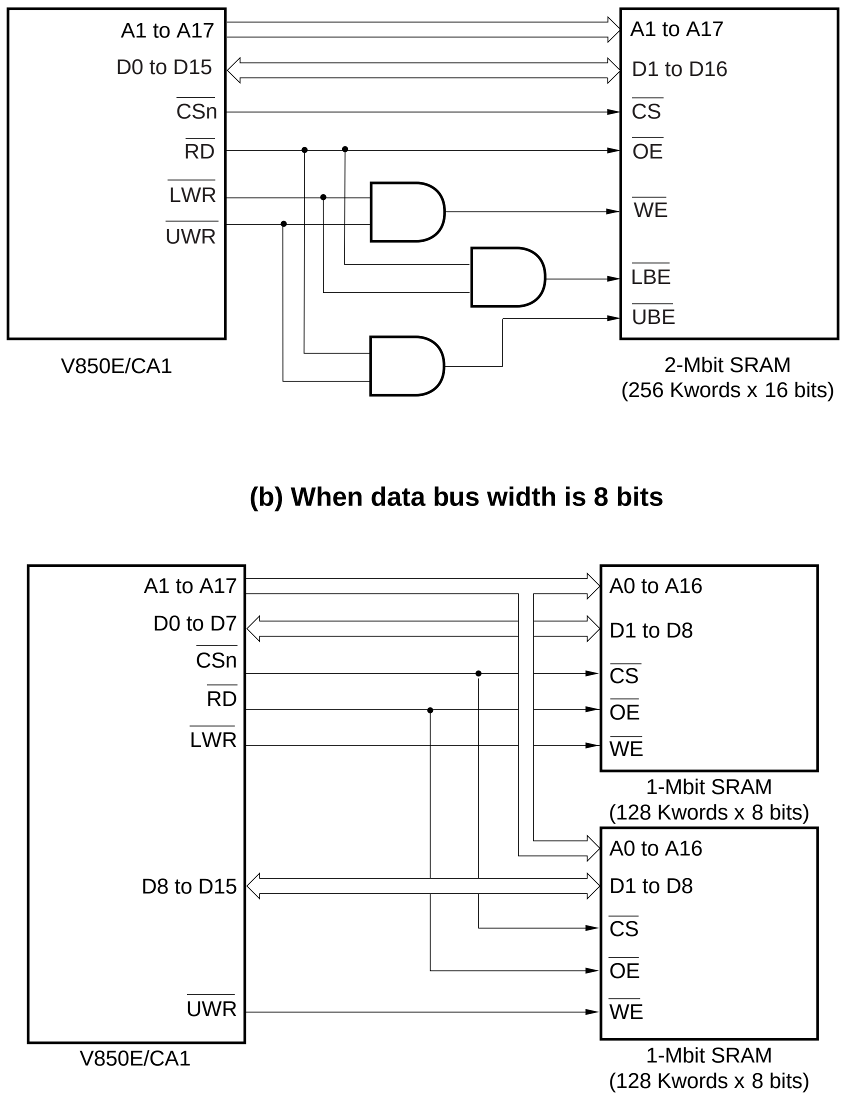
- Up to 3 idle states can be inserted after the read/write cycle through setting of the BCC register.

- Up to 3 address set up wait starts can be inserted through setting of the ASC register.

5.1.2 SRAM connections

An example of connection to SRAM is shown below.

Figure 5-1: Example of Connection to SRAM



(a) When data bus width is 16 bits

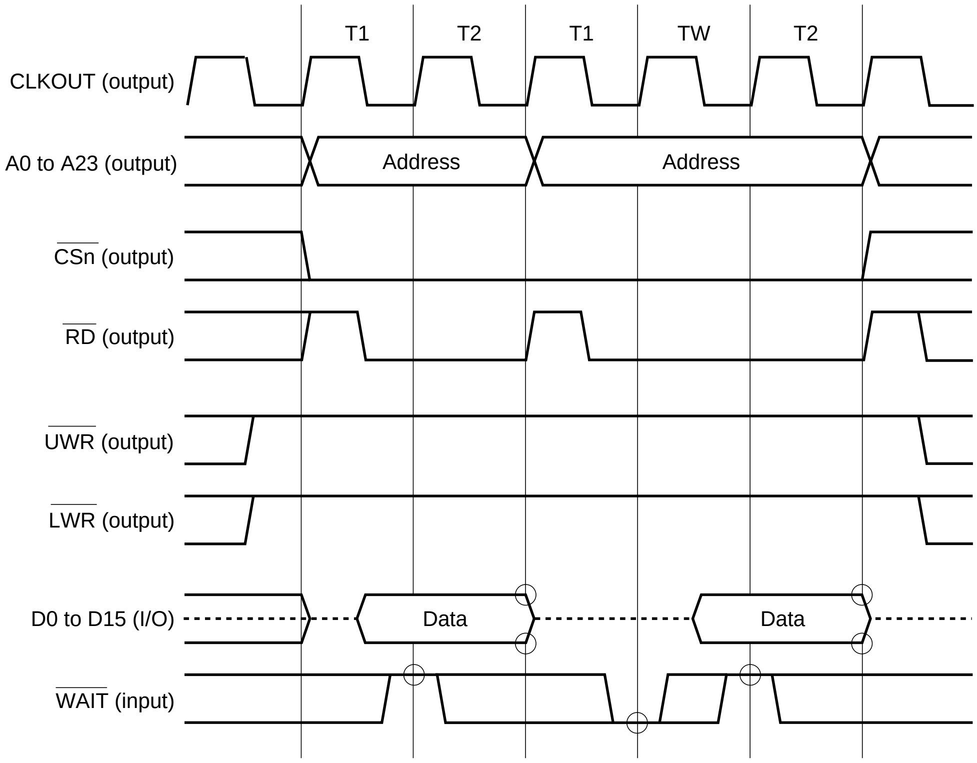
(b) When data bus width is 8 bits

Remark: $\overline{\text{CSn}}$ = $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$

5.1.3 SRAM, external ROM, external I/O access

Figure 5-2: SRAM, External ROM, External I/O Access Timing (1/6)

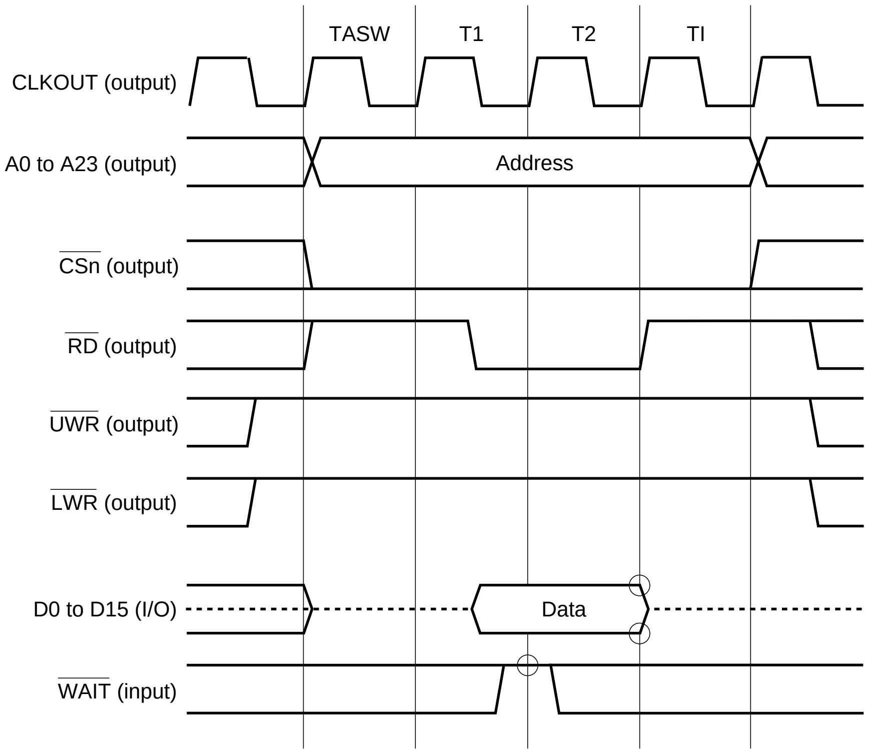
(a) During read


T1
T2
T1
TW
T2
CLKOUT (output)
A0 to A23 (output)
Address
Address
$\overline{\text{CSn}}$ (output)
$\overline{\text{RD}}$ (output)
$\overline{\text{UWR}}$ (output)
$\overline{\text{LWR}}$ (output)
D0 to D15 (I/O)
Data
Data
$\overline{\text{WAIT}}$ (input)

Remarks: 1. The circles ❍ indicate the sampling timing.
2. The broken line indicates the high-impedance state.
3. $\overline{\text{CSn}}$ = $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$

Figure 5-2: SRAM, External ROM, External I/O Access Timing (2/6)

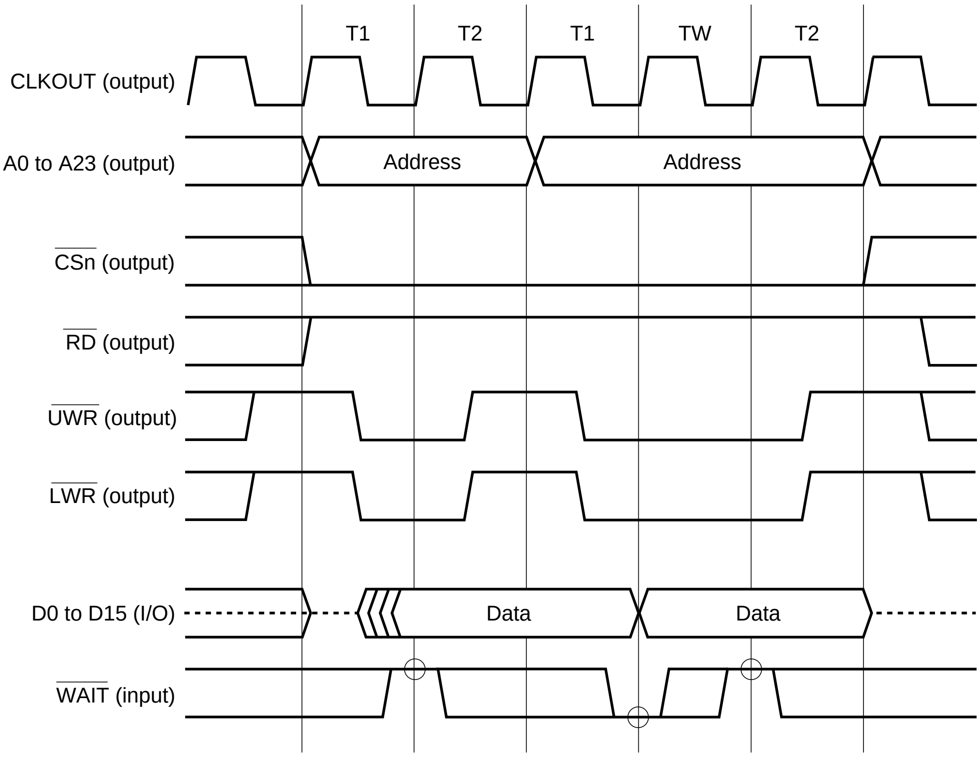
(b) During read (address setup wait, idle state insertion)



Remarks: **1.** The circles ❍ indicate the sampling timing.
2. The broken line indicates the high-impedance state.
3. $\overline{\text{CSn}}$ = $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$

Figure 5-2: SRAM, External ROM, External I/O Access Timing (3/6)

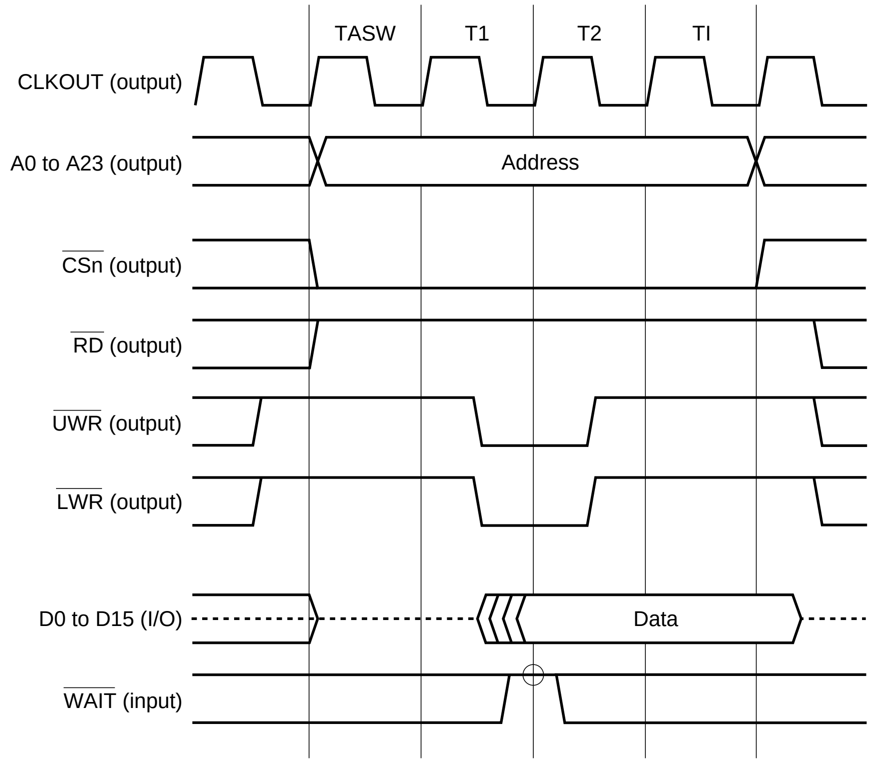
(c) During write


T1
T2
T1
TW
T2
CLKOUT (output)
A0 to A23 (output)
Address
Address
$\overline{\text{CSn}}$ (output)
$\overline{\text{RD}}$ (output)
$\overline{\text{UWR}}$ (output)
$\overline{\text{LWR}}$ (output)
D0 to D15 (I/O)
Data
Data
$\overline{\text{WAIT}}$ (input)

Remarks: **1.** The circles ❍ indicate the sampling timing.
2. The broken line indicates the high-impedance state.
3. $\overline{\text{CSn}}$ = $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$

Figure 5-2: SRAM, External ROM, External I/O Access Timing (4/6)

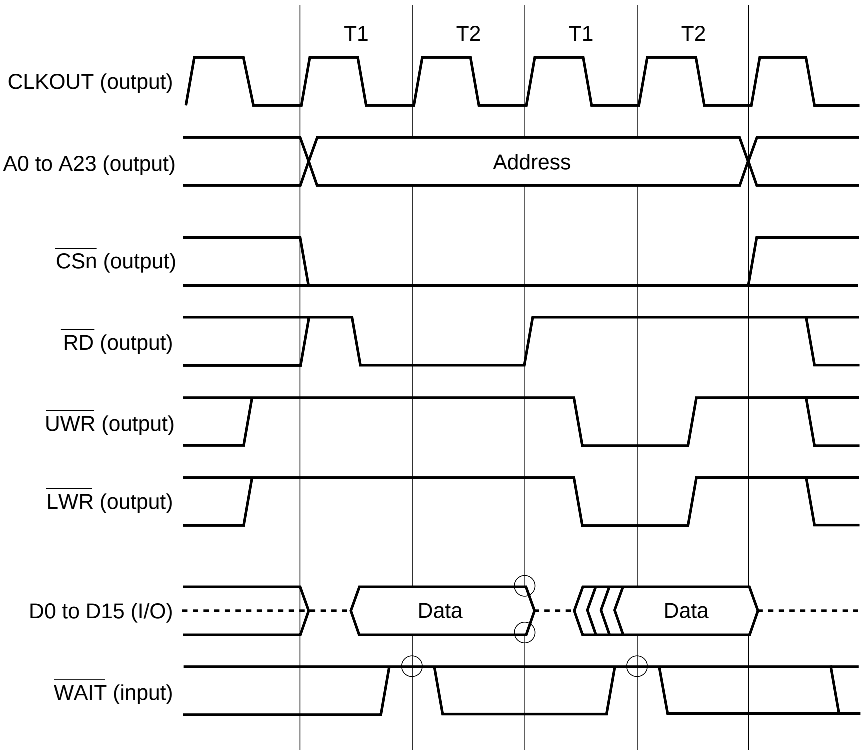
(d) During write (address setup wait, idle state insertion)



Remarks: **1.** The circles ❍ indicate the sampling timing.
2. The broken line indicates the high-impedance state.
3. $\overline{\text{CSn}}$ = $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$

Figure 5-2: SRAM, External ROM, External I/O Access Timing (5/6)

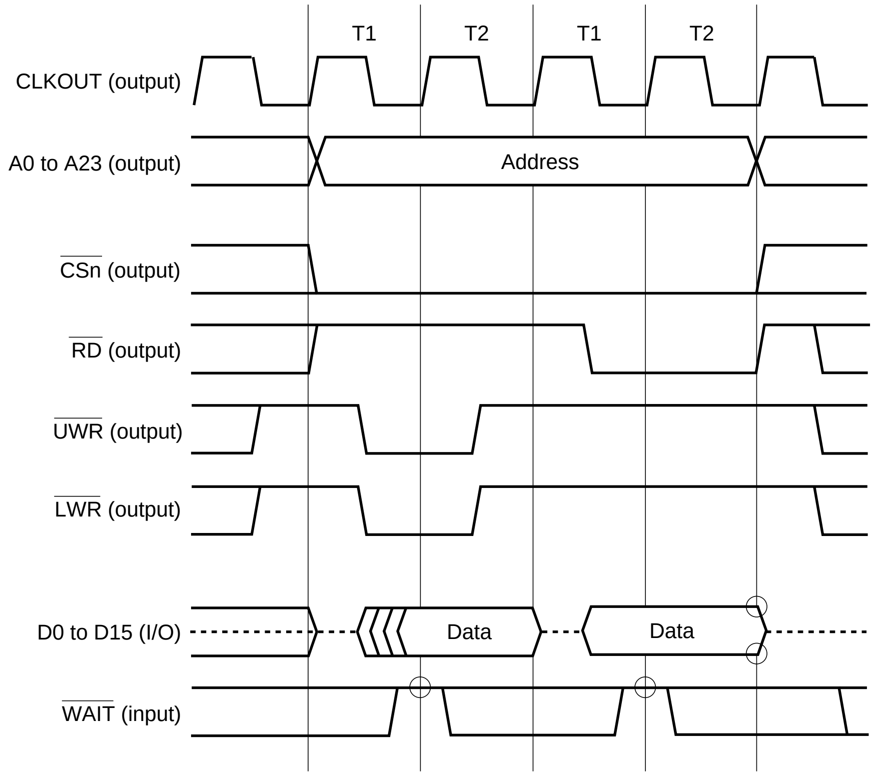
(e) When read → write operation



Remarks: **1.** The circles ❍ indicate the sampling timing.
2. The broken line indicates the high-impedance state.
3. $\overline{\text{CSn}}$ = $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$

Figure 5-2: SRAM, External ROM, External I/O Access Timing (6/6)

(f) When write → read operation



Remarks: **1.** The circles ❍ indicate the sampling timing.
2. The broken line indicates the high-impedance state.
3. $\overline{\text{CSn}}$ = $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$

5.2 Page ROM Controller (ROMC)

The page ROM controller (ROMC) is provided for access to ROM (page ROM) with the page access function.
Comparison of addresses with the immediately preceding bus cycle is carried out and wait control for normal access (off-page) and page access (on-page) is executed. This controller can handle page widths from 8 to 128 bytes.

5.2.1 Features

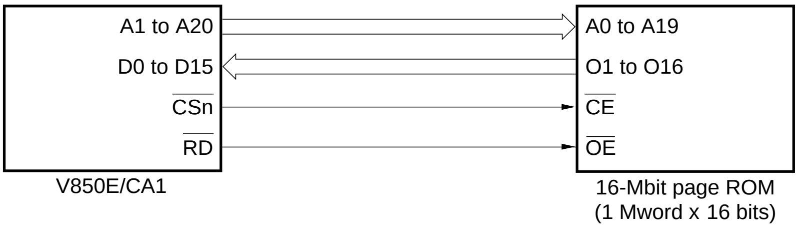
- Direct connection to 8-bit/16-bit page ROM supported
- In case of 16-bit bus width: 4/8/16/32/64 word page access supported
- In case of 8-bit bus width: 8/16/32/64/128 word page access supported
- Page ROM access a minimum of 2 states.
- On-page judgment function
- Addresses to be compared can be changed through setting of the PRC register.
- Up to 7 states of programmable data waits can be inserted during the on-page cycle through setting of the PRC register.
- Up to 7 states of programmable data wait can be inserted during the off-page cycle through setting of the DWC0 and DWC1 registers.
- Waits can be controlled with pin input.

5.2.2 Page ROM connections

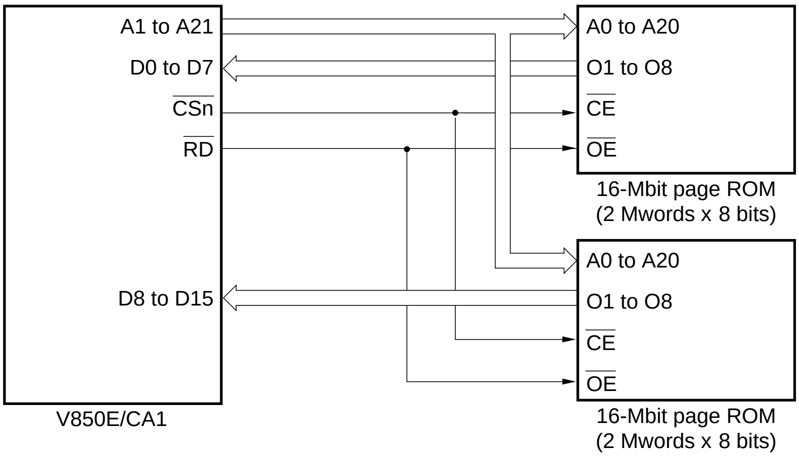
Examples of page ROM connections are shown below.

Figure 5-3: Example of Page ROM Connections

(a) In case of 16-bit data bus width



(b) In case of 8-bit data bus width



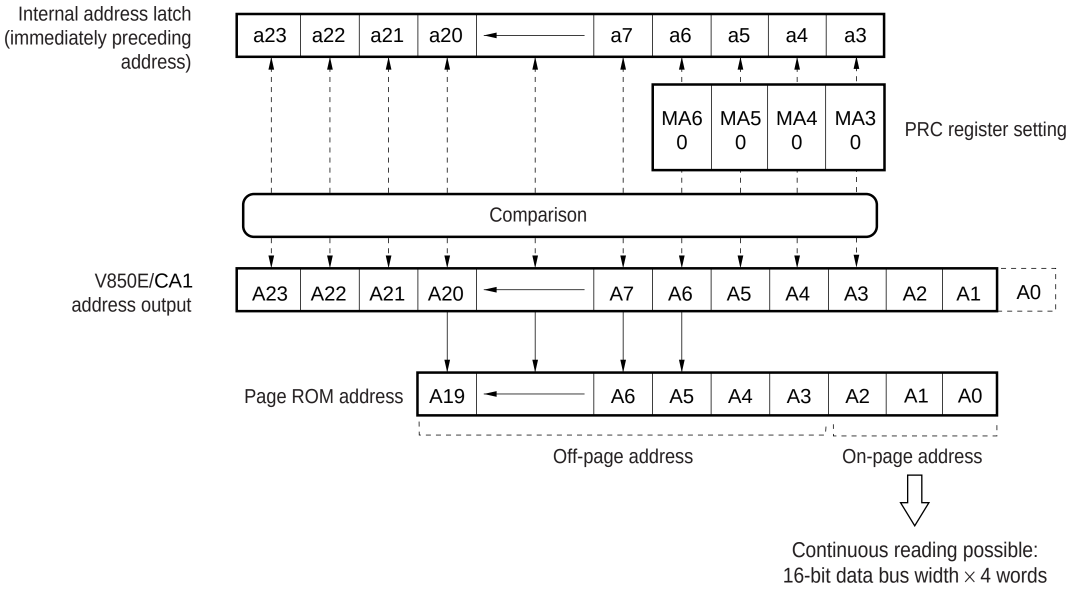
Remark: $\overline{\text{CSn}}$ = $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$

5.2.3 On-page/off-page judgment

Whether a page ROM cycle is on-page or off-page is judged by latching the address of the previous cycle and comparing it with the address of the current cycle.
Through the page ROM configuration register (PRC), according to the configuration of the connected page ROM and the number of continuously readable bits, one of the addresses (A3 to A5) is set as the masking address (no comparison is made).

Figure 5-4: On-Page/Off-Page Judgment during Page ROM Connection (1/2)

(a) In case of 16-Mbit (1 M × 16 bits) page ROM (4-word page access)


Internal address latch (immediately preceding address)
a23 a22 a21 a20 a7 a6 a5 a4 a3
MA6 0 MA5 0 MA4 0 MA3 0
PRC register setting
Comparison
V850E/CA1 address output
A23 A22 A21 A20 A7 A6 A5 A4 A3 A2 A1 A0
Page ROM address
A19 A6 A5 A4 A3 A2 A1 A0
Off-page address
On-page address
Continuous reading possible: 16-bit data bus width × 4 words

(b) In case of 16-Mbit (1 M × 16 bits) page ROM (8-word page access)

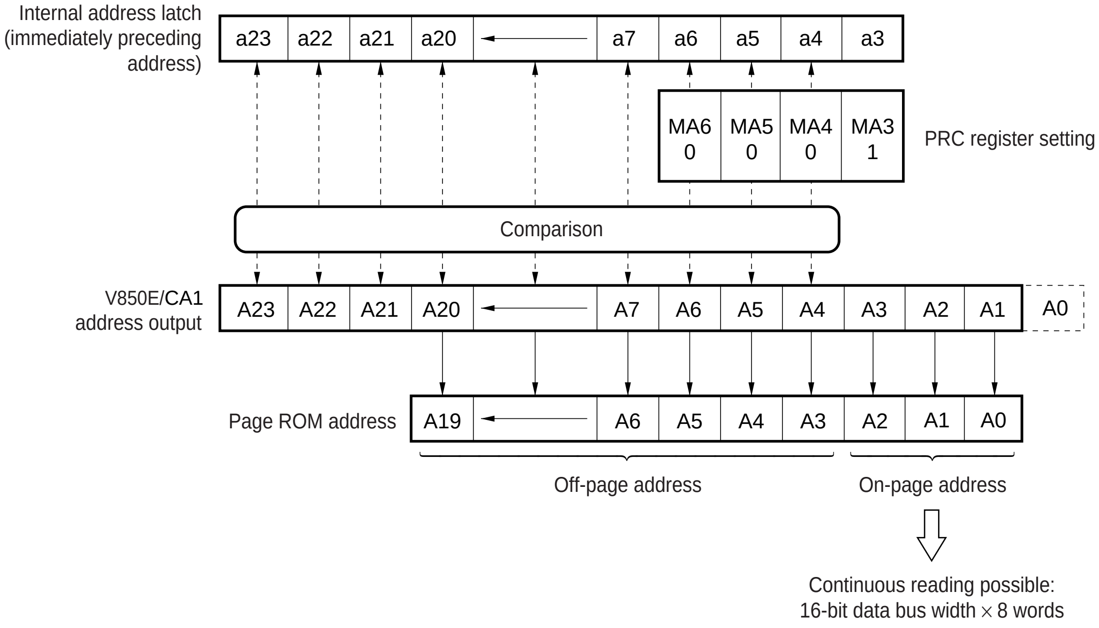
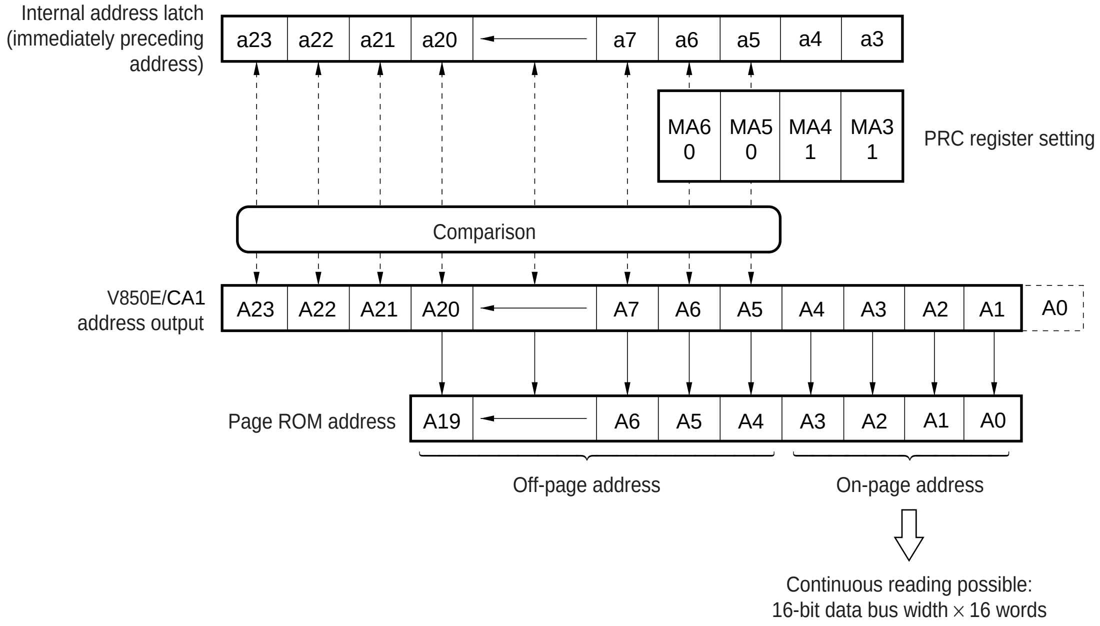

Internal address latch (immediately preceding address)
a23 a22 a21 a20 a7 a6 a5 a4 a3
MA6 0 MA5 0 MA4 0 MA3 1
PRC register setting
Comparison
V850E/CA1 address output
A23 A22 A21 A20 A7 A6 A5 A4 A3 A2 A1 A0
Page ROM address
A19 A6 A5 A4 A3 A2 A1 A0
Off-page address
On-page address
Continuous reading possible: 16-bit data bus width × 8 words

Figure 5-4: On-Page/Off-Page Judgment during Page ROM Connection (2/2)

(c) In case of 32-Mbit (2 M × 16 bits) page ROM (16-word page access)



5.2.4 Page ROM configuration register (PRC)

This register specifies whether page ROM on-page access is enabled or disabled. If on-page access is enabled, the masking address (no comparison is made) out of the addresses (A3 to A6) corresponding to the configuration of the page ROM being connected to and the number of bits that can be read continuously, as well as the number of waits corresponding to the internal system clock, are set.
This register can be read/written in 16-bit units.

Figure 5-5: Page ROM Configuration Register (PRC)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
PRC	0	PRW2	PRW1	PRW0	0	0	0	0	0	0	0	0	MA6	MA5	MA4	MA3	FFFFF49AH	7000H

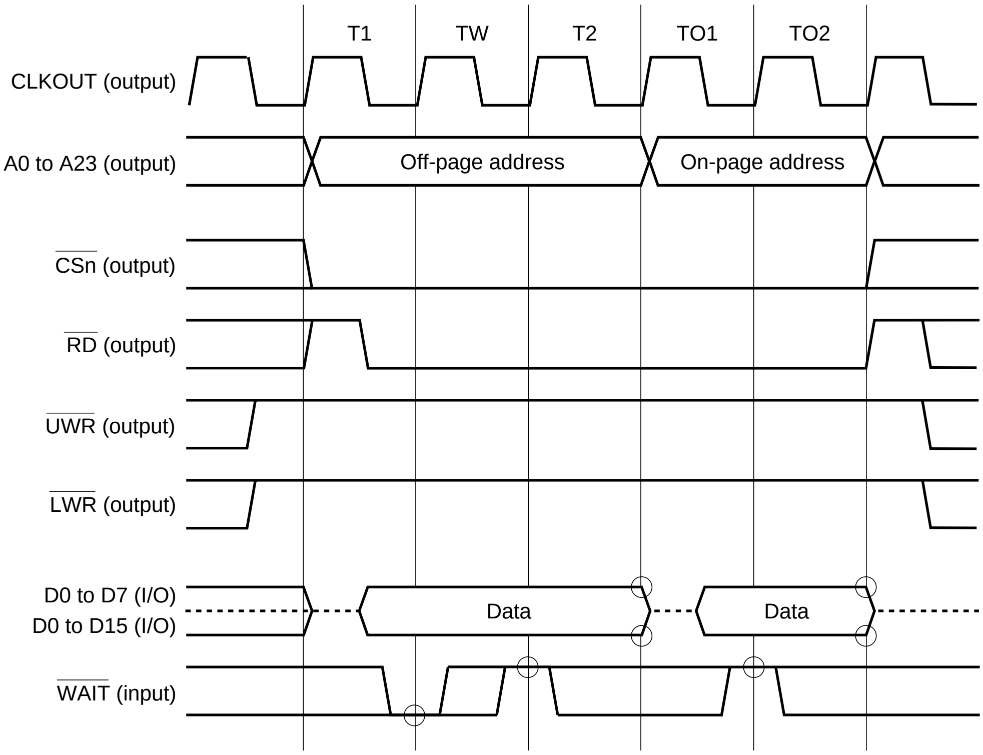
<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>14 to 12</td><td>PRW2 to PRW0</td><td>Page-ROM On-page Wait Control
Sets the number of waits corresponding to the internal system clock.
The number of waits set by this bit are inserted only when on-page. When off-page, the waits set by registers DWC0 and DWC1 are inserted.
<table>
<tr><th>PRW2</th><th>PRW1</th><th>PRW0</th><th>Number of Inserted Wait Cycles</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</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7</td></tr>
</table></td></tr>
<tr><td>3 to 0</td><td>MA6 to MA3</td><td>Mask Address
Each respective address (A6 to A3) corresponding to MA6 to MA3 is masked (masked by 1). The masked address is not subject to comparison during on/off-page judgment.
It is set according to the number of continuously readable bits.
<table>
<tr><th>MA6</th><th>MA5</th><th>MA4</th><th>MA3</th><th>Number of Continuously Readable Bits</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>4 words × 16 bits (8 words × 8 bits)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>8 words × 16 bits (16 words × 8 bits)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>16 words × 16 bits (32 words × 8 bits)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>32 words × 16 bits (64 words × 8 bits)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>64 words × 16 bits (128 words × 8 bits)</td></tr>
</table></td></tr>
</table>

Caution: Write to the PRC register after reset, and then do not change the set value. Also, do not access an external memory area other than that for this initialization routine until initial setting of the PRC register is finished. However, it is possible to access external memory areas whose initialization has been finished.

5.2.5 Page ROM access

Figure 5-6: Page ROM Access Timing (1/4)

(a) During read (when half word/word access with 8-bit bus width or when word access with 16-bit bus width)


T1
TW
T2
TO1
TO2
CLKOUT (output)
A0 to A23 (output)
Off-page address
On-page address
$\overline{\text{CSn}}$ (output)
$\overline{\text{RD}}$ (output)
$\overline{\text{UWR}}$ (output)
$\overline{\text{LWR}}$ (output)
D0 to D7 (I/O)
D0 to D15 (I/O)
Data
Data
$\overline{\text{WAIT}}$ (input)

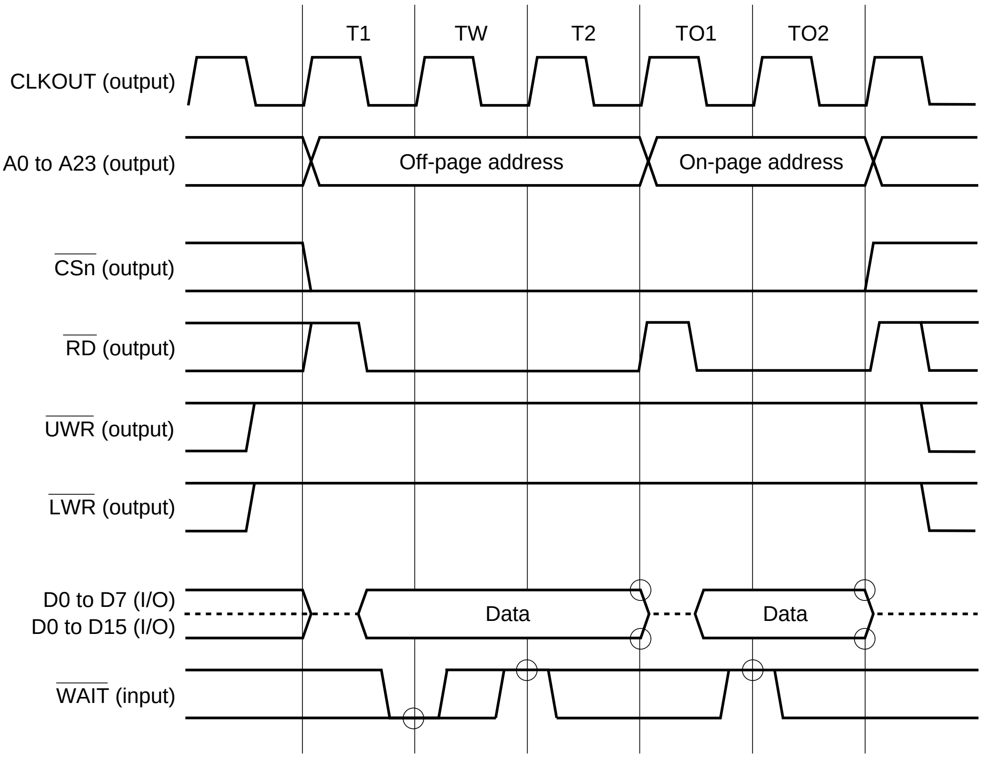
Remarks: 1. The circles ❍ indicate the sampling timing.

2. The broken line indicates the high-impedance state.

3. $\overline{\text{CSn}}$ = $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$

Figure 5-6: Page ROM Access Timing (2/4)

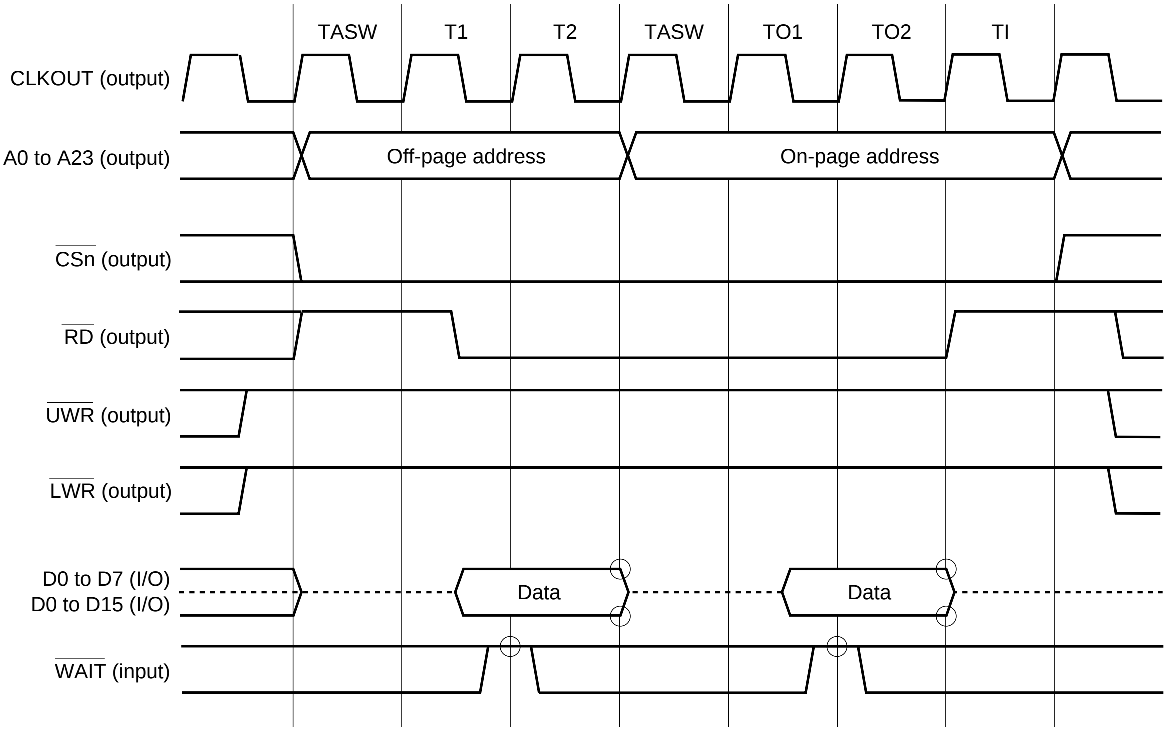
(b) During read (when byte access with 8-bit bus width or when byte/half word access with 16-bit bus width)



Remarks: **1.** The circles ❍ indicate the sampling timing.
2. The broken line indicates the high-impedance state.
3. $\overline{\text{CSn}}$ = $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$

Figure 5-6: Page ROM Access Timing (3/4)

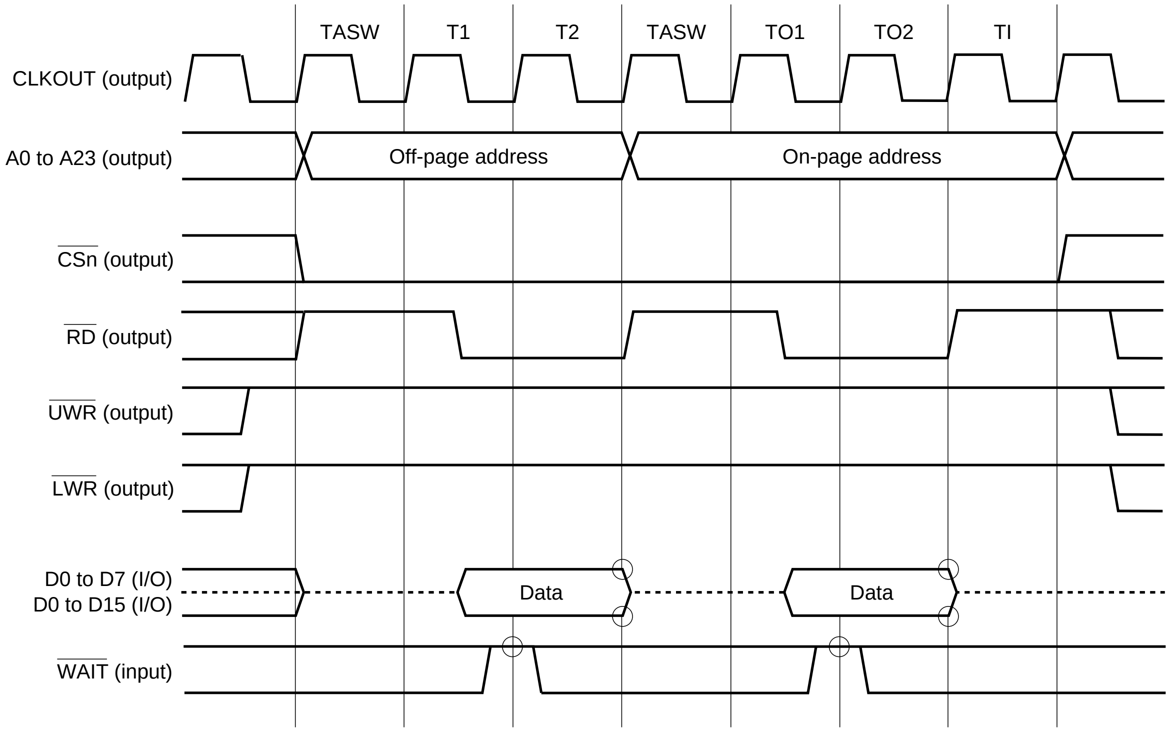
(c) During read (address setup wait, idle state insertion) (when half word/word access with 8-bit bus width or when word access with 16-bit bus width)



Remarks: **1.** The circles ❍ indicate the sampling timing.
2. The broken line indicates the high-impedance state.
3. $\overline{\text{CSn}}$ = $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$

Figure 5-6: Page ROM Access Timing (4/4)

(d) During read (address setup wait, idle state insertion) (when byte access with 8-bit bus width or when byte/half word access with 16-bit bus width)



Remarks: **1.** The circles ❍ indicate the sampling timing.
2. The broken line indicates the high-impedance state.
3. $\overline{\text{CSn}}$ = $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$

[MEMO]

Chapter 6 DMA Functions (DMA Controller)

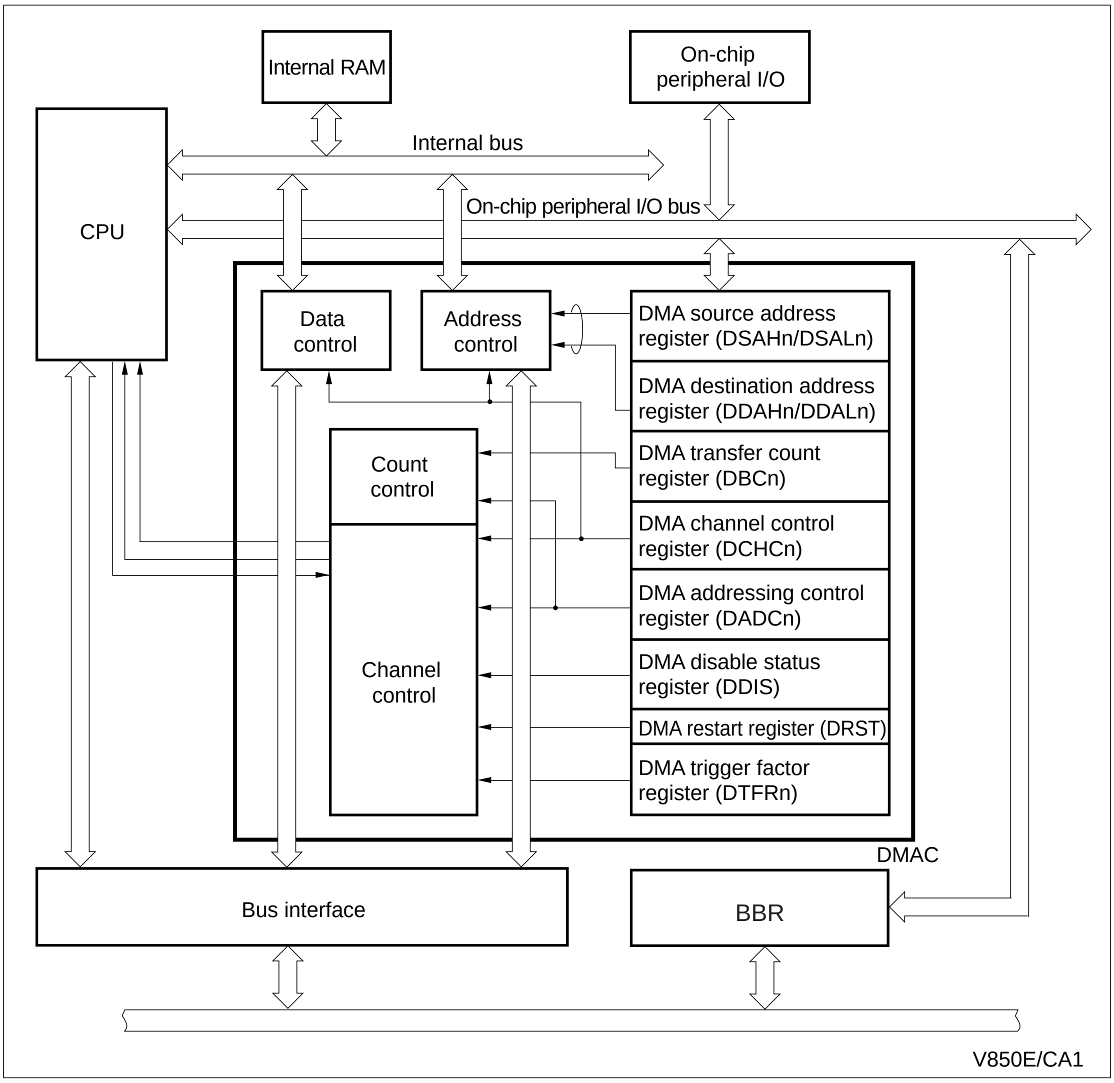
The V850E/CA1 / ATOMIC includes a direct memory access (DMA) controller (DMAC) that executes and controls DMA transfer.
The DMAC controls data transfer between memory and I/O or among I/Os, based on DMA requests issued by the on-chip peripheral I/O, or software triggers (memory refers to internal RAM).

6.1 Features

- 4 independent DMA channels
- Transfer units: 8, 16 and 32 bits
- Maximum transfer count: 65,536 (2^{16})
- Two-cycle transfer
- Three transfer modes
  - Single transfer mode
  - Single-step transfer mode
  - Block transfer mode
- Transfer requests
  - Request by interrupts from on-chip peripheral I/O
  - Requests by software trigger
- Transfer objects
  - Internal RAM ↔ I/O
  - I/O ↔ Internal RAM
  - I/O ↔ I/O
- Next address setting function

6.2 Configuration

Figure 6-1: Block Diagram of DMA Controller Configuration


Internal RAM
On-chip peripheral I/O
Internal bus
CPU
On-chip peripheral I/O bus
Data control
Address control
DMA source address register (DSAHn/DSALn)
DMA destination address register (DDAHn/DDALn)
Count control
DMA transfer count register (DBCn)
DMA channel control register (DCHCn)
DMA addressing control register (DADCn)
Channel control
DMA disable status register (DDIS)
DMA restart register (DRST)
DMA trigger factor register (DTFRn)
DMAC
Bus interface
BBR
V850E/CA1

Remark: n = 0 to 3

6.3 Control Registers

6.3.1 DMA source address registers 0 to 3 (DSA0 to DSA3)

These registers are used to set the DMA source addresses (28 bits each) for DMA channel n (n = 0 to 3). 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 **6.9 Next Address Setting Function**)

(1) DMA source address registers 0H to 3H (DSAH0 to DSAH3)

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

Caution: When setting an address of a peripheral I/O register for the source address, be sure to specify an address between FFFF000H and FFFFFFFH. An address of the peripheral I/O register image (3FFF000H to 3FFFFFFH) must not be specified.

Figure 6-2: DMA Source Address Registers H0 to H3 (DSAH0 to DSAH3)

	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: 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) DMA source address registers L0 to L3 (DSAL0 to DSAL3)

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

Figure 6-3: DMA Source Address Registers L0 to L3 (DSAL0 to DSAL3)

	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.

6.3.2 DMA destination address registers 0 to 3 (DDA0 to DDA3)

These registers are used to set the DMA destination address (28 bits each) for DMA channel n (n = 0 to 3). 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 **6.9 Next Address Setting Function**)

(1) DMA destination address registers H0 to H3 (DDAH0 to DDAH3)

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

Caution: When setting an address of a peripheral I/O register for the destination address, be sure to specify an address between FFFF000H and FFFFFFFH. An address of the peripheral I/O register image (3FFF000H to 3FFFFFFH) must not be specified.

Figure 6-4: DMA Destination Address Registers 0H to 3H (DDA0H to DDA3H)

	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: 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) DMA destination address registers L0 to L3 (DDAL0 to DDAL3)

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

Figure 6-5: DMA Destination Address Registers L0 to L3 (DDAL0 to DDAL3)

	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.

6.3.3 DMA transfer count registers 0 to 3 (DBC0 to DBC3)

These 16-bit registers are used to set the transfer counts for DMA channels n (n = 0 to 3). 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 **6.9 Next Address Setting Function**)
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 immediately before. (Refer to **6.9 Next Address Setting Function**)
These registers can be read/written in 16-bit units.

Figure 6-6: DMA Transfer Count Registers 0 to 3 (DBC0 to DBC3)

	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>

6.3.4 DMA addressing control registers 0 to 3 (DADC0 to DADC3)

These 16-bit registers are used to control the DMA transfer modes for DMA channel n (n = 0 to 3). These registers cannot be accessed during DMA operation.
They can be read/written in 16-bit units.

Figure 6-7: DMA Addressing Control Registers 0 to 3 (DADC0 to DADC3)

	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 (n = 0 to 3).
<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 (n = 0 to 3).
<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>
<tr><td>3, 2</td><td>TM1, TM0</td><td>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>Single-step transfer mode</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></td></tr>
</table>

6.3.5 DMA channel control registers 0 to 3 (DCHC0 to DCHC3)

These 8-bit registers are used to control the DMA transfer operating mode for DMA channel n (n = 0 to 3).
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.)

Figure 6-8: DMA Channel Control Registers 0 to 3 (DCHC0 to DCHC3)

	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	This status bit 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 this bit is set to 1 at terminal count output, the Enn bit is not cleared to 0 and the DMA transfer enable state is retained. Moreover, the next DMA transfer request can be accepted even when the TCn bit is not read. 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. This bit is cleared to 0 when DMA transfer ends. It is also cleared to 0 when DMA transfer is forcibly terminated by means of setting the INITn bit to 1 or by NMI input. 0: DMA transfer disabled 1: DMA transfer enabled

Remark: n = 0 to 3

6.3.6 DMA disable status register (DDIS)

This register holds the contents of the ENn bit of the DCHCn register during NMI input (n = 0 to 3).
This register is read-only in 8-bit or 1-bit units.

Figure 6-9: DMA Disable Status Register (DDIS)

	7	6	5	4	3	2	1	0	Address	Initial value
DDIS	0	0	0	0	CH3	CH2	CH1	CH0	FFFFF0F0H	00H

Bit Position	Bit Name	Function
3 to 0	CH3 to CH0	Reflects the contents of the ENn bit of the DCHCn register during NMI input. The contents of this register are held until the next NMI input or until the system is reset.

6.3.7 DMA restart register (DRST)

This register is used to restart DMA transfer that has been forcibly interrupted by a non-maskable interrupt (NMI). The ENn bit of this register and the ENn bit of the DCHCn register are linked to each other (n = 0 to 3). Following forcible interrupt by NMI input, the DMA channel that was interrupted is confirmed from the contents of the DDIS register, and DMA transfer is restarted by setting the ENn bit of the corresponding channel to 1.
This register can be read/written in 8-bit or 1-bit units.

Figure 6-10: DMA Restart Register (DRST)

	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

6.3.8 DMA trigger factor registers 0 to 3 (DTFR0 to DTFR3)

These 8-bit registers are used to control the DMA transfer start trigger through interrupt requests from on-chip peripheral I/O.
The interrupt requests set with these registers serve as DMA transfer start factors.
These registers can be read/written in 8-bit/1-bit units.

Figure 6-11: DMA Trigger Factor Registers 0 to 3 (DTFR0 to DTFR3) (1/3)

	7	6	5	4	3	2	1	0	Address	Initial value
DTFR0	0	0	IFC5	IFC4	IFC3	IFC2	IFC1	IFC0	FFFFF810H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
DTFR1	0	0	IFC5	IFC4	IFC3	IFC2	IFC1	IFC0	FFFFF812H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
DTFR2	0	0	IFC5	IFC4	IFC3	IFC2	IFC1	IFC0	FFFFF814H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
DTFR3	0	0	IFC5	IFC4	IFC3	IFC2	IFC1	IFC0	FFFFF816H	00H

Figure 6-11: DMA Trigger Factor Registers 0 to 3 (DTFR0 to DTFR3) (2/3)

Bit Position	Bit Name	Function
5 to 0	IFC5 to IFC0	Sets the interrupt source that serves as the DMA start factor.

IFC5	IFC4	IFC3	IFC2	IFC1	IFC0	Interrupt Source
0	0	0	0	0	0	DMA request from on-chip peripheral I/O disabled
0	0	0	0	0	1	CINTLPOW
0	0	0	0	1	0	AD/INTDET
0	0	0	0	1	1	INTWT
0	0	0	1	0	0	TINTCMD0
0	0	0	1	0	1	TINTCMD1
0	0	0	1	1	0	INTWTI
0	0	0	1	1	1	INT0
0	0	1	0	0	0	INT1
0	0	1	0	0	1	INT2
0	0	1	0	1	0	TINTOVE00
0	0	1	0	1	1	TINTOVE10
0	0	1	1	0	0	TINTCCE00/INTPE00
0	0	1	1	0	1	TINTCCE10/INTPE10
0	0	1	1	1	0	TINTCCE20/INTPE20
0	0	1	1	1	1	TINTCCE30/INTPE30
0	1	0	0	0	0	TINTCCE40/INTPE40
0	1	0	0	0	1	TINTCCE50/INTPE50
0	1	0	0	1	0	TINTOVE01
0	1	0	0	1	1	TINTOVE11
0	1	0	1	0	0	TINTCCE01/INTPE01
0	1	0	1	0	1	TINTCCE11/INTPE11
0	1	0	1	1	0	TINTCCE21/INTPE21
0	1	0	1	1	1	TINTCCE31/INTPE31
0	1	1	0	0	0	TINTCCE41/INTPE41
0	1	1	0	0	1	TINTCCE51/INTPE51
0	1	1	0	1	0	TINTOVE02
0	1	1	0	1	1	TINTOVE12
0	1	1	1	0	0	TINTCCE02/INTPE02

Figure 6-11: DMA Trigger Factor Registers 0 to 3 (DTFR0 to DTFR3) (3/3)

Bit Position	Bit Name	Function
5 to 0	IFC5 to IFC0	Sets the interrupt source that serves as the DMA start factor.

IFC5	IFC4	IFC3	IFC2	IFC1	IFC0	Interrupt Source
0	1	1	1	0	1	TINTCCE12/INTPE12
0	1	1	1	1	0	TINTCCE22/INTPE22
0	1	1	1	1	1	TINTCCE32/INTPE32
1	0	0	0	0	0	TINTCCE42/INTPE42
1	0	0	0	0	1	TINTCCE52/INTPE52
1	0	0	0	1	0	INTAD
1	0	0	0	1	1	INTMAC
1	0	0	1	0	0	INTACT
1	0	0	1	0	1	CAN1REC
1	0	0	1	1	0	CAN1TRX
1	0	0	1	1	1	CAN1ERR
1	0	1	0	0	0	CAN2REC
1	0	1	0	0	1	CAN2TRX
1	0	1	0	1	0	CAN2ERR
1	0	1	0	1	1	CAN3REC
1	0	1	1	0	0	CAN3TRX
1	0	1	1	0	1	CAN3ERR
1	0	1	1	1	0	INTCSI0
1	0	1	1	1	1	INTCSI1
1	1	0	0	0	0	INTSER0
1	1	0	0	0	1	INTSR0
1	1	0	0	1	0	INTST0
1	1	0	0	1	1	INTSER1
1	1	0	1	0	0	INTSR1
1	1	0	1	0	1	INTST1
1	1	0	1	1	0	INTSER2
1	1	0	1	1	1	INTSR2
1	1	1	0	0	0	INTST2
Other than above						Setting prohibited

Cautions: 1. Be sure to stop DMA operation before making changes to DTFRn register settings (n = 0 to 3).

2. An interrupt request input in a standby mode (IDLE or software STOP mode) cannot be used as a DMA transfer start factor.

6.4 DMA Bus States

6.4.1 Types of bus states

The DMAC bus states consist of the following 8 states.

(1) TI state

The TI state is an idle state, during which no access request is issued.

(2) T0 state

DMA transfer ready state (state in which a DMA transfer request has been issued and the bus mastership is acquired for the first DMA transfer).

(3) T1R state

The bus enters the T1R state at the beginning of a read operation in the two-cycle transfer mode. Address driving starts. After entering the T1R state, the bus invariably enters the T2R state.

(4) T2R state

The T2R state corresponds to the last state of a read operation in the two-cycle transfer mode, or to a wait state.
In the last T2R state, read data is sampled. After entering the last T2R state, the bus invariably enters the T1W state.

(5) T2RI state

State in which the bus is ready for DMA transfer to on-chip peripheral I/O or internal RAM (state in which the bus mastership is acquired for DMA transfer to on-chip peripheral I/O or internal RAM).
After entering the last T2RI state, the bus invariably enters the T1W state.

(6) T1W state

The bus enters the T1W state at the beginning of a write operation in the two-cycle transfer mode. Address driving starts. After entering the T1W state, the bus invariably enters the T2W state.

(7) T2W state

The T2W state corresponds to the last state of a write operation in the two-cycle transfer mode, or to a wait state.
In the last T2W state, the write strobe signal is made inactive.

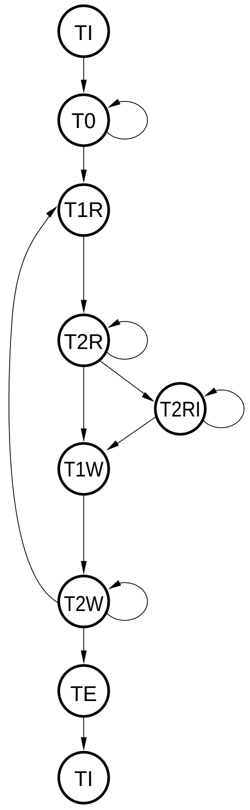
(8) TE state

The TE state corresponds to DMA transfer completion. The DMAC generates the internal DMA transfer completion signal and various internal signals are initialized. After entering the TE state, the bus invariably enters the TI state.

6.4.2 DMAC bus cycle state transition

Except for the block transfer mode, each time the processing for a DMA transfer is completed, the bus mastership is released.

Figure 6-12: DMAC Bus Cycle (Two-Cycle Transfer) State Transition



6.5 Transfer Mode

6.5.1 Single transfer mode

In single transfer mode, the DMAC releases the bus at 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.

6.5.2 Single-step transfer mode

In single-step transfer mode, the DMAC releases the bus at each byte/halfword/word transfer. Once a DMA transfer request signal is received, transfer is performed again. 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.

6.5.3 Block transfer mode

In the block transfer mode, once transfer starts, 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, another DMA transfer can be acknowledged.

6.6 Transfer Types

6.6.1 Two-cycle transfer

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 destination address is output and writing is performed from the DMAC to the destination.

6.7 Transfer Object

6.7.1 Transfer type and transfer object

Table 6-1 lists the relationships between transfer type and transfer object (√: transfer enabled, ×: transfer disabled).

Table 6-1: Relationship Between Transfer Type and Transfer Object

		Destination				
		Two-Cycle Transfer				
		Internal ROM	On-Chip Peripheral I/O	External I/O	Internal RAM	External Memory
Source	On-chip peripheral I/O	×	√	×	√	×
	External I/O	×	×	×	×	×
	Internal RAM	×	√	×	×	×
	External memory	×	×	×	×	×
	Internal ROM	×	×	×	×	×

Cautions: 1. The operation is not guaranteed for combinations of transfer destination and source marked with "×" in Table 6-1.

2. Addresses between 3FFF000H and 3FFFFFFH cannot be specified for the source and destination address of DMA transfer. Be sure to specify an address between FFFF000H and FFFFFFFH.

6.8 DMA Channel Priorities

The DMA channel priorities are fixed as follows.

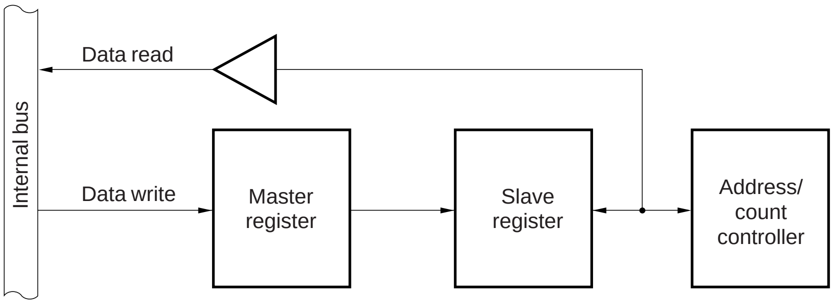
DMA channel 0 > DMA channel 1 > DMA channel 2 > DMA channel 3

These priorities are valid in the TI state only. In the block transfer mode, the channel used for transfer is never switched.
In the single-step transfer mode, if a higher priority DMA transfer request is issued while the bus is released (in the TI state), the higher priority DMA transfer request is acknowledged.

6.9 Next Address Setting 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 configuration (n = 0 to 3).
When the terminal count is issued, these registers are automatically rewritten with the value that was set immediately before.
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 to 1 (however, the DMA transfer end interrupt may be issued even if DMA transfer is automatically started).
Figure 6-13 shows the configuration of the buffer register.

Figure 6-13: Buffer Register Configuration



6.10 DMA Transfer Start Factors

There are two types of DMA transfer start factors, as shown below.

(1) Request from software

If the STGn, ENn, and TCn bits of the DCHCn register are set as follows, DMA transfer starts (n = 0 to 3).

- STGn bit = 1
- ENn bit = 1
- TCn bit = 0

(2) Request from on-chip peripheral I/O

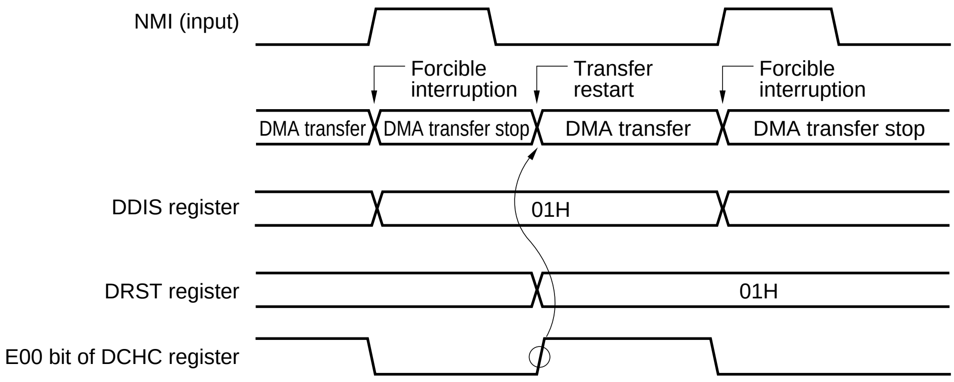
If, when the ENn and TCn bits of the DCHCn register are set as shown below, an interrupt request is issued from the on-chip peripheral I/O that is set in the DTFRn register, DMA transfer starts (n = 0 to 3).

- ENn bit = 1
- TCn bit = 0

6.11 Forcible Interruption

DMA transfer can be forcibly interrupted by NMI input during DMA transfer.
At such a time, the DMAC resets the ENn bit of the DCHCn register of all channels to 0 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 (n = 0 to 3).
In the single-step transfer mode or block transfer mode, the DMA transfer request is held in the DMAC. If the ENn bit is set to 1, DMA transfer restarts from the point where it was interrupted.
In the single transfer mode, if the ENn bit is set to 1, the next DMA transfer request is acknowledged and DMA transfer starts.

Figure 6-14: Example of Forcible Interruption of DMA Transfer



6.12 DMA Transfer End

6.12.1 DMA transfer end interrupt

When DMA transfer ends and the TCn bit of the DCHCn register is set to 1, a DMA transfer end interrupt (INTDMAn) is issued to the interrupt controller (INTC) (n = 0 to 3).

6.12.2 Terminal count output upon DMA transfer end

The terminal count signal becomes active for one clock during the last DMA transfer cycle.

6.13 Forcible Termination

In addition to the forcible interruption operation by means of NMI input, DMA transfer can be forcibly terminated by the INITn bit of the DCHCn register (n = 0 to 3).

Remark: 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 **6.9 Next Address Setting Function** and **6.3.4 DMA addressing control registers 0 to 3 (DADC0 to DADC3)**).

6.14 Precautions

(1) Memory boundary

The transfer operation is not guaranteed if the source or the destination address exceeds the area of DMA objects (internal RAM, or peripheral I/O) during DMA transfer.

(2) Transfer of misaligned data

DMA transfer of 16-bit/32-bit bus width misaligned data is not supported.

(3) Times related to DMA transfer

The overhead before and after DMA transfer and the minimum execution clock for DMA transfer are shown below.

- Internal RAM access: 2 clocks

(4) Bus arbitration for CPU

The CPU can access on-chip peripheral I/O, and internal RAM not undergoing DMA transfer.
While data transfer is being executed between internal RAMs, the CPU can access external memory and peripheral I/O.

(5) Interrupt factors

DMA transfer is interrupted if a bus hold is issued.
If the factor (bus hold) interrupting DMA transfer disappears, DMA transfer promptly restarts.

Chapter 7 Interrupt/Exception Processing Function

The V850E/CA1 / ATOMIC is provided with a dedicated interrupt controller (INTC) for interrupt servicing and can process a total of 62 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.
The V850E/CA1 / ATOMIC 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. Interrupt servicing starts after no fewer than 11 system clocks (550 ns (@ 20 MHz)) following the generation of an interrupt request.

7.1 Features

- Interrupts
  - Non-maskable interrupts: 2 sources
  - Maskable interrupts: 60 sources
  - 8 levels of programmable priorities (maskable interrupts)
  - Multiple interrupt control according to priority
  - Masks can be specified for each maskable interrupt request.
  - Noise elimination[Note], edge detection, and valid edge specification for external interrupt request signals.

Note: For details refer to chapter 7.4 “Noise Elimination Circuit” on page 207.

- Exceptions
  - Software exceptions: 32 sources
  - Exception traps: 2 sources (illegal opcode exception and debug trap)

Interrupt/exception sources are listed in Table 7-1.

Table 7-1: Interrupt/Exception Source List (Sheet 1 of 3)

Type	Classification	Interrupt/Exception Source				Default Priority	Exception Code	Handler Address	Restored PC
		Name	Controlling Register	Generating Source	Generating Unit				
Reset	Interrupt	RESET	–	$\overline{\text{RESET}}$ input	Pin	–	0000H	00000000H	Undefined
Non-maskable	Interrupt	NMIVC	–	VCMPOUT / NMI Input	Voltage comparator / NMI-Pin	–	0010H	00000010H	nextPC
		NMIWDT	–	Watchdog timer	WDT	–	0020H	00000020H	nextPC
Software exception	Exception	TRAP0n$^{\text{Note}}$	–	TRAP instruction	–	–	004nH$^{\text{Note}}$	00000040H	nextPC
	Exception	TRAP1n$^{\text{Note}}$	–	TRAP instruction	–	–	005nH$^{\text{Note}}$	00000050H	nextPC
Exception trap	Exception	ILGOP/ DBTRAP	–	Illegal opcode/ DBTRAP instruction	–	–	0060H	00000060H	nextPC
Maskable	Interrupt	CINTLPOW	PIC0	Low Power	Voltage Comparator	0	0080H	00000080H	nextPC
	Interrupt	AD/INTDET	PIC1	Power Fail	A/D converter	1	0090H	00000090H	nextPC
	Interrupt	INTWT	PIC2	Real Time Clock Divider Tick	Watch timer	2	00A0H	000000A0H	nextPC
	Interrupt	TINTCMD0	PIC3	Compare Match	Timer D0	3	00B0H	000000B0H	nextPC
	Interrupt	TINTCMD1	PIC4	Compare Match	Timer D1	4	00C0H	000000C0H	nextPC
	Interrupt	INTWTI	PIC5	interval Timer	Watch Timer	5	00D0H	000000D0H	nextPC
	Interrupt	INT0	PIC6	INT0 input	Pin	6	00E0H	000000E0H	nextPC
	Interrupt	INT1	PIC7	INT1 input	Pin	7	00F0H	000000F0H	nextPC
	Interrupt	INT2	PIC8	INT2 input	Pin	8	0100H	00000100H	nextPC
	Interrupt	TINTOVE00	PIC9	Time base Overflow	Timer E0	9	0110H	00000110H	nextPC
	Interrupt	TINTOVE10	PIC10	Time base Overflow	Timer E0	10	0120H	00000120H	nextPC
	Interrupt	TINTCCE00 /INTPE00	PIC11	CC coincidence / Pin	TimerE0/ INTPE00	11	0130H	00000130H	nextPC
	Interrupt	TINTCCE10 /INTPE10	PIC12	CC coincidence / Pin	TimerE0/ INTPE10	12	0140H	00000140H	nextPC
	Interrupt	TINTCCE20 /INTPE20	PIC13	CC coincidence / Pin	TimerE0/ INTPE20	13	0150H	00000150H	nextPC
	Interrupt	TINTCCE30 /INTPE30	PIC14	CC coincidence / Pin	TimerE0/ INTPE30	14	0160H	00000160H	nextPC
	Interrupt	TINTCCE40 /INTPE40	PIC15	CC coincidence / Pin	TimerE0/ INTPE40	15	0170H	00000170H	nextPC
	Interrupt	TINTCCE50 /INTPE50	PIC16	CC coincidence / Pin	TimerE0/ INTPE50	16	0180H	00000180H	nextPC
	Interrupt	TINTOVE01	PIC17	Time base Overflow	Timer E1	17	0190H	00000190H	nextPC
	Interrupt	TINTOVE11	PIC18	Time base Overflow	Timer E1	18	01A0H	000001A0H	nextPC
	Interrupt	TINTCCE01 /INTPE01	PIC19	CC coincidence / Pin	TimerE1/ INTPE01	19	01B0H	000001B0H	nextPC
	Interrupt	TINTCCE11 /INTPE11	PIC20	CC coincidence / Pin	TimerE1/ INTPE11	20	01C0H	000001C0H	nextPC
	Interrupt	TINTCCE21 /INTPE21	PIC21	CC coincidence / Pin	TimerE1/ INTPE21	21	01D0H	000001D0H	nextPC
	Interrupt	TINTCCE31 /INTPE31	PIC22	CC coincidence / Pin	TimerE1/ INTPE31	22	01E0H	000001E0H	nextPC

Table 7-1: Interrupt/Exception Source List (Sheet 2 of 3)

Type	Classification	Interrupt/Exception Source				Default Priority	Exception Code	Handler Address	Restored PC
		Name	Controlling Register	Generating Source	Generating Unit				
Maskable	Interrupt	TINTCCE41/INTPE41	PIC23	CC coincidence / Pin	TimerE1/INTPE41	23	01F0H	000001F0H	nextPC
	Interrupt	TINTCCE51/INTPE51	PIC24	CC coincidence / Pin	TimerE1/INTPE51	24	0200H	00000200H	nextPC
	Interrupt	TINTOVE02	PIC25	Time base Overflow	Timer E2	25	0210H	00000210H	nextPC
	Interrupt	TINTOVE12	PIC26	Time base Overflow	Timer E2	26	0220H	00000220H	nextPC
	Interrupt	TINTCCE02/INTPE02	PIC27	CC coincidence / Pin	TimerE2/INTPE02	27	0230H	00000230H	nextPC
	Interrupt	TINTCCE12/INTPE12	PIC28	CC coincidence / Pin	TimerE2/INTPE12	28	0240H	00000240H	nextPC
	Interrupt	TINTCCE22/INTPE22	PIC29	CC coincidence / Pin	TimerE2/INTPE22	29	0250H	00000250H	nextPC
	Interrupt	TINTCCE32/INTPE32	PIC30	CC coincidence / Pin	TimerE2/INTPE32	30	0260H	00000260H	nextPC
	Interrupt	TINTCCE42/INTPE42	PIC31	CC coincidence / Pin	TimerE2/INTPE42	31	0270H	00000270H	nextPC
	Interrupt	TINTCCE52/INTPE52	PIC32	CC coincidence / Pin	TimerE2/INTPE52	32	0280H	00000280H	nextPC
	Interrupt	INTAD	PIC33	AD conversion end	A/D	33	0290H	00000290H	nextPC
	Interrupt	INTMAC	PIC34	MAC interrupt CGINTP 1-2	FCAN	34	02A0H	000002A0H	nextPC
	Interrupt	INTACT	PIC35	MAC interrupt CGINTP 7	FCAN	35	02B0H	000002B0H	nextPC
	Interrupt	CAN1REC	PIC36	CAN1 receive interrupt pending	FCAN1	36	02C0H	000002C0H	nextPC
	Interrupt	CAN1TRX	PIC37	CAN1 transmit interrupt pending	FCAN1	37	02D0H	000002D0H	nextPC
	Interrupt	CAN1ERR	PIC38	CAN1 error interrupt pending	FCAN1	38	02E0H	000002E0H	nextPC
	Interrupt	CAN2REC	PIC39	CAN2 receive interrupt pending	FCAN2	39	02F0H	000002F0H	nextPC
	Interrupt	CAN2TRX	PIC40	CAN2 transmit interrupt pending	FCAN2	40	0300H	00000300H	nextPC
	Interrupt	CAN2ERR	PIC41	CAN2 error interrupt pending	FCAN2	41	0310H	00000310H	nextPC
	Interrupt	CAN3REC	PIC42	CAN3 receive interrupt pending	FCAN3	42	0320H	00000320H	nextPC
	Interrupt	CAN3TRX	PIC43	CAN3 transmit interrupt pending	FCAN3	43	0330H	00000330H	nextPC
	Interrupt	CAN3ERR	PIC44	CAN3 error interrupt pending	FCAN3	44	0340H	00000340H	nextPC
	Interrupt	INTCSI0	PIC45	CSI0 transmission/ reception complete	CSI0	45	0350H	00000350H	nextPC
	Interrupt	INTCSI1	PIC46	CSI1 transmission / reception complete	CSI1	46	0360H	00000360H	nextPC
	Interrupt	INTSER0	PIC47	UART0 reception error	UART0	47	0370H	00000370H	nextPC
	Interrupt	INTSR0	PIC48	UART0 reception completion	UART0	48	0380H	00000380H	nextPC
	Interrupt	INTST0	PIC49	UART0 transmission completion	UART0	49	0390H	00000390H	nextPC
	Interrupt	INTSER1	PIC50	UART1 reception error	UART1	50	03A0H	000003A0H	nextPC
	Interrupt	INTSR1	PIC51	UART1 reception completion	UART1	51	03B0H	000003B0H	nextPC

Table 7-1: Interrupt/Exception Source List (Sheet 3 of 3)

Type	Classification	Interrupt/Exception Source				Default Priority	Exception Code	Handler Address	Restored PC
		Name	Controlling Register	Generating Source	Generating Unit				
Maskable	Interrupt	INTST1	PIC52	UART1 transmission completion	UART1	52	03C0H	000003C0H	nextPC
	Interrupt	INTSER2	PIC53	UART2 reception error	UART2	53	03D0H	000003D0H	nextPC
	Interrupt	INTSR2	PIC54	UART2 reception completion	UART2	54	03E0H	000003E0H	nextPC
	Interrupt	INTST2	PIC55	UART2 transmission completion	UART2	55	03F0H	000003F0H	nextPC
	Interrupt	INTDMA0	PIC56	DAM completed	DMA0	56	0400H	00000400H	nextPC
	Interrupt	INTDMA1	PIC57	DMA completed	DMA1	57	0410H	00000410H	nextPC
	Interrupt	INTDMA2	PIC58	DMA completed	DMA2	58	0420H	00000420H	nextPC
	Interrupt	INTDMA3	PIC59	DMA completed	DMA3	59	0430H	00000430H	nextPC
	Interrupt	reserved	PIC60	reserved	reserved	60	0440H	00000440H	nextPC
	Interrupt	reserved	PIC61	reserved	reserved	61	0450H	00000450H	nextPC
	Interrupt	reserved	PIC62	reserved	reserved	62	0460H	00000460H	nextPC
	Interrupt	reserved	PIC63	reserved	reserved	63	0470H	00000470H	nextPC

Note: n = 0 to FH

Remarks:

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).

7.2 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 V850E/CA1 / ATOMIC are available for the following two requests:

- NMI pin input (NMIVC)
- Non-maskable watchdog timer interrupt request (NMIWDT)

When the valid edge specified by bits EDN1, EDN0 of the voltage comparator mode register (VCMPM) is detected on the NMI pin, or the comparator output, depending on the NSOCE bit of voltage comparator mode register (VCMPM), the interrupt occurs.
The watchdog timer interrupt request (NMIWDT) is only effective as non-maskable interrupt if the WDTM 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 > NMIVC

Note that if an NMIVC or NMIWDT request is generated while NMIVC is being serviced, the service is executed as follows.

(1) If an NMIVC is generated while NMIVC is being serviced

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

(2) If an NMIWDT request is generated while NMIVC is being serviced

If the PSW.NP bit remains set (1) while NMIVC is being serviced, the new NMIWDT request is held pending. The pending NMIWDT request is acknowledge after servicing of the current NMIVC request has finished (after execution of the RETI instruction).
If the PSW.NP bit is cleared (0) while NMIVC is being serviced, the newly generated NMIWDT request is executed (NMIVC servicing is halted).

Remark: PSW.NP: The NP bit of the PSW register.

Cautions: 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 NMIVC 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 NMIVC interrupt afterwards cannot be acknowledged correctly.

7.2.1 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 (00000010H) corresponding to the non-maskable interrupt to the PC, and transfers control.

The processing configuration of a non-maskable interrupt is shown in Figure 7-1.

Figure 7-1: Processing Configuration of Non-Maskable Interrupt

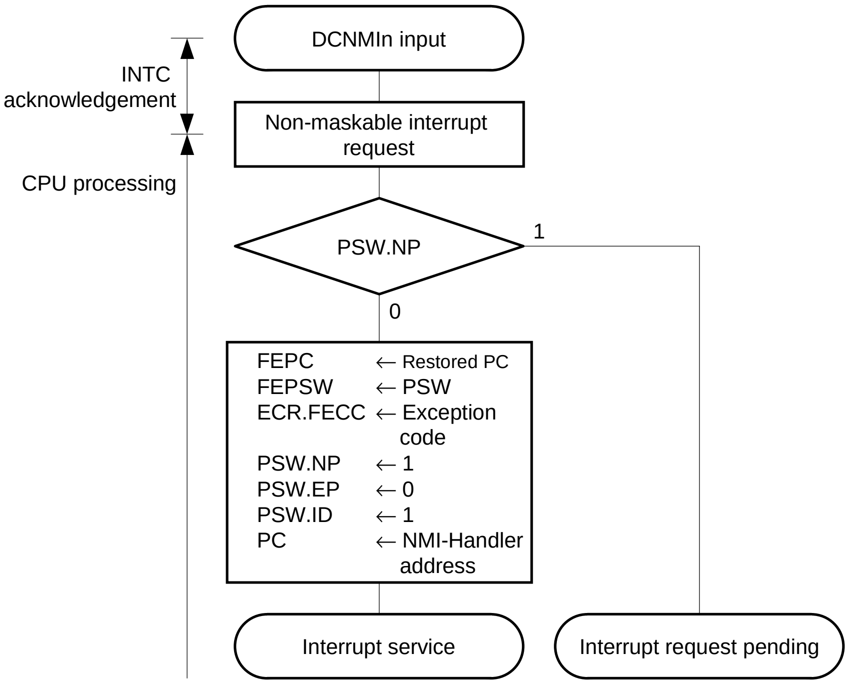
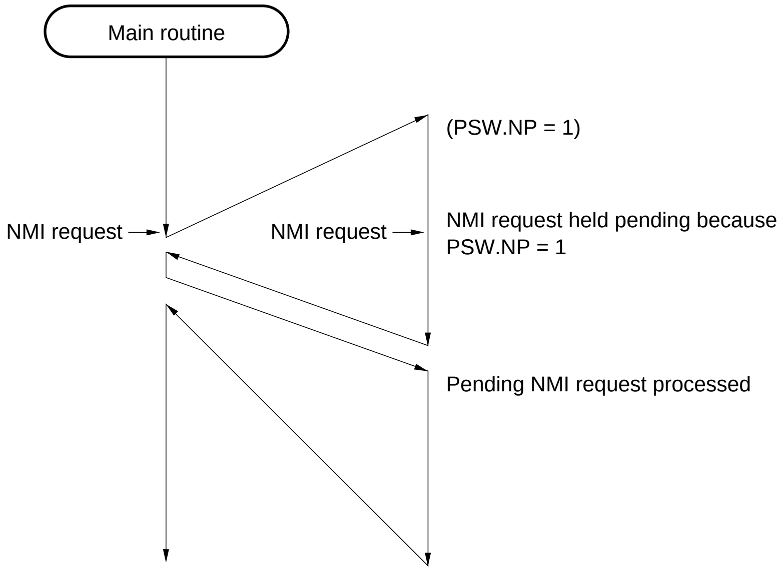


Figure 7-2: Acknowledging Non-Maskable Interrupt Request

(a) If a new NMIVC request is generated while an NMIVC service program is being executed


Main routine
(PSW.NP = 1)
NMI request
NMI request
NMI request held pending because
PSW.NP = 1
Pending NMI request processed

(b) If a new NMIVC request is generated twice while an NMIVC service program is being executed

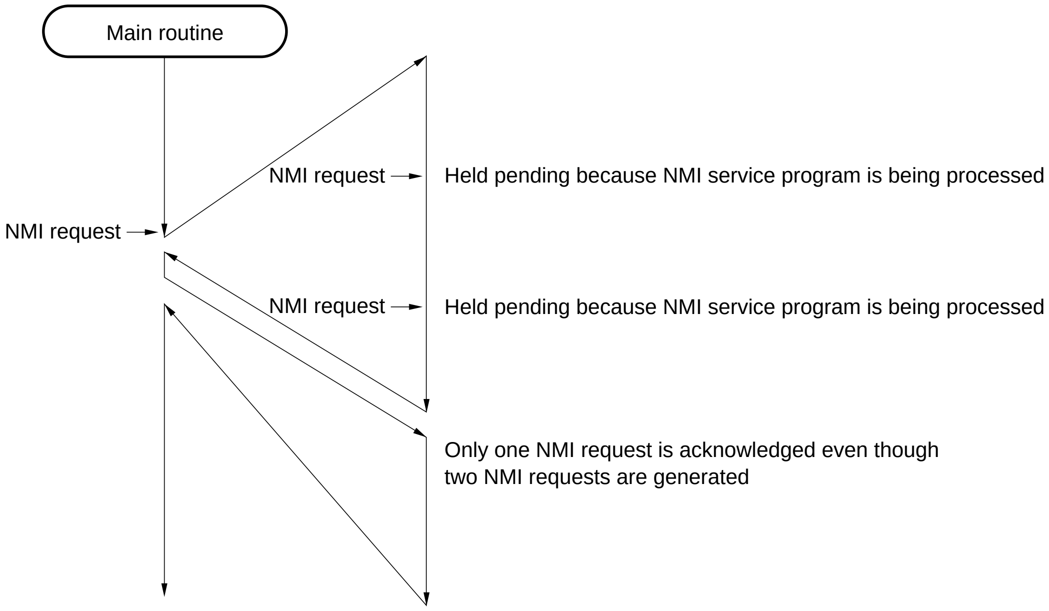

Main routine
NMI request
Held pending because NMI service program is being processed
NMI request
NMI request
Held pending because NMI service program is being processed
Only one NMI request is acknowledged even though
two NMI requests are generated

Figure 7-3: Example of Non-Maskable Interrupt Request Acknowledgement Operation (1/2)

(c) Multiple NMI requests generated at the same time

NMIVC and NMIWD requests generated simultaneously

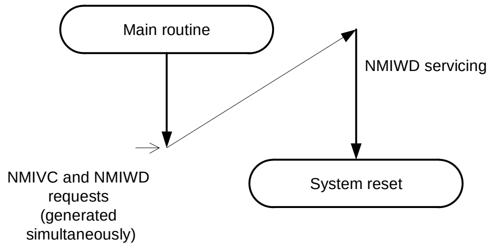
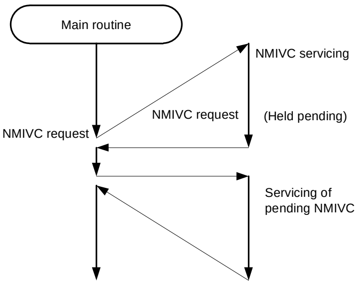
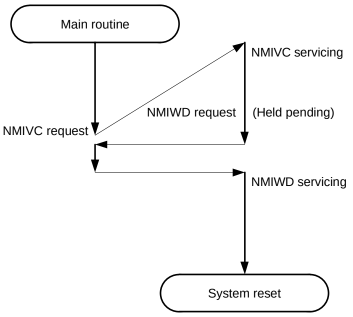
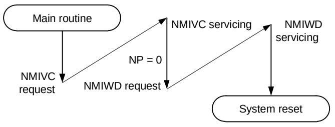
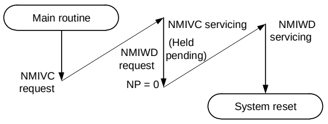
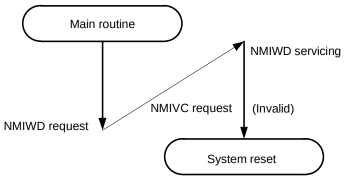
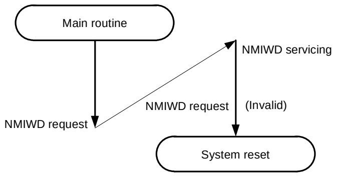


Figure 7-3: Example of Non-Maskable Interrupt Request Acknowledgement Operation (2/2)

(d) NMI request generated during NMI servicing

NMI being serviced	NMI request generated during NMI servicing	
	NMIVC	NMIWD
NMIVC	• NMIVC request generated during NMIVC servicing  Main routine NMIVC servicing NMIVC request (Held pending) NMIVC request Servicing of pending NMIVC	• NMIWD request generated during NMIVC servicing (NP = 1 retained before NMIWD request)  Main routine NMIVC servicing NMIWD request (Held pending) NMIVC request NMIWD servicing System reset • NMIWD request generated during NMIVC servicing (NP = 0 set before NMIWD request)  Main routine NMIVC servicing NMIWD servicing NP = 0 NMIVC request NMIWD request System reset • NMIWD request generated during NMIVC servicing (NP = 0 set after NMIWD request)  Main routine NMIVC servicing NMIWD servicing (Held pending) NMIWD request NMIVC request NP = 0 System reset
NMIWD	• NMIVC request generated during NMIWD servicing  Main routine NMIWD servicing NMIVC request (Invalid) NMIWD request System reset	• NMIWD request generated during NMIWD servicing  Main routine NMIWD servicing NMIWD request (Invalid) NMIWD request System reset

7.2.2 Restore

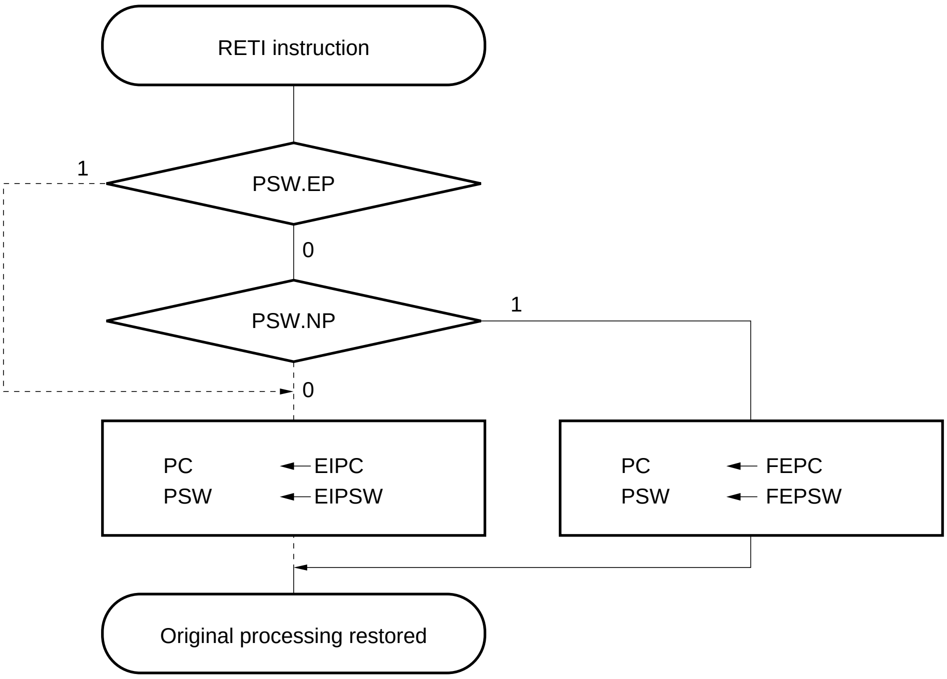
(1) NMIVC

Execution is restored from the non-maskable interrupt (NMIVC) 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 7-4 illustrates how the RETI instruction is processed.

Figure 7-4: 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.

Remark: The solid line indicates the CPU processing flow.

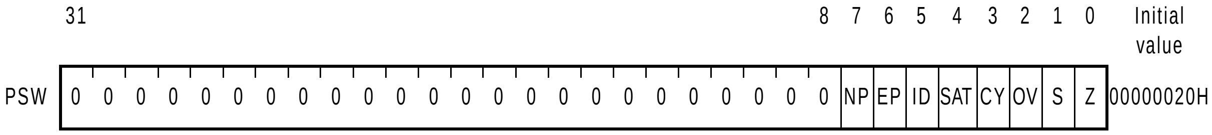
(2) NMIWDT

Restoring by RETI instruction is not possible. Perform a system reset after interrupt servicing.

7.2.3 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.

Figure 7-5: Non-maskable Interrupt Status Flag (NP)



	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

7.2.4 Edge detection function

The behaviour of the non-maskable interrupt (NMIVC) can be specified by the voltage comparator mode register. The valid edge of the external NMI pin input can be specified by the EDN1, EDN0 bits, whereas the source has to be selected by the NSOCE bit.
The register can be read/written in 8-bit or 1-bit units.

Figure 7-6: Voltage Comparator Mode Register (VCMPM)

.

	7	6	5	4	3	2	1	0	Address	R/W	Initial value
VCMPM	VCEN	NSOCE	EFBK	0	EDN1	EDN0	EDM1	EDM0	FFFFF860H	R/W	00H

<table>
<tr><th>Bit position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>7</td><td>VCEN</td><td>Enables the voltage comparator.</td></tr>
<tr><td>6</td><td>NSOCE</td><td>Specifies the NMI source.
0: NMI pin
1: Comparator output</td></tr>
<tr><td>5</td><td>EFBK</td><td>Enables comparator feedback.</td></tr>
<tr><td>3, 2</td><td>EDN1, EDN0</td><td>Specifies the NMI pin’s valid edge.
<table>
<tr><th>EDN1</th><th>EDN0</th><th>Operation</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Falling edge</td></tr>
<tr><td>1</td><td>1</td><td>Both falling and rising edges</td></tr>
</table></td></tr>
<tr><td>1, 0</td><td>EDM1, EDM0</td><td>Specifies the valid edge for maskable comparator interrupt.
<table>
<tr><th>EDM1</th><th>EDM0</th><th>Operation</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Falling edge</td></tr>
<tr><td>1</td><td>1</td><td>Both falling and rising edges</td></tr>
</table></td></tr>
</table>

7.3 Maskable Interrupts

Maskable interrupt requests can be masked by interrupt control registers. The V850E/CA1 / ATOMIC has 60 maskable interrupt sources.
If two or more maskable interrupt requests are generated at the same time, they are acknowledged according to the default priority. In addition to the default priority, eight levels of priorities can be specified by using the interrupt control registers (programmable priority control).
When an interrupt request has been acknowledged, the acknowledgement of other maskable interrupt requests is disabled and the interrupt disabled (DI) status is set.
When the EI instruction is executed in an interrupt processing routine, the interrupt enabled (EI) status is set, which enables servicing of interrupts having a higher priority than the interrupt request in progress (specified by the interrupt control register). Note that only interrupts with a higher priority will have this capability; interrupts with the same priority level cannot be nested.
However, if multiple interrupts are executed, the following processing is necessary.

(1) Save EIPC and EIPSW in memory or a general-purpose register before executing the EI instruction.
(2) Execute the DI instruction before executing the RETI instruction, then reset EIPC and EIPSW with the values saved in (1).

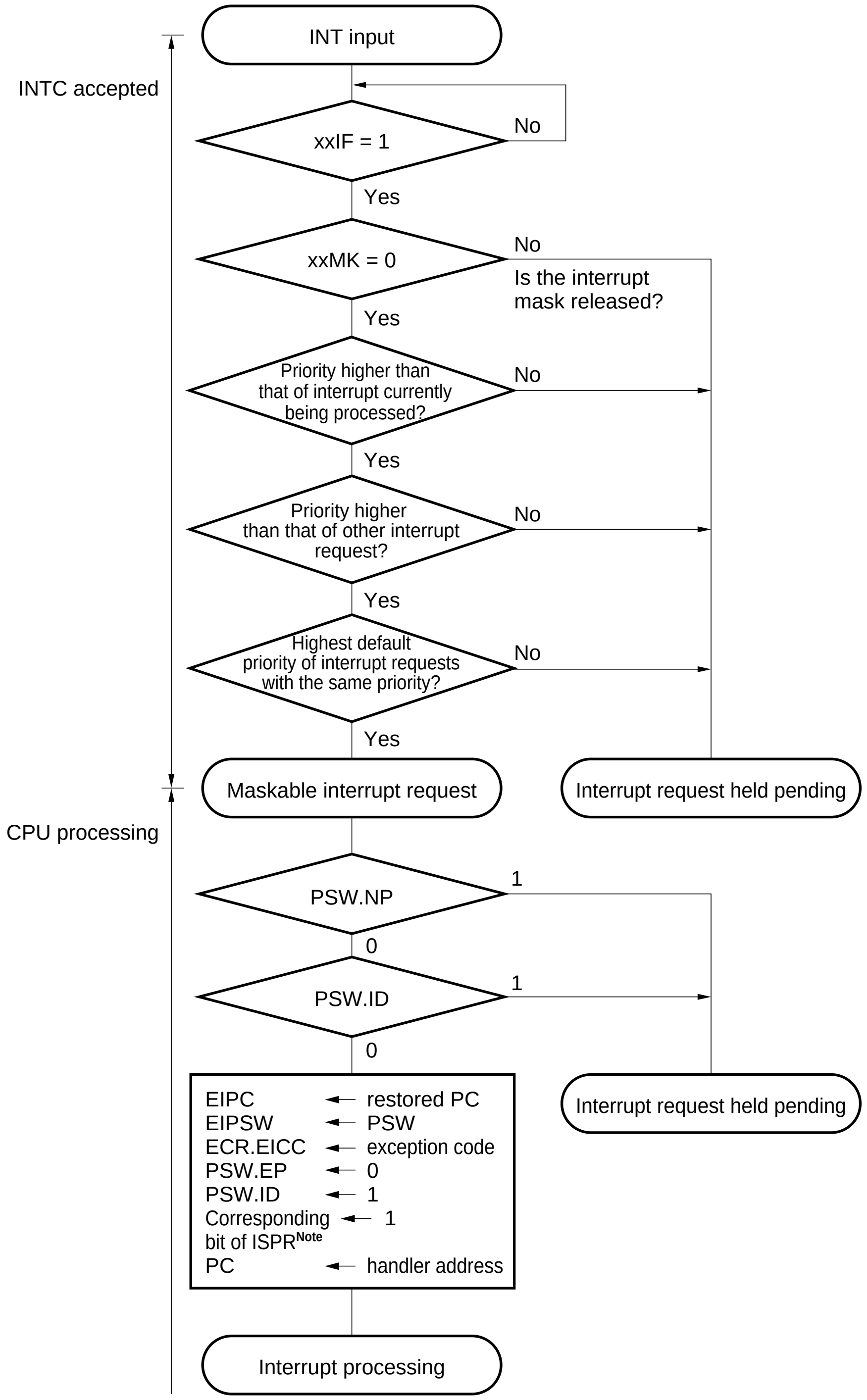
7.3.1 Operation

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 7-7.

Figure 7-7: Maskable Interrupt Processing


INT input
INTC accepted
xxIF = 1
No
Yes
xxMK = 0
No
Is the interrupt mask released?
Yes
Priority higher than that of interrupt currently being processed?
No
Yes
Priority higher than that of other interrupt request?
No
Yes
Highest default priority of interrupt requests with the same priority?
No
Yes
Maskable interrupt request
Interrupt request held pending
CPU processing
PSW.NP
1
0
PSW.ID
1
0
EIPC ← restored PC
EIPSW ← PSW
ECR.EICC ← exception code
PSW.EP ← 0
PSW.ID ← 1
Corresponding bit of ISPR^Note ← 1
PC ← handler address
Interrupt request held pending
Interrupt processing

Note: For the ISPR register, see **7.3.6 In-service priority register (ISPR)**.

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.

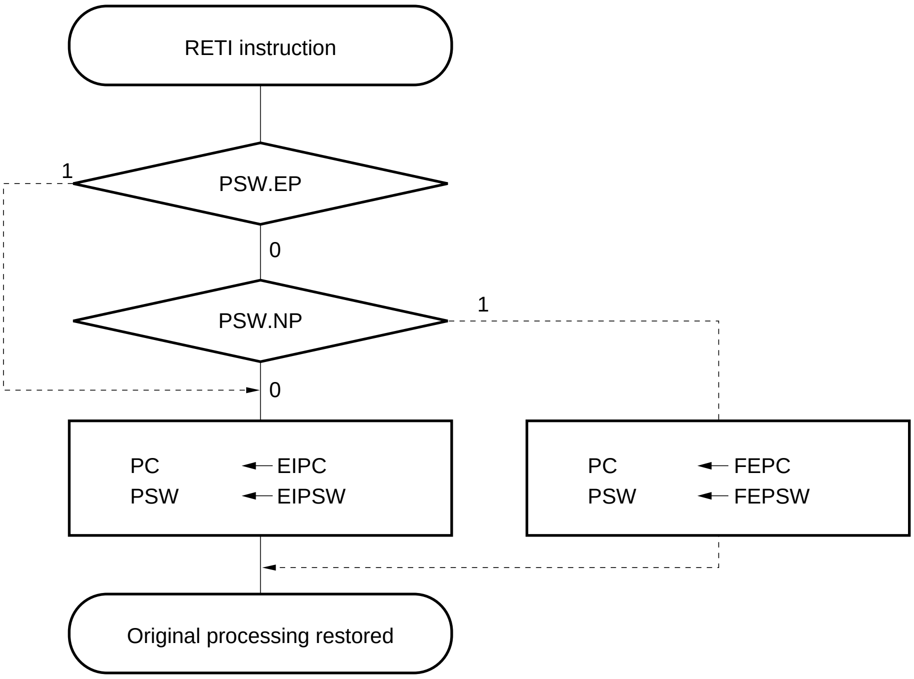
7.3.2 Restore

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 7-8 illustrates the processing of the RETI instruction.

Figure 7-8: RETI Instruction Processing



Note: For the ISPR register, see **7.3.6 In-service priority register (ISPR)**.

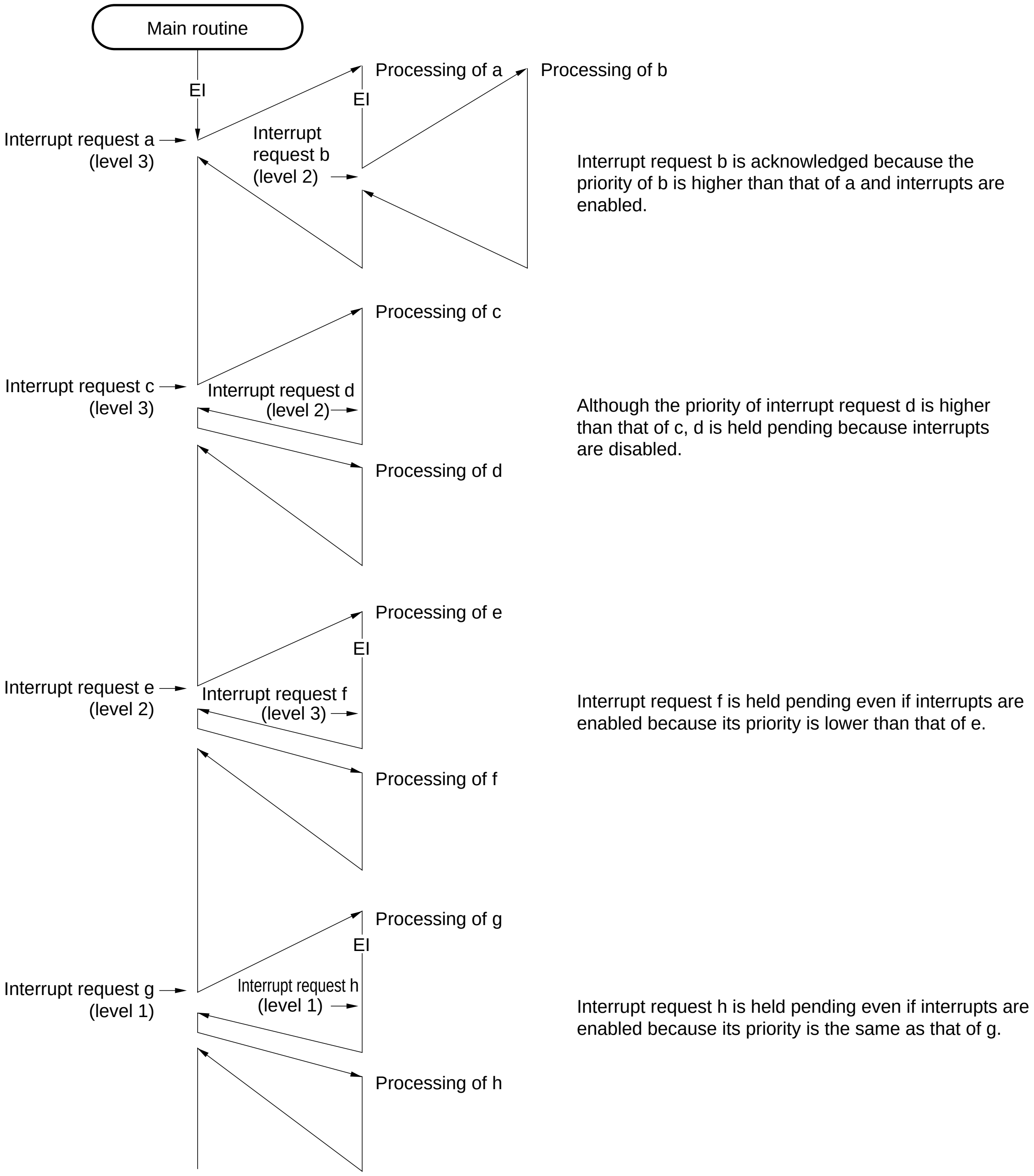
Caution: When the PSW.EP bit and the PSW.NP bit are changed by the LDSR instruction during maskable interrupt processing, in order to restore the PC and PSW correctly during recovery by the RETI instruction, it is necessary to set PSW.EP back to 0 and PSW.NP back to 0 using the LDSR instruction immediately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

Remark: The solid lines show the CPU processing flow.

7.3.3 Priorities of maskable interrupts

The V850E/CA1 / ATOMIC 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 **Table 7-1: Interrupt/Exception Source List (Sheet 1 of 3)**. The programmable priority control customizes interrupt requests into eight levels by setting the priority level specification flag.
Note that when an interrupt request is acknowledged, the ID flag of PSW is automatically set to 1. Therefore, when multiple interrupts are to be used, clear the ID flag to 0 beforehand (for example, by placing the EI instruction in the interrupt service program) to set the interrupt enable mode.

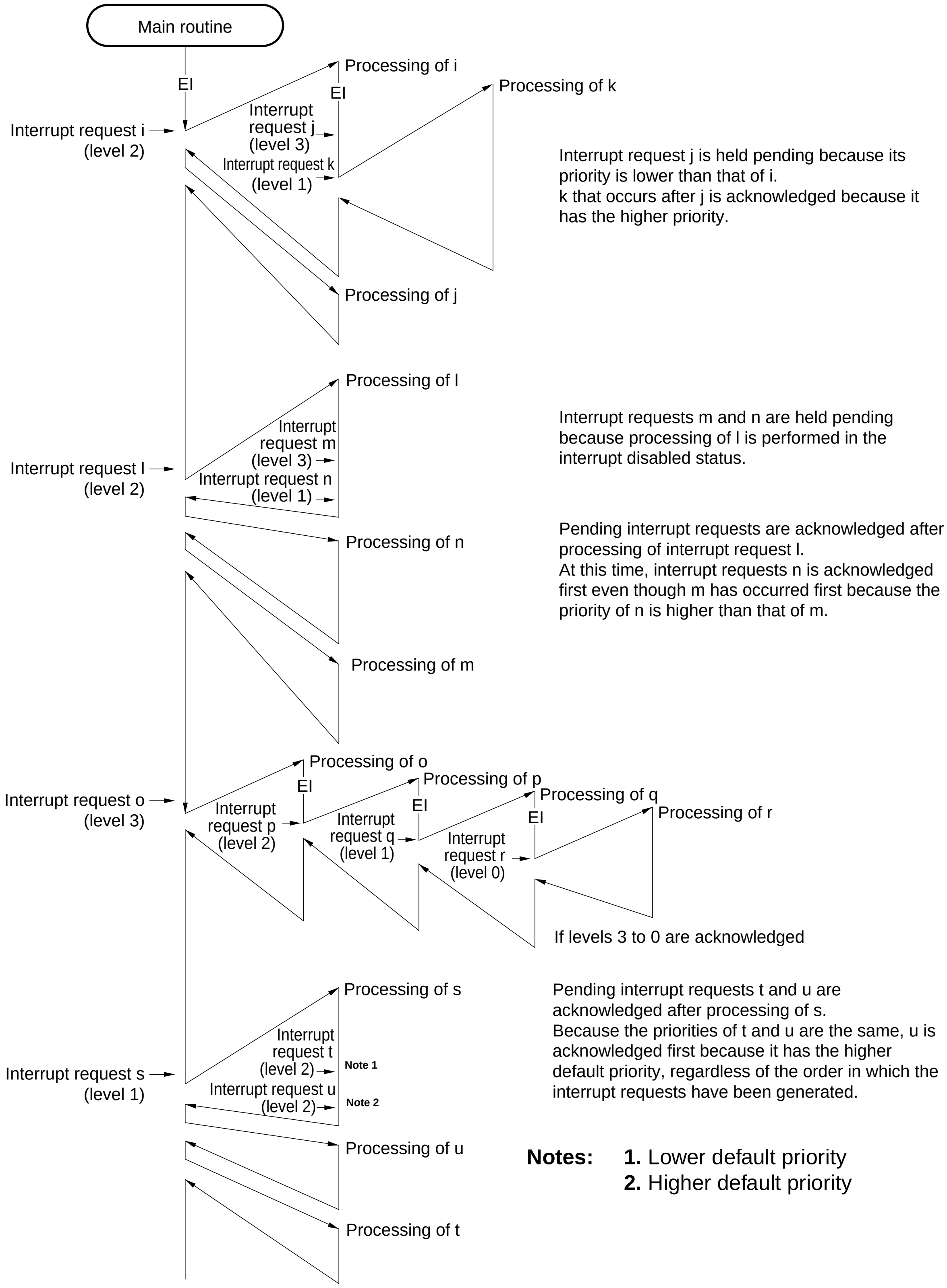
Figure 7-9: Example of Processing in Which Another Interrupt Request Is Issued While an Interrupt Is Being Processed (1/2)


Main routine
EI
Interrupt request a (level 3)
Processing of a
EI
Interrupt request b (level 2)
Processing of b
Interrupt request b is acknowledged because the priority of b is higher than that of a and interrupts are enabled.
Processing of c
Interrupt request c (level 3)
Interrupt request d (level 2)
Although the priority of interrupt request d is higher than that of c, d is held pending because interrupts are disabled.
Processing of d
Processing of e
EI
Interrupt request e (level 2)
Interrupt request f (level 3)
Interrupt request f is held pending even if interrupts are enabled because its priority is lower than that of e.
Processing of f
Processing of g
EI
Interrupt request g (level 1)
Interrupt request h (level 1)
Interrupt request h is held pending even if interrupts are enabled because its priority is the same as that of g.
Processing of h

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.

Remarks:
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 7-9: Example of Processing in Which Another Interrupt Request Is Issued While an Interrupt Is Being Processed (2/2)



Interrupt request j is held pending because its priority is lower than that of i.
k that occurs after j is acknowledged because it has the higher priority.

Interrupt requests m and n are held pending because processing of l is performed in the interrupt disabled status.

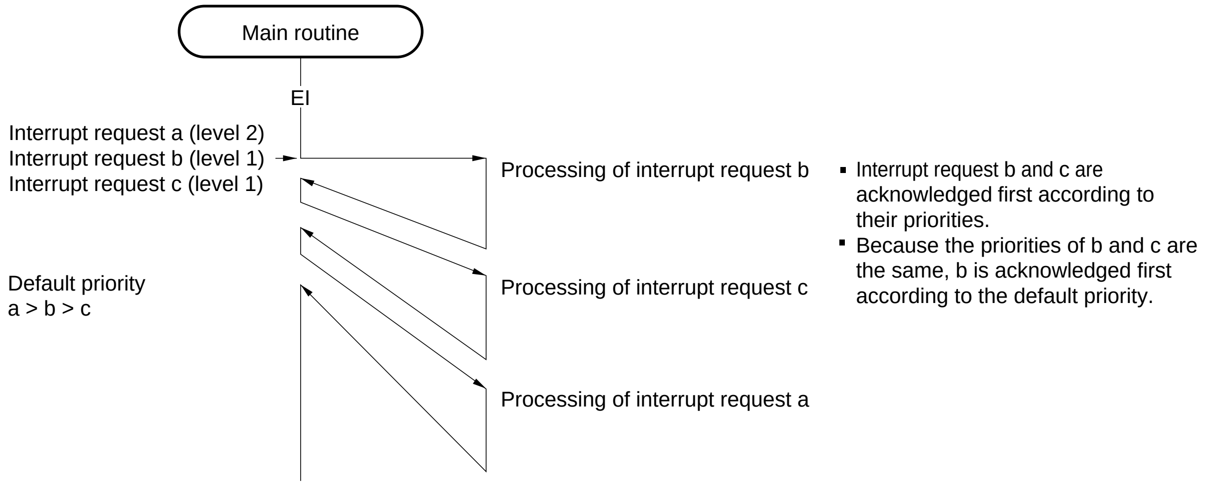
Pending interrupt requests are acknowledged after processing of interrupt request l.
At this time, interrupt requests n is acknowledged first even though m has occurred first because the priority of n is higher than that of m.

Pending interrupt requests t and u are acknowledged after processing of s.
Because the priorities of t and u are the same, u is acknowledged first because it has the higher default priority, regardless of the order in which the interrupt requests have been generated.

Notes: **1.** Lower default priority
2. Higher default priority

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.

Figure 7-10: Example of Processing Interrupt Requests Simultaneously Generated


Main routine
EI
Interrupt request a (level 2)
Interrupt request b (level 1)
Interrupt request c (level 1)
Default priority
a > b > c
Processing of interrupt request b
Processing of interrupt request c
Processing of interrupt request a

- Interrupt request b and c are acknowledged first according to their priorities.
- Because the priorities of b and c are the same, b is acknowledged first according to the default priority.

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.

7.3.4 Interrupt control register (PICn)

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.

Figure 7-11: Interrupt Control Register (PICn)

	7	6	5	4	3	2	1	0	Address	Initial value
PICn	PIFn	PMKn	0	0	0	PPRn2	PPRn1	PPRn0	FFFFF110H to FFFF18EH	47H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>7</td><td>PIFn</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>PMKn</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>PPRn2 to PPRn0</td><td>8 levels of priority order are specified for each interrupt.
<table>
<tr><th>PPRn2</th><th>PPRn1</th><th>PPRn0</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>

Remark: n = 0 to 59

The address and bit of each interrupt control register are shown in the following Table 7-2.

Table 7-2: Addresses and Bits of Interrupt Control Registers (Sheet 1 of 2)

Address	Register	Bit							
		7	6	5	4	3	2	1	0
FFFFF110H	PIC0	PIF0	PMK0	0	0	0	PPR02	PPR01	PPR00
FFFFF112H	PIC1	PIF1	PMK1	0	0	0	PPR12	PPR11	PPR10
FFFFF114H	PIC2	PIF2	PMK2	0	0	0	PPR22	PPR21	PPR20
FFFFF116H	PIC3	P0IF3	PMK3	0	0	0	PPR32	P0PR31	PPR30
FFFFF118H	PIC4	P1IF4	PMK4	0	0	0	PPR42	P1PR41	PPR40
FFFFF11AH	PIC5	PIF5	PMK5	0	0	0	PPR52	PPR51	PPR50
FFFFF11CH	PIC6	PIF6	PMK6	0	0	0	PPR62	PPR61	PPR60
FFFFF11EH	PIC7	PIF7	PMK7	0	0	0	PPR72	PPR71	PPR70
FFFFF120H	PIC8	PIF8	PMK8	0	0	0	PPR82	PPR81	PPR80
FFFFF122H	PIC9	PIF9	PMK9	0	0	0	PPR92	PPR91	PPR90
FFFFF124H	PIC10	PIF10	PMK10	0	0	0	PPR102	PPR101	PPR100
FFFFF126H	PIC11	PIF11	PMK11	0	0	0	PPR112	PPR111	PPR110
FFFFF128H	PIC12	PIF12	PMK12	0	0	0	PPR122	PPR121	PPR120
FFFFF12AH	PIC13	PIF13	PMK13	0	0	0	PPR132	PPR131	PPR130
FFFFF12CH	PIC14	PIF14	PMK14	0	0	0	PPR142	PPR141	PPR140
FFFFF12EH	PIC15	PIF15	PMK15	0	0	0	PPR152	PPR151	PPR150
FFFFF130H	PIC16	PIF16	PMK16	0	0	0	PPR162	PPR161	PPR160
FFFFF132H	PIC17	PIF17	PMK17	0	0	0	PPR172	PPR171	PPR170
FFFFF134H	PIC18	PIF18	PMK18	0	0	0	PPR182	PPR181	PPR180
FFFFF136H	PIC19	PIF19	PMK19	0	0	0	PPR192	PPR191	PPR190
FFFFF138H	PIC20	PIF20	PMK20	0	0	0	PPR202	PPR201	PPR100
FFFFF13AH	PIC21	PIF21	PMK21	0	0	0	PPR212	PPR11	PPR110
FFFFF13CH	PIC22	PIF22	PMK22	0	0	0	PPR222	PPR221	PPR220
FFFFF13EH	PIC23	PIF23	PMK23	0	0	0	PPR232	PPR231	PPR230
FFFFF140H	PIC24	PIF24	PMK24	0	0	0	PPR242	PPR241	PPR240
FFFFF142H	PIC25	PIF25	PMK25	0	0	0	PPR252	PPR251	PPR250
FFFFF144H	PIC26	PIF26	PMK26	0	0	0	PPR262	PPR261	PPR260
FFFFF146H	PIC27	PIF27	PMK27	0	0	0	PPR272	PPR271	PPR270
FFFFF148H	PIC28	PIF28	PMK28	0	0	0	PPR282	PPR281	PPR280
FFFFF14AH	PIC29	PIF29	PMK29	0	0	0	PPR292	PPR291	PPR290
FFFFF14CH	PIC30	PIF30	PMK30	0	0	0	PPR302	PPR301	PPR300
FFFFF14EH	PIC31	PIF31	PMK31	0	0	0	PPR312	PPR311	PPR310
FFFFF150H	PIC32	PIF32	PMK32	0	0	0	PPR322	PPR321	PPR320
FFFFF152H	PIC33	PIF33	PMK33	0	0	0	PPR332	PPR331	PPR330
FFFFF154H	PIC34	PIF34	PMK34	0	0	0	PPR342	PPR341	PPR340
FFFFF156H	PIC35	PIF35	PMK35	0	0	0	PPR352	PPR351	PPR350
FFFFF158H	PIC36	PIF36	PMK36	0	0	0	PPR362	PPR361	PPR360
FFFFF15AH	PIC37	PIF37	PMK37	0	0	0	PPR372	PPR371	PPR370
FFFFF15CH	PIC38	PIF38	PMK38	0	0	0	PPR382	PPR381	PPR380
FFFFF15EH	PIC39	PIF39	PMK39	0	0	0	PPR392	PPR391	PPR390
FFFFF160H	PIC40	PIF40	PMK40	0	0	0	PPR402	PPR401	PPR400
FFFFF162H	PIC41	PIF41	PMK41	0	0	0	PPR412	PPR411	PPR410
FFFFF164H	PIC42	PIF42	PMK42	0	0	0	PPR422	PPR421	PPR420

Table 7-2: Addresses and Bits of Interrupt Control Registers (Sheet 2 of 2)

Address	Register	Bit							
		7	6	5	4	3	2	1	0
FFFFF166H	PIC43	PIF43	PMK43	0	0	0	PPR432	PPR431	PPR430
FFFFF168H	PIC44	PIF44	PMK44	0	0	0	PPR442	PPR441	PPR440
FFFFF16AH	PIC45	PIF45	PMK45	0	0	0	PPR452	PPR451	PPR450
FFFFF16CH	PIC46	PIF46	PMK46	0	0	0	PPR462	PPR461	PPR460
FFFFF16EH	PIC47	PIF47	PMK47	0	0	0	PPR472	PPR471	PPR470
FFFFF170H	PIC48	PIF48	PMK48	0	0	0	PPR482	PPR481	PPR480
FFFFF172H	PIC49	PIF49	PMK49	0	0	0	PPR492	PPR491	PPR490
FFFFF174H	PIC50	PIF50	PMK50	0	0	0	PPR502	PPR501	PPR500
FFFFF176H	PIC51	PIF51	PMK51	0	0	0	PPR512	PPR511	PPR510
FFFFF178H	PIC52	PIF52	PMK52	0	0	0	PPR522	PPR521	PPR520
FFFFF17AH	PIC53	PIF53	PMK53	0	0	0	PPR532	PPR531	PPR530
FFFFF17CH	PIC54	PIF54	PMK54	0	0	0	PPR542	PPR541	PPR540
FFFFF17EH	PIC55	PIF55	PMK55	0	0	0	PPR552	PPR551	PPR550
FFFFF180H	PIC56	PIF56	PMK56	0	0	0	PPR562	PPR561	PPR560
FFFFF182H	PIC57	PIF57	PMK57	0	0	0	PPR572	PPR571	PPR570
FFFFF184H	PIC58	PIF58	PMK58	0	0	0	PPR582	PPR581	PPR580
FFFFF186H	PIC59	PIF59	PMK59	0	0	0	PPR592	PPR591	PPR590

Remark: For the interrupt source to the respective controlling registers PICn (n=0 to 59) refer to Table 7-1, “Interrupt/Exception Source List (Sheet 1 of 3),” on page 184.

7.3.5 Interrupt mask registers 0 to 3 (IMR0 to IMR3)

These registers set the interrupt mask state for the maskable interrupts.
The PMKn bit of the IMR0 to IMR3 registers is equivalent to the PMKn bit of the PICn register.
IMRm registers can be read/written in 16-bit units (m = 0 to 3).
When the IMRm register is divided into two registers: higher 8 bits (IMRmH register) and lower 8 bits (IMRmL register), these registers can be read/written in 8-bit or 1-bit units (m = 0 to 3).
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).

Figure 7-12: Interrupt Mask Registers 0 to 3 (IMR0 to IMR3)

	15	14	13	12	11	10	9	8	Address	Initial value
IMR0	PMK15	PMK14	PMK13	PMK12	PMK11	PMK10	PMK9	PMK8	FFFFF100H	FFFFH
	7	6	5	4	3	2	1	0		
	PMK7	PMK6	PMK5	PMK4	PMK3	PMK2	PMK1	PMK0		

	15	14	13	12	11	10	9	8	Address	Initial value
IMR1	PMK31	PMK30	PMK29	PMK28	PMK27	PMK26	PMK25	PMK24	FFFFF102H	FFFFH
	7	6	5	4	3	2	1	0		
	PMK23	PMK22	PMK21	PMK20	PMK19	PMK18	PMK17	PMK16		

	15	14	13	12	11	10	9	8	Address	Initial value
IMR2	PMK47	PMK46	PMK45	PMK44	PMK43	PMK42	PMK41	PMK40	FFFFF104H	FFFFH
	7	6	5	4	3	2	1	0		
	PMK39	PMK38	PMK37	PMK36	PMK35	PMK34	PMK33	PMK32		

	15	14	13	12	11	10	9	8	Address	Initial value
IMR3	1	1	1	1	PMK59	PMK58	PMK57	PMK56	FFFFF106H	FFFFH
	7	6	5	4	3	2	1	0		
	PMK55	PMK54	PMK53	PMK52	PMK51	PMK50	PMK49	PMK48		

Bit Position	Bit Name	Function
15 to 0	PMKn	Interrupt mask flag. 0: Interrupt servicing enabled 1: Interrupt servicing disabled (pending)

Remark: n: peripheral unit number (refer to Table 7-2, “Addresses and Bits of Interrupt Control Registers (Sheet 1 of 2),” on page 203).

7.3.6 In-service priority register (ISPR)

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.

Figure 7-13: In-Service Priority Register (ISPR)

	7	6	5	4	3	2	1	0	Address	Initial value
ISPR	ISPR7	ISPR6	ISPR5	ISPR4	ISPR3	ISPR2	ISPR1	ISPR0	FFFFF1FAH	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

Remark: n = 0 to 7 (priority level)

7.3.7 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.

Figure 7-14: Maskable Interrupt Status Flag (ID)

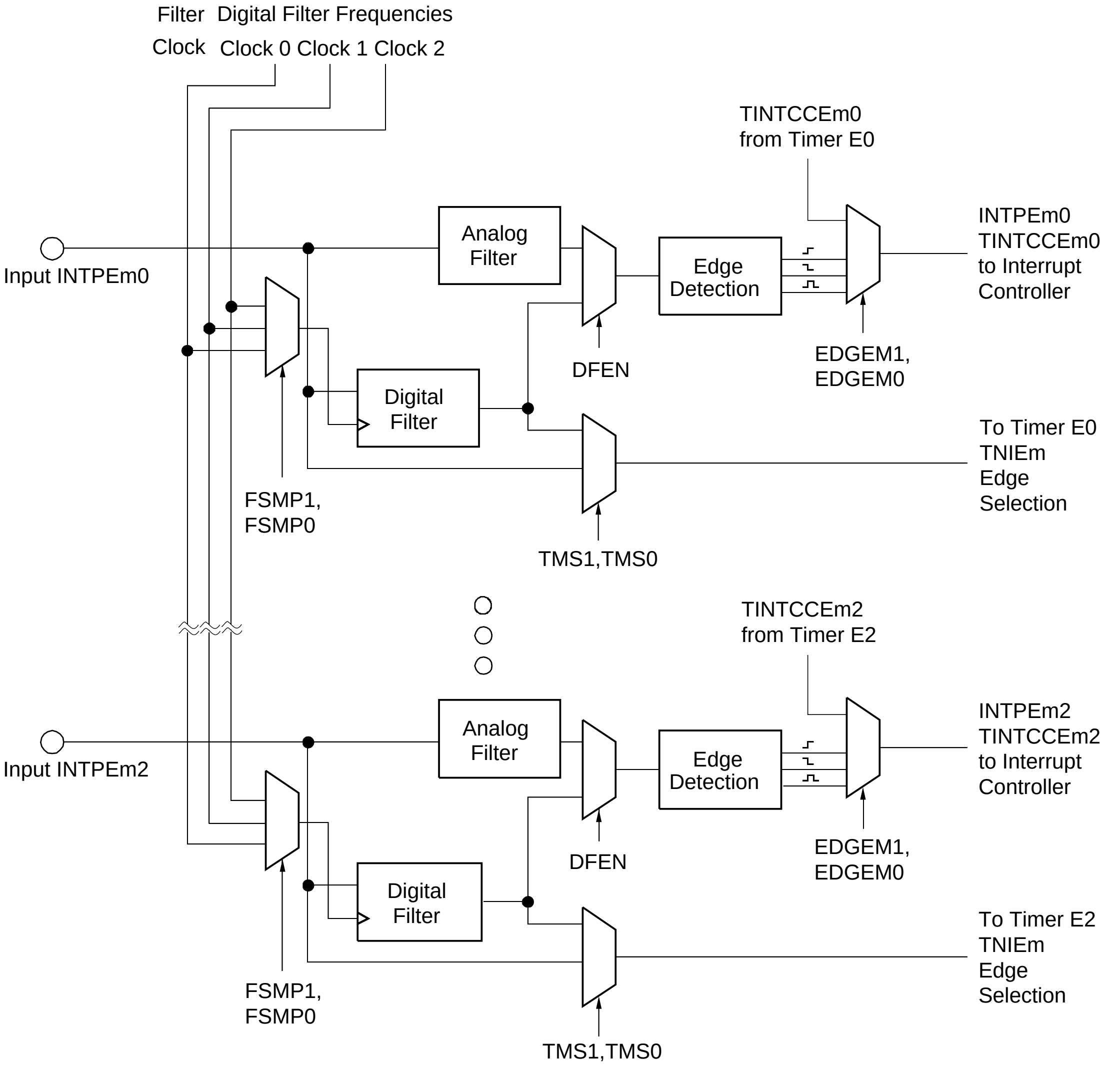
	31																							8	7	6	5	4	3	2	1	0	Initial value
PSW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NP	EP	ID	SAT	CY	OV	S	Z	00000020H

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.

7.4 Noise Elimination Circuit

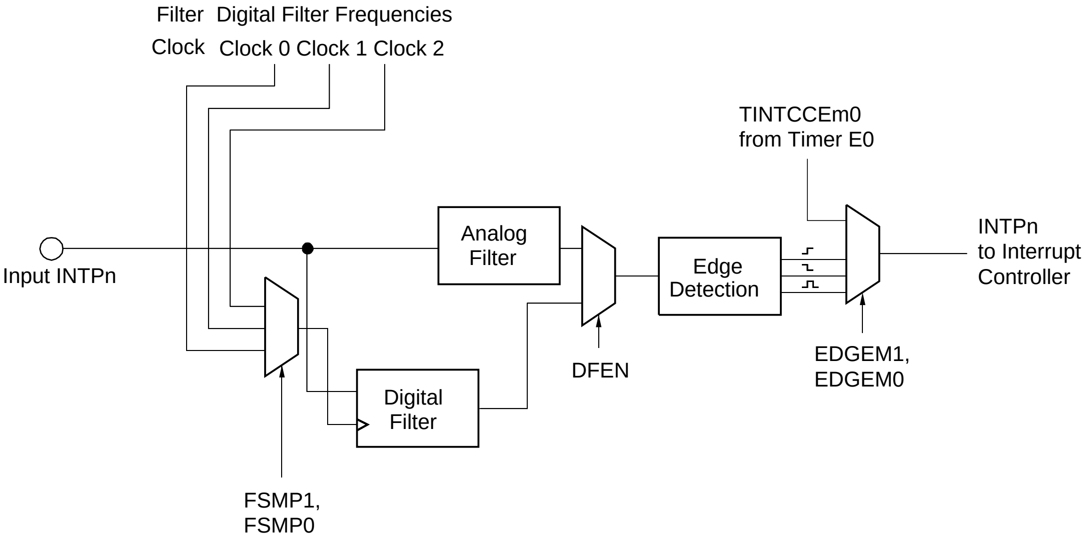
V850E/CA1 / ATOMIC is provided with filter / edge detection circuits for ports 3, 4, 5 and port 6 (3 channels).
The circuit consists from programmable digital filter, analog filter, edge detection and input source selection.

Figure 7-15: Timer E Input Circuit Overview



Remark: m = 0 to 5

Figure 7-16: Port Interrupt Input Circuit Overview



Remark: n = 0 to 2

7.4.1 Analog Filter

The analogue filter consists of a comparator stage, which compares the input pin level against a delayed input pin level. The filter output follows the filter input, if this compare operation matches. The delay stage is set to a **fixed delay** of **50 ns.** The tolerance of this delay is at about **70%**. This defines the filter frequency in a range from **12 MHz** to **67 MHz**.
An edge detection circuitry can detect rising, falling or both edges (selectable).

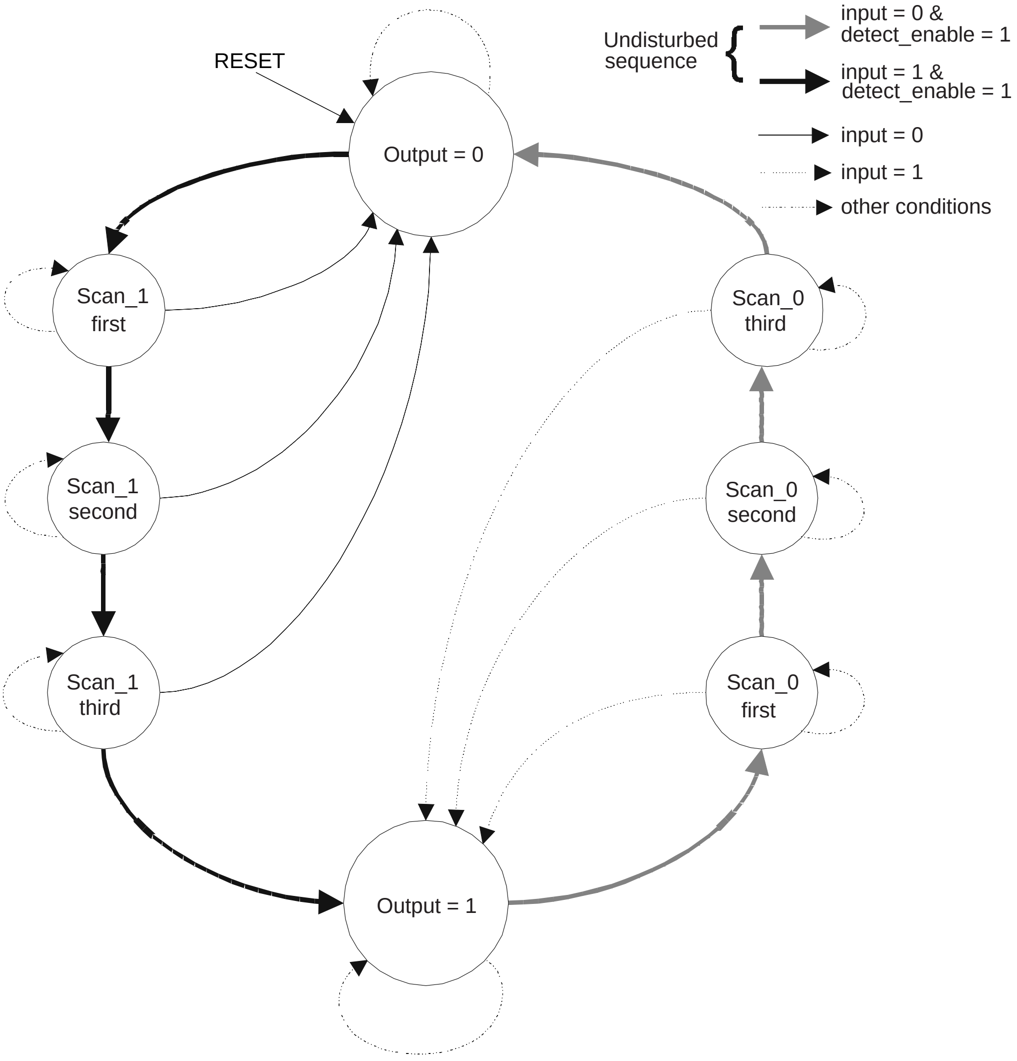
7.4.2 Digital Filter

Behavioral Description

The digital filter simply samples the input with the ***negative*** clock edge of the f_{CPU} internal system clock. The negative clock edge is used to suppress the sampling of glitches caused by the V850E/CA1 / ATOMIC hardware and its external circuitry itself (assuming that all outputs from the V850E/CA1 / ATOMIC change their values only on positive edges of the f_{CPU} clock or any derived sub-clock).
The digital filter's behaviour is described as follows:
The digital filter samples the input signal with its f_{CPU} operating frequency (negative clock edge). To recognize a new value for its output, that differs from the current output state, *4* subsequent samples are required, where each sample has read the same new input value.
However, to *accept* an input sample to be relevant for the new value, the level of the detection enable signal has to be high. The detection enable signal is a divided clock derived from the system clock, with frequencies: f_{CPU}, $f_{CPU}/2$, $f_{CPU}/4$.
To *reject* an input sample sequence, only one violation of an input sample against the sequence of equal and accepted input samples is sufficient. Here, the detection enable signal is not relevant.

The following figure illustrates the behaviour of the filter's state machine.

Figure 7-17: Digital Filter State Machine Diagram



7.4.3 Interrupt trigger mode selection

The valid edge of the INTP pins can be selected by program. The edge that can be selected as the valid edge is one of the following.

- Rising edge
- Falling edge
- Both the rising and falling edges

7.4.4 Filter Edge Detect Mode Register (FEM0n to FEM5n) for Timer E Input Pins (n=0 to 2)

This registers are 8-bit register that control the filter function for the input interrupt pins INTPEmn and the Timer E input (TINEmn) (m=0 to 5, n=0 to 2). Additional they define the interrupt source and interrupt edge selection to the dedicated interrupt controller.
They can be read or written in 8- or 1-bit units.

Figure 7-18: Timer E Input Pin Filter Edge Detect Mode Registers (FEM0n to FEM5n) (n=0 to 2) (1/2)

	7	6	5	4	3	2	1	0	Address	Initial value
FEM00	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF880H	00H
FEM01	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF890H	00H
FEM02	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF8A0H	00H
FEM10	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF881H	00H
FEM11	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF891H	00H
FEM12	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF8A1H	00H
FEM20	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF882H	00H
FEM21	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF892H	00H
FEM22	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF8A2H	00H
FEM30	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF883H	00H
FEM31	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF893H	00H
FEM32	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF8A3H	00H
FEM40	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF884H	00H
FEM41	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF894H	00H
FEM42	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF8A4H	00H
FEM50	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF885H	00H
FEM51	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF895H	00H
FEM52	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	TMS1	TMS0	FFFFF8A5H	00H

Figure 7-18: Timer E Input Pin Filter Edge Detect Mode Registers (FEM0n to FEM5n) (n=0 to 2) (2/2)

<table>
<tr><th>Bit Name</th><th>Description</th></tr>
<tr><td>DFEN</td><td>Digital filter enable
Selects analog or digital filter for interrupt input.
0: Analog filter
1: Digital filter
Note: Refer to Figure 7-15: “Timer E Input Circuit Overview” on page 207</td></tr>
<tr><td>FSMP1,
FSMP0,</td><td>Filter sampling rate
Selects the sampling clock for the digital filter.
<table>
<tr><th>FSMP1</th><th>FSMP0</th><th>Sampling clock (clock input)</th></tr>
<tr><td>0</td><td>0</td><td>f_{CPU} / 1</td></tr>
<tr><td>0</td><td>1</td><td>f_{CPU} / 2</td></tr>
<tr><td>1</td><td>0</td><td>f_{CPU} / 4</td></tr>
<tr><td>1</td><td>1</td><td>reserved</td></tr>
</table></td></tr>
<tr><td>EDGEM1,
EDGEM0</td><td>Edge selection for INTPEmn to interrupt controller
Selects active edge for interrupt generation.
<table>
<tr><th>EDGEM1</th><th>EDGEM0</th><th>Edge selection</th></tr>
<tr><td>0</td><td>0</td><td>Internal interrupt source direct</td></tr>
<tr><td>0</td><td>1</td><td>Positive edge</td></tr>
<tr><td>1</td><td>0</td><td>Falling edge</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table>
Remark: m = 0 to 5, n = 0 to 2</td></tr>
<tr><td>TMS1,
TMS0</td><td>Input mode for function input of Timer E
Selects filter mode for peripheral function input.
<table>
<tr><th>TMS1</th><th>TMS0</th><th>Input mode</th></tr>
<tr><td>0</td><td>0</td><td>Direct port input</td></tr>
<tr><td>0</td><td>1</td><td>Digital filtered input</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td></tr>
</table></td></tr>
</table>

7.4.5 Filter Edge Detect Mode Register (FEM0n to FEM5n) for INT0, INT1 and INT2 Input Pins

This registers are 8-bit register that control the analog or digital filter function for the INT2 to INT0 inputs and the edge selection to the dedicated interrupt controller.
They can be read or written in 8- or 1-bit units.

Figure 7-19: INT0, INT1 and INT2 Input Pin Filter Edge Detect Mode Registers (FEM03 to FEM23)

	7	6	5	4	3	2	1	0	Address	Initial value
FEM03	DFEN	0	FSMP1	FSMP0	EDGEM1	EDGEM0	0	0	FFFFF8B0H	00H
FEM13	DFEN1	0	FSMP1	FSMP0	EDGEM1	EDGEM0	0	0	FFFFF8B1H	00H
FEM23	DFEN2	0	FSMP1	FSMP0	EDGEM1	EDGEM0	0	0	FFFFF8B2H	00H

<table>
<tr><th>Bit Name</th><th>Description</th></tr>
<tr><td>DFEN</td><td>Digital filter enable
Selects analog or digital filter for interrupt input.
0: Analog filter
1: Digital filter
<b>Note:</b> Refer to Figure 7-16: “Port Interrupt Input Circuit Overview” on page 208</td></tr>
<tr><td>FSMP1,
FSMP0,</td><td>Filter sampling rate
Selects the sampling clock for the digital filter.
<table>
<tr><th>FSMP1</th><th>FSMP0</th><th>Sampling clock (clock input)</th></tr>
<tr><td>0</td><td>0</td><td>f_{CPU} / 1</td></tr>
<tr><td>0</td><td>1</td><td>f_{CPU} / 2</td></tr>
<tr><td>1</td><td>0</td><td>f_{CPU} / 4</td></tr>
<tr><td>1</td><td>1</td><td>reserved</td></tr>
</table></td></tr>
<tr><td>EDGEM1,
EDGEM0</td><td>Edge selection for INTn to interrupt controller
Selects active edge for interrupt generation.
<table>
<tr><th>EDGEM1</th><th>EDGEM0</th><th>Sampling clock (clock input)</th></tr>
<tr><td>0</td><td>0</td><td>Internal interrupt source direct</td></tr>
<tr><td>0</td><td>1</td><td>Positive edge</td></tr>
<tr><td>1</td><td>0</td><td>Falling edge</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table>
<b>Remark:</b> n = 0 to 2</td></tr>
</table>

7.5 Software Exception

A software exception is generated when the CPU executes the TRAP instruction, and can be always acknowledged.

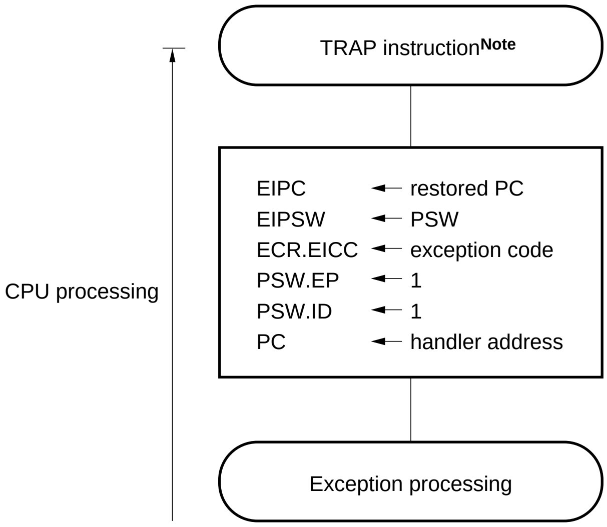
7.5.1 Operation

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 7-20 illustrates the processing of a software exception.

Figure 7-20: 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.

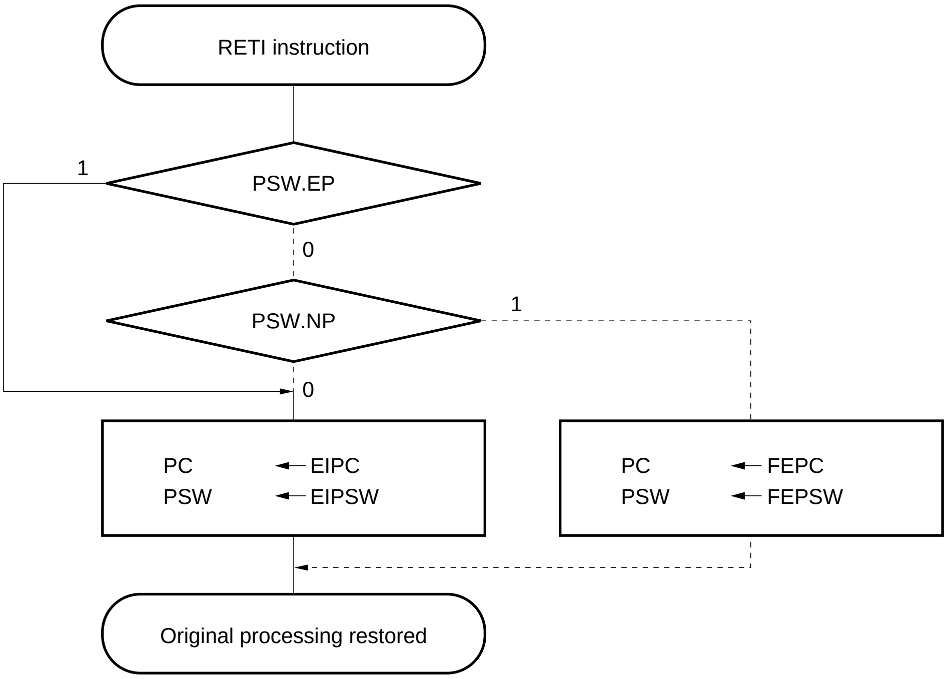
7.5.2 Restore

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 7-21 illustrates the processing of the RETI instruction.

Figure 7-21: RETI Instruction Processing



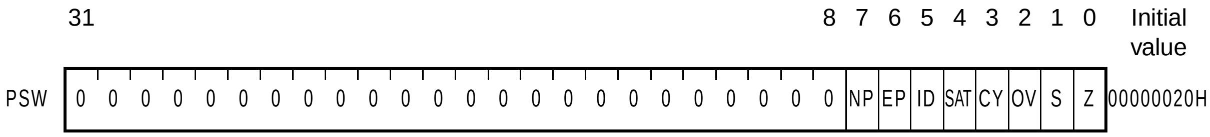
Caution: When the PSW.EP bit and the PSW.NP bit are changed by the LDSR instruction during the software exception processing, in order to restore the PC and PSW correctly during recovery by the RETI instruction, it is necessary to set PSW.EP back to 1 using the LDSR instruction immediately before the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

Remark: The solid lines show the CPU processing flow.

7.5.3 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.

Figure 7-22: Exception Status Flag (EP)



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.

7.6 Exception Trap

An exception trap is an interrupt that is requested when an illegal execution of an instruction takes place. In the V850E/CA1 / ATOMIC, an illegal opcode exception (ILGOP: Illegal Opcode Trap) is considered as an exception trap.

7.6.1 Illegal opcode definition

The illegal instruction has an opcode (bits 10 to 5) of 111111B, a sub-opcode (bits 26 to 23) 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
× × × × ×	1 1 1 1 1 1	× × × × ×	× × × × ×	0 1 1 1 to 1 1 1 1	× × × × × ×	0

Remark: ×: Arbitrary

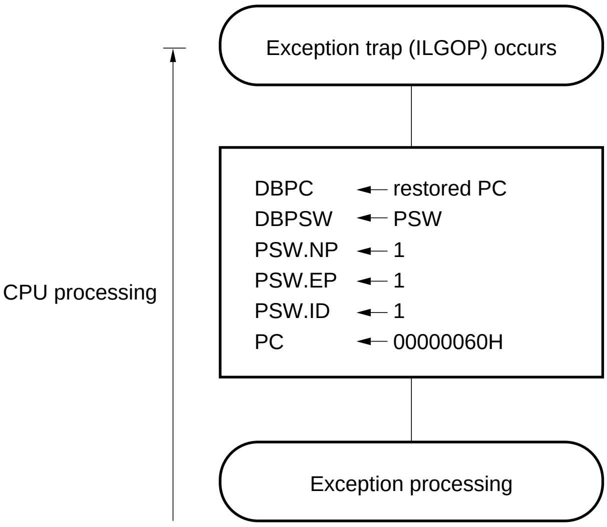
(1) Operation

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 7-23 illustrates the processing of the exception trap.

Figure 7-23: Exception Trap Processing



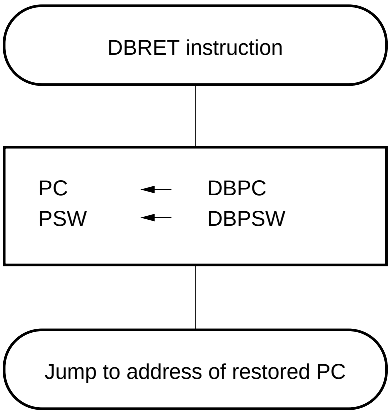
(2) Restore

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 7-24 illustrates the restore processing from an exception trap.

Figure 7-24: Restore Processing from Exception Trap



7.6.2 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.

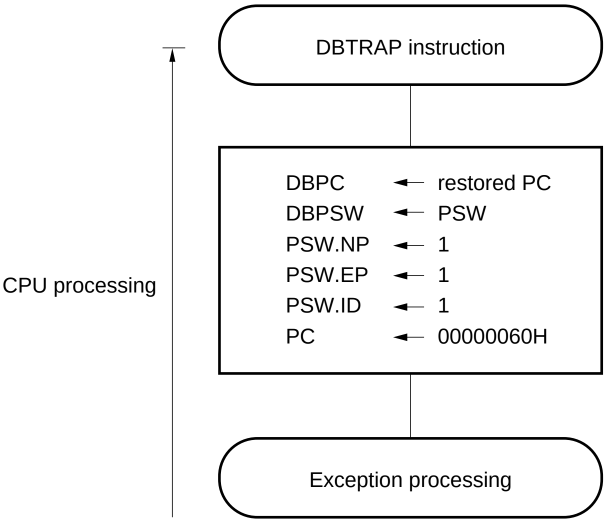
(1) Operation

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 7-25 illustrates the processing of the debug trap.

Figure 7-25: Debug Trap Processing



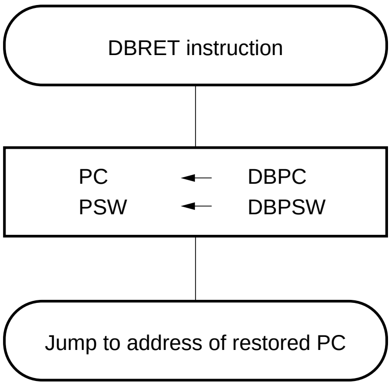
(2) Restore

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 7-26 illustrates the restore processing from a debug trap.

Figure 7-26: Restore Processing from Debug Trap



7.7 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)
...
...                                                      ← 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                      ← Exception such as TRAP instruction acknowledged.
...
• Saved value restored to EIPSW
• Saved value restored to EIPC
• RETI instruction
```

The priority order for multiple interrupt processing control has 8 levels, from 0 to 7 for each maskable interrupt request (0 is the highest priority), but it can be set as desired via software. Setting of the priority order level is done using the PPRn0 to PPRn2 bits of the interrupt control request register (PICn), which is provided for each maskable interrupt request. After system reset, an interrupt request is masked by the PMKn bit and the priority order is set to level 7 by the PPRn0 to PPRn2 bits.

The priority order of maskable interrupts is as follows.

(High) Level 0 > Level 1 > Level 2 > Level 3 > Level 4 > Level 5 > Level 6 > Level 7 (Low)

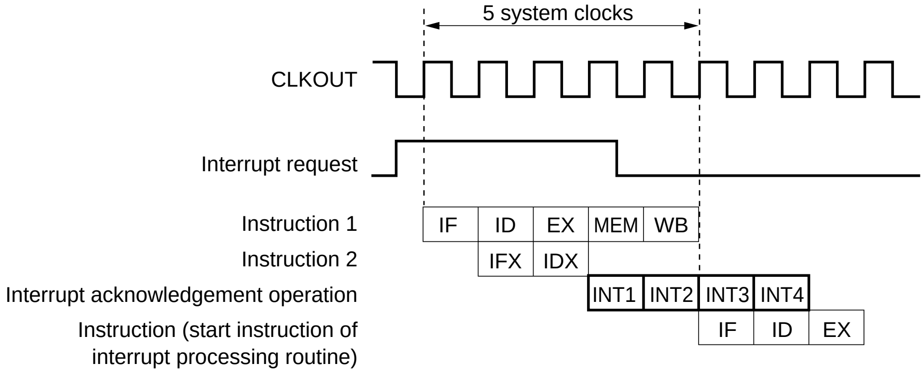
Interrupt processing that has been suspended as a result of multiple processing control is resumed after the processing of the higher priority interrupt has been completed and the RETI instruction has been executed.
A pending interrupt request is acknowledged after the current interrupt processing has been completed and the RETI instruction has been executed.

Caution: In a non-maskable interrupt processing routine (time until the RETI instruction is executed), maskable interrupts are suspended and not acknowledged.

7.8 Interrupt Response Time

The following table describes the V850E/CA1 / ATOMIC interrupt response time (from interrupt generation to start of interrupt processing).

Figure 7-27: Pipeline Operation at Interrupt Request Acknowledgment (Outline)



Remark: INT1 to INT4:Interrupt acknowledgment processing
IFX:Invalid instruction fetch
IDX:Invalid instruction decode

Table 7-3: Interrupt Response Time

Interrupt Response Time (Internal System Clocks)				Condition
	Internal Interrupt	External interrupt		
		INTP0 to INTP2, INTPE00 to INTPE52		
Minimum	5	5 + analog delay time	5 + digital noise filter	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	11 + digital noise filter	

7.9 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).

[MEMO]

Chapter 8 Clock Generator

8.1 Features

- Multiplication function by PLL synthesizer: 4 x multiplication

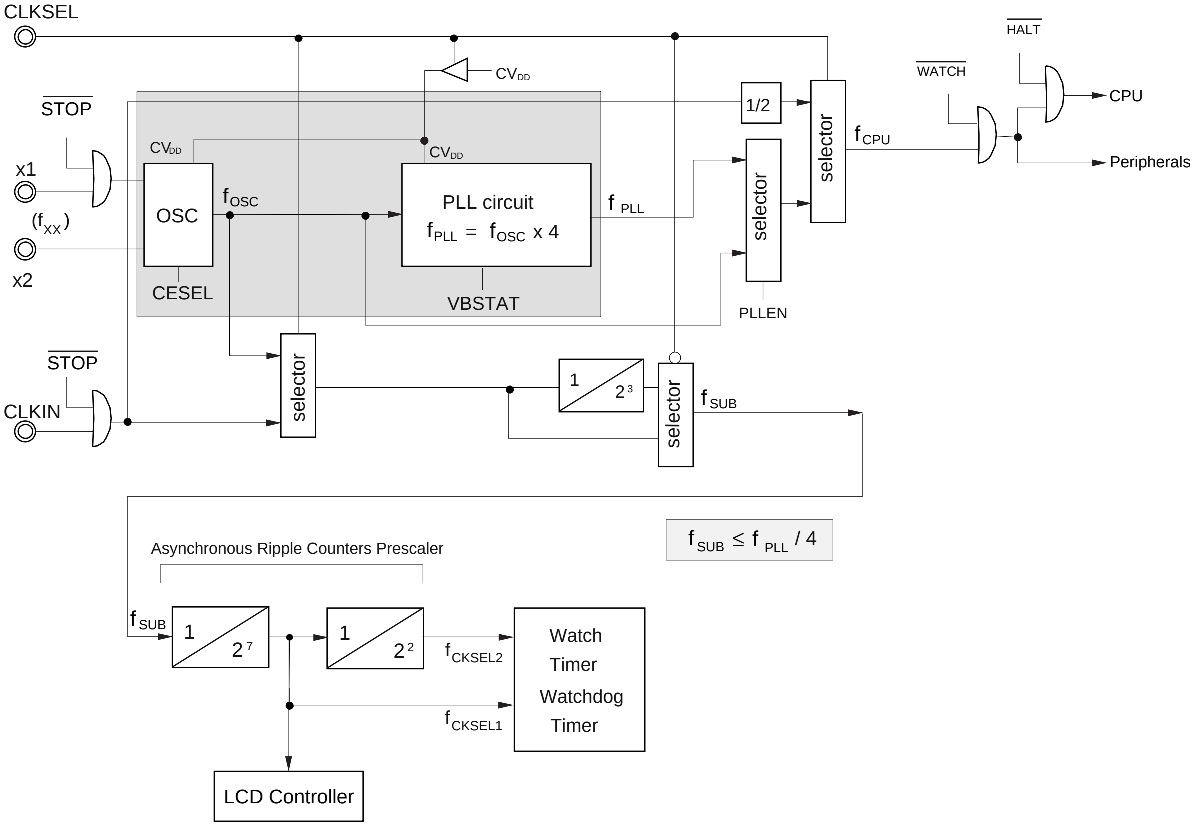
- Clock sources
  - Oscillation through oscillator connection
  - External clock input

- Power save modes
  - Watch mode
  - HALT mode
  - IDLE mode
  - STOP mode

- low power sub-clock for watch, watchdog and LCD to reduce power consumption in watch mode

8.2 Configuration

Figure 8-1: Block Diagram of the Clock Generator

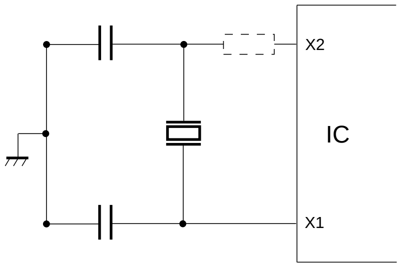


This block diagram does not necessarily show the exact wiring in hardware but the functional structure. For example the CLKSEL pin is not connected to the CV_{DD} of the PLL block but the function is as if.

8.3 Main system clock oscillator

The main system clock oscillator oscillates with a crystal resonator or ceramic resonator connected to the X1 and X2 pins.

Figure 8-2: Main system clock oscillator



8.4 Control Registers

8.4.1 Clock Control Register (CKC)

This is a 8-bit register that controls the clock management.
Data can be written to it only in a sequence of specific instructions so that its contents are not easily rewritten in case of program hang-up.
This register can be read or written in 8- or 1-bit units.

	7	6	5	4	3	2	1	0	Address	R/W	At Reset
CKC	PLLEN	0	TBCS	CESEL	0	0	0	0	FFFFF822H	R/W	00H

<table>
<tr><th>Bit name</th><th>Function</th></tr>
<tr><td>PLLEN</td><td>PLLEN enable bit
This bit enables or disables PLL operation.
0: PLL disabled
1: PLL enabled
<b>Remark:</b> PLL is enabled when “1” is written to this bit. Passing stabilization time and synchronization stage PLL output is used for system clock.</td></tr>
<tr><td>CESEL</td><td>Clock selection bit (X1,X2 pin function)
This bit sets PLL for proper operation with resonator or external oscillator.
0: Oscillation is enabled for a resonator
1: Oscillation is disabled<b>Note</b>
<b>Note:</b> If direct mode is selected by CLKSEL pin = 1, CESEL must be set to 1 by software after $\overline{\text{RESET}}$.
<b>Remark:</b> If CESEL is set to 1, oscillator stabilization stage will be skipped whenever power save mode is released by any interrupt or NMI request.</td></tr>
<tr><td>TBCS</td><td>Time base counter selection
This bit sets the clock source for the time base counter which is used to ensure the oscillator stabilization time after software STOP mode has been released by any interrupt or NMI request, and Flash stabilization time after the watch mode has been released.
In OSC mode (CESEL = 0):
<table>
<tr><th>TBCS</th><th>fxx = 4 MHz</th><th>fxx = 5 MHz</th></tr>
<tr><td>0</td><td>12.5 ms</td><td>10 ms</td></tr>
<tr><td>1</td><td>25 ms</td><td>20 ms</td></tr>
</table>
In direct mode (CESEL = 1):
<table>
<tr><th>TBCS</th><th>fxx = 4 MHz</th><th>fxx = 5 MHz</th></tr>
<tr><td>0</td><td>1.25 ms</td><td>1 ms</td></tr>
<tr><td>1</td><td>2.5 ms</td><td>2 ms</td></tr>
</table>
fxx: external oscillator frequency (clocking frequency / 2)
<b>Remark:</b> If CESEL is set to 1, oscillator stabilization stage will be skipped whenever power save mode is released by any interrupt or NMI request.</td></tr>
</table>

Caution: Data is set to the registers by the following sequence:

1. **Write the set data to the command register (PHCMD) (see Chapter 3.5 “Specific Registers” on page 115).**
2. **Write the set data to the destination register (CKC)**

To write data to the CKC register, use the store instruction (ST/SST) and bit manipulation instruction (SET1/CLR1/NOT1).
The contents of this register can be read in the normal sequence.

(1) Start-up after any Power-Save Mode condition recommendation

It is recommended to proceed in the following way when performing power-save functions:

1. Switch off the PLL (PLLEN bit of CKC)
2. Wait for “PLL switch off status” by reading VBSTAT bit in PSTAT register.
3. Enter the desired power-save function
4. (Processor waits on a start-up condition).
5. (Condition fulfilled: Processor wakes up)
6. Let the processor execute at least 5 NOP instructions
7. Let the processor run in a loop that causes a wait for at least 20 Microseconds
8. Switch on the PLL, if desired (PLLEN bit of CKC)
9. Continue with normal operation

8.4.2 PLL Status Register (PSTAT)

This is an 8-bit register that indicates PLL status.
This register can be read in 8- or 1-bit units.

	7	6	5	4	3	2	1	0	Address	R/W	At Reset
PSTAT	VBSTAT	0	0	0	0	0	0	0	FFFFF824H	R	xxH

Bit name	Function
VBSTAT	VBCLOCK status 0: PLL clock is not used for f_{CPU} (system clock) 1: PLL clock is used for f_{CPU} (system clock)

8.4.3 Clock select pin

Two basic operating modes are provided: OSC mode and direct mode. The operating mode is selected by specifying the CLKSEL pin.

Table 8-1: PLL mode / direct mode

CLKSEL	Description
0	Normal mode using a resonator (crystal or ceramic)
1	Direct clock input from CLOCKIN$^{\text{Note}}$

Note: If direct mode is selected by CLKSEL pin = 1, CESEL must be set to 1 by software after reset.

The CLKSEL pin level should be fixed according to the application system. Switching this pin during operation may cause malfunction.
In OSC mode, the external signal supplied by a connected resonator is used to generate the system clock that can be multiplied by the PLL synthesizer by setting the PLLEN bit. The input signal from an external resonator is recommended to range from 4 MHz to 5 MHz, and the PLL synthesizer multiplies the source clock signal by 4. Therefore, a system clock of up to approximately 20 MHz is available as shown below, which is suitable for application systems requiring low noise and low power consumption.

(1) Available clock frequencies in the OSC mode:

Table 8-2: Relation system clock to resonator frequency

System clock frequency (f_{CPU}) PLL On	System clock frequency (f_{CPU}) PLL Off	Frequency of external resonator (f_{XX})
20.000 MHz	5.0000 MHz	5.0000 MHz
16.000 MHz	4.0000 MHz	4.0000 MHz

In direct mode, an external clock whose frequency is twice the required system clock frequency should be connected in the same way as conventional types. In this mode, the OSC and the PLL synthesizer do not operate. Therefore, less power is consumed than in other modes. Consequently, this product is suitable for application systems that do not require very high frequency operation. To reduce noise most efficiently, the frequency of the externally supplied clock by CLKIN pin is recommended to be set to 40 MHz (when system clock f_{CPU} = 20 MHz).

(2) Available configurations:

Table 8-3: CESEL setting

	CLKSEL = 0	CLKSEL = 1
CESEL = 0	Normal mode using a resonator (crystal or ceramic)	Not used
CESEL = 1	Prohibited	Direct Mode using an external clock[Note]

Note: If direct mode is selected by CLKSEL pin = 1, CESEL must be set to 1 by software after reset.

8.5 Power Saving Functions

8.5.1 General

The device provides the following power saving functions. These modes can be combined and switched to suit the target application, which enables effective implementation of low-power systems.

Table 8-4: Power Saving Modes Overview

Clock Source		Mode	Operation of		Clock Supply to			
			oscillator	PLL	peripherals	CPU	LCD	watch
OSC mode	Initial Mode	Normal	❍	–	❍	❍	❍	❍
	PLL enabled	Normal	❍	❍	❍	❍	❍	❍
		watch mode/ IDLE[Note]	❍	–	–	–	(❍)	❍
		HALT	❍	❍	❍	–	❍	❍
		STOP	–	–	–	–	–	–
Direct mode		Normal	–	–	❍	❍	❍	❍
		watch mode/ IDLE[Note]	–	–	–	–	(❍)	❍
		HALT	–	–	❍	–	❍	❍
		STOP	–	–	–	–	–	–

Remarks: **1.** ❍ Operates

2. – Stopped

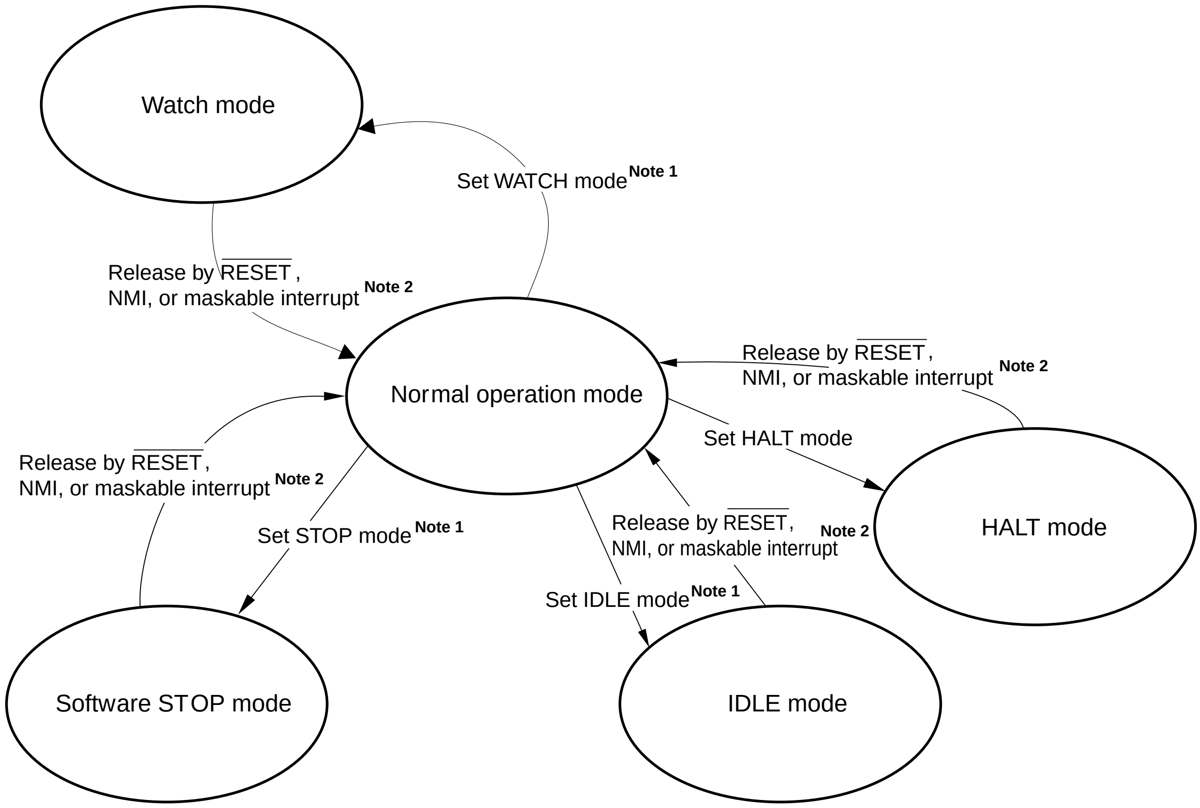
3. In addition the comparator circuit needs to be switched off in watch mode and STOP mode to save power consumption.

4. ( ): The LCD-driver probably have to be switched off in watch mode and/or in low voltage mode to meet the power consumption requirements.

Note: Only for flash devices, in IDLE mode flash memory is active, in watch mode the power supply to the flash memory is switched off.

Figure 8-3 shows the operation of the clock generator in normal operation mode, HALT mode, IDLE mode, WATCH mode, and software STOP mode.
An effective low power consumption system can be realized by combining these modes and switching modes according to the required use.

Figure 8-3: Power Save Mode State Transition Diagram


Watch mode
Set WATCH mode Note 1
Release by RESET, NMI, or maskable interrupt Note 2
Normal operation mode
Release by RESET, NMI, or maskable interrupt Note 2
Set HALT mode
HALT mode
Release by RESET, NMI, or maskable interrupt Note 2
Set STOP mode Note 1
Release by RESET, NMI, or maskable interrupt Note 2
Set IDLE mode Note 1
Software STOP mode
IDLE mode

Notes: **1.** Switch off PLL if activated before.
2. Enable PLL if required

The following table shows the supplied operating frequencies of all macros if a 4 MHz crystal is applied to the oscillator circuit or an external clock signal with 16 MHz is applied to the CLOCKIN pin.

Table 8-5: Power Saving Mode Frequencies

Clock Source		Mode	Operation of		Clock Supply to			
			oscillator	PLL	peripherals	CPU	LCD prescaler	watch/ watch-dog prescaler
OSC mode	Initial Status	Normal	❍	–	4 MHz	4 MHz	4 MHz / $[2^7]$	4 MHz / $[2^7, 2^9]$
	PLL enabled	Normal	❍	❍	16 MHz	16 MHz	4 MHz / $[2^7]$	4 MHz / $[2^7, 2^9]$
		watch mode/ IDLE[Note1]	❍	–	–	–	4 MHz / $[2^7]$	4 MHz / $[2^7, 2^9]$
		HALT	❍	❍	16 MHz	–	4 MHz / $[2^7]$	4 MHz / $[2^7, 2^9]$
		STOP	–	–	–	–	–	–
Direct mode		Normal	–	–	16 MHz	16 MHz	32 MHz / $[2^{10}]$	32 MHz / $[2^{10}, 2^{12}]$
		watch mode/ IDLE[Note1]	–	–	–	–	32 MHz / $[2^{10}]$	32 MHz / $[2^{10}, 2^{12}]$
		HALT	–	–	16 MHz		32 MHz / $[2^{10}]$	32 MHz / $[2^{10}, 2^{12}]$
		STOP	–	–	–	–	–	–

Note: Only for flash devices: In IDLE mode flash memory is active, in watch mode the power supply to the flash memory is switched off.

8.5.2 Power Save Modes Outline

V850E/CA1 / ATOMIC is provided with the following standby modes: HALT, IDLE, WATCH, and software STOP.
Application systems, which are designed so that these modes are switched appropriately according to operation purposes, reduce power consumption efficiently.

(1) HALT mode:

In this mode supply of the operating clock to the CPU is stopped whereby other on-chip peripheral functions continue to operate. Combining this mode with the normal operating mode to provide intermittent operations enables the overall system power consumption to be reduced.
This mode is entered by executing the dedicated instruction (HALT).

(2) IDLE mode:

In this mode, the clock generator continues to operate but stopping the supply of internal system clock stops the overall system. As it is not necessary to secure the oscillation stabilization time, it is possible to switch to the normal operating mode quickly in response to a release signal.
This mode provides low power consumption, where the power is only consumed from the OSC, Watch/Watchdog, LCD and the flash memory.
This mode is entered by setting registers with software.

(3) WATCH mode:

In this mode, the clock generator stop to supply the clock excluding Watch/Watchdog timer unit. The entire system stops. This mode provides ultra-low power consumption, where the power consumed is only from OSC, Watch/ Watchdog and LCD circuit. (In addition clock supply to LCD may be stopped by software.)
This mode is entered by setting registers with software.

(4) Software STOP mode:

In this mode, the clock generator is stopped and the entire system stops. This mode provides ultra-low power consumption, where the power consumed is only leakage current.
This mode is entered by setting registers with software.

Remark: In the HALT mode, both the oscillator and the PLL continue to operate depend on PLLEN bit.
By using PLL mode control, PLL is turn off to achieve low power.
However, when the external clock drives this product, the oscillator is stopped. In contrast, the PLL synthesizer stops in the direct mode.

8.5.3 HALT mode

In this mode, the CPU clock is stopped, though the clock generators (oscillator and PLL synthesizer) continue to operate for supplying clock signals to other peripheral function circuits.
Setting the HALT mode when the CPU is idle reduces the total system power consumption.
In the HALT mode, program execution is stopped but the contents of all registers and internal RAM prior are retained as is.
On-chip peripheral hardware irrelevant to the CPU instruction execution also continues to operate. The state of the various hardware units in the HALT mode is tabulated below.

Table 8-6: Operating states in HALT mode

Items	Operation
Clock generator	Operating
Internal system clock	Operating
WT, WDT, LCD clock	Operating
CPU	Stopped
I/O line	Unchanged
Peripheral function	Operating
Internal data	Retains all internal data before entering HALT mode, such as CPU registers, status, data, and on-chip RAM.
CLKOUT pin	Clock output (when not inhibited by port setting)
D[15:0], A[23:0], $\overline{\text{RD}}$, $\overline{\text{UWR}}$/ $\overline{\text{LWR}}$, $\overline{\text{CS}}$[2:4], $\overline{\text{WAIT}}$	Operates

Remark: Even after the HALT instruction is executed, instruction fetch operations continue until the internal instruction prefetch queue is full. After the queue becomes full, the CPU stops with the items set as tabulated above.

HALT mode release:

The HALT mode can be released by a non-maskable interrupt request, an unmasked maskable interrupt request, or $\overline{\text{RESET}}$ signal input.

(1) Release by interrupt request

The HALT mode is released unconditionally by an unmasked maskable interrupt request regardless of its priority level. However, if the HALT mode is entered during execution of an interrupt handler, the operation differs on interrupt priority levels as follows:

(a)If an interrupt request less prioritized than the currently serviced interrupt request is generated, the HALT mode is released but the interrupt is not acknowledged. The interrupt request itself is retained.

(b) If an interrupt request (including a non-maskable one) prioritized than the currently serviced interrupt request is generated, the interrupt request is acknowledged along with the HALT mode release.

Table 8-7: Operation after HALT mode release by interrupt request

Release cause	EI state	DI state
NMI request	Branches to handler address.	
Maskable interrupt request	Branches to handler address, or executes the next instruction.	Executes the next instruction.

Remark: If HALT mode is entered during execution of a particular interrupt handler and an unmasked interrupt request with a higher priority than the previous one is subsequently generated, the program branches to the vector address for the latter interrupt.

(2) Release by $\overline{\text{RESET}}$ pin input

This operation is the same as normal reset operation.

8.5.4 IDLE Mode

In this mode, the CPU clock is stopped resulting in stop of the entire system, though the clock generators (oscillator and PLL synthesizer) continue to operate.
As it is not necessary to secure the oscillator oscillation stabilization time and the PLL lock-up time, it is possible to quickly switch to the normal operating mode in response to a release cause. The IDLE mode can be entered by configuring the PSM and PSC registers.
In the IDLE mode, program execution is stopped but the contents of all registers and internal RAM prior to entering this mode are retained. On-chip peripheral hardware operation is also stopped.

The state of the various hardware units in the IDLE mode is tabulated below.

Table 8-8: Operating States in IDLE Mode

Items	Operation
Clock generator	Operating
Internal system clock	Stopped
WT, WDT, LCD clock	Operating
CPU	Stopped
I/O line	Unchanged
Peripheral function	Stops exclude watch/watchdog, LCD
Internal data	Retains all internal data before entering IDLE mode, such as CPU registers, status, data, and on-chip RAM.
D[15:0], A[23:0]	Hi-Z
$\overline{\text{RD}}$, $\overline{\text{UWR}}$/$\overline{\text{LWR}}$, $\overline{\text{CS}}$[4:2]	H
CLKOUT	L
$\overline{\text{WAIT}}$	Input value is not sampled

IDLE mode release:

Release operation is same as release from HALT mode.
The IDLE mode is released by NMI, $\overline{\text{RESET}}$ signal input, or an unmasked maskable interrupt request.

(a) Release by Interrupt input:
When the IDLE mode is released, the NMI request is acknowledged.
If the IDLE mode is entered during the execution of NMI handler, the IDLE mode is released but the interrupt is not acknowledged. The interrupt itself is retained.

(b) Release by $\overline{\text{RESET}}$ input:
This operation is the same as normal reset operation.

8.5.5 WATCH mode

In this mode f_{CPU} clock is stopped while the oscillator continue to operate to achieve low power, though only oscillator & Watch/watchdog timer continue to operate.
This mode compensates the HALT modes concerning the oscillator stabilization time and power consumption. Only for the Flash memory an additional stabilization time is required.
This mode is entered by configuring the PSM and PSC registers.
In the WATCH mode, program execution is stopped but the contents of all registers and internal RAM prior to entering this mode are retained. On-chip other peripheral hardware operation is also stopped.

The state of the various hardware units in the WATCH mode is tabulated below.

Table 8-9: Operating States in WATCH Mode

Items	Operation
Clock generator	Operating
Internal system clock	Stopped
WT, WDT, LCD clock	Operating
CPU	Stopped
I/O line	Unchanged
Peripheral function	Stops exclude watch/watchdog, LCD
Internal data	Retains all internal data before entering WATCH mode, such as CPU registers, status, data, and on-chip RAM.
D[15:0], A[23:0]	Hi-Z
$\overline{\text{RD}}$, $\overline{\text{UWR}}$/$\overline{\text{LWR}}$, $\overline{\text{CS}}$[4:2]	H
CLKOUT	L
$\overline{\text{WAIT}}$	Input value is not sampled

Watch mode release:

The WATCH mode can be released by a non-maskable interrupt request, an unmasked maskable interrupt request, or $\overline{\text{RESET}}$ signal input.

(1) Release by interrupt request:

The WATCH mode is released unconditionally by an unmasked maskable interrupt request regardless of its priority level. After 1 ms has passed, CPU starts operation. However, if the WATCH mode is entered during execution of an interrupt handler, the operation differs on interrupt priority levels as follows:

(a) If an interrupt request less priorities than the currently serviced interrupt request is generated, the WATCH mode is release but the interrupt is not acknowledged. The interrupt request itself is retained.

(b) If an interrupt request (including a non-maskable one) priorities than the currently serviced interrupt request is generated, the interrupt request is acknowledged along with the WATCH mode release.

Table 8-10: Operation after WATCH mode release by interrupt request

Release cause	EI state	DI state
NMI request	Branches to handler address.	
Maskable interrupt request	Branches to handler address, or executes the next instruction.	Executes the next instruction.

Remark: If WATCH mode is entered during execution of a particular interrupt handler and an unmasked interrupt request with a higher priority than the previous one is subsequently generated, the program branches to the vector address for the later interrupt.

(2) When released by $\overline{\text{RESET}}$ input

This operation is the same as normal reset operation. 1 ms Flash charge pump stabilization time must be ensured by reset input.

(3) When released by WATCHDOG TIMER $\overline{\text{RESET}}$ input

After 1 ms has passed, CPU starts operation.

Remark: Before entering the WATCH mode the PLL must be switched off by software. After the WATCH mode has been released the PLL can be switched on again. However, the start-up of the PLL causes always a certain delay of some Milliseconds. During this time, the clock operates, but the CPU operation is suspended due to clock security reasons.
If it is required to have a fast response when waking up from WATCH mode, the PLL should not be re-enabled after waking up, as this causes again the delay. In this case, time-relevant reactions of the CPU should be done first, before re-enabling the PLL.

8.5.6 Software STOP mode

In this mode, the CPU clock is stopped including the clock generators (oscillator and PLL synthesizer), resulting in stop of the entire system for ultra-low power consumption (the only consumed is device leakage current). When this mode is released, the oscillation stabilization time for the oscillator should be secured until the system clock is stabilized. However, when the external clock operates this product, securing the oscillation stabilization time for the oscillator until the system clock is stabilized is unnecessary. In the direct mode as well, the lock-up time does not have to be secured.
This mode is entered by setting the PSM & PSC register.
In this mode, the program execution stops, but the contents of all registers and internal RAM prior to entering this mode are retained. V850E/CA1 / ATOMIC peripherals operations are also stopped.

The state of the various hardware units in the software STOP mode is tabulated below.

Table 8-11: Operating States in STOP Mode

Items	Operation
Clock generator	Stopped
Internal system clock	Stopped
WT, WDT, LCD clock	Stopped
CPU	Stopped
I/O line**Note**	Unchanged
Peripheral function	Stopped
Internal data**Note**	Retains all previous internal data, such as CPU registers, status, data, and on-chip RAM.
D[15:0], A[23:0]	Hi-Z
$\overline{\text{RD}}$, $\overline{\text{UWR}}$/$\overline{\text{LWR}}$, $\overline{\text{CS}}$[4:2]	H
CLKOUT	L
$\overline{\text{WAIT}}$	Input value is not sampled

Note: When the V_{DD} value is within the operating range. However, even if V_{DD} falls below the lowest operating voltage, the internal RAM content is retained as long as the data retention voltage V_{DDDR} is maintained.

STOP mode release:

The STOP mode can be released by a non-maskable interrupt request, an unmasked maskable interrupt request, or $\overline{\text{RESET}}$ signal input.

8.6 Register Description

8.6.1 Power Save Control Register (PSC)

This is an 8-bit register that controls the power save mode.
Data can be written only in a sequence of specific instructions so that its contents are not easily rewritten in case of program hang-up (see Chapter 3.5 "Specific Registers" on page 115).

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

	7	6	5	4	3	2	1	0	Address	R/W	At Reset
PSC	0[Note]	NMI1M	NMI0M	INTM	0	0	STB	0	FFFFF1FEH	R/W	00H

Bit name	Function
STB	Power save mode specification 0: IDLE, WATCH, STOP mode are released 1: IDLE, WATCH, STOP mode are entered
INTM	Intsignal release mask 0: Release by maskable interrupt 1: Don't release by maskable interrupt
NMI0M	Intsignal release mask 0: Release by NMIVC 1: Don't release by NMIVC
NMI1M	Intsignal release mask 0: Release by NMIWDT 1: Don't release by NMIWDT

Note: If this bit is set to 1, proper operation can not be guaranteed!

Data is set in the power save control register (PSC) according to the following sequence.

<1> Set the power save mode register (PSM) (with the following instructions).

- Store instruction (ST/SST instruction)
- Bit manipulation instruction (SET1/CLR1/NOT1 instruction)

<2> Prepare data in any one of the general-purpose registers to set to the specific register.

<3> Write arbitrary data to the command register (PRCMD).

<4> Set the power save control register (PSC) (with the following instructions).

- Store instruction (ST/SST instruction)
- Bit manipulation instruction (SET1/CLR1/NOT1 instruction)

<5> Assert the NOP instructions (5 instructions (<5> to <9>).

Sample coding

```
<1> ST.B    r11, PSM [r0]      ; Set PSM register
<2> MOV     0x04, r10
<3> ST.B    r10, PRCMD [r0]    ; Write PRCMD register
<4> ST.B    r10, PSC [r0]      ; Set PSC register
<5> NOP                        ; Dummy instruction
<6> NOP                        ; Dummy instruction
<7> NOP                        ; Dummy instruction
<8> NOP                        ; Dummy instruction
<9> NOP                        ; Dummy instruction
(next instruction)             ; Execution routine after software STOP mode and IDLE mode release
```

No special sequence is required to read the specific register.

Cautions: 1. A store instruction for the command register does not accept interrupts. This coding is made on assumption that <3> and <4> above are executed by the program with consecutive store instructions. If another instruction is set between <3> and <4>, the above sequence may become ineffective when the interrupt is accepted by that instruction, and a malfunction of the program may result.

2. Although the data written to the PHCMD register is dummy data, use the same register as the general register used in specific register setting <4> for writing to the PHCMD register (<3>). The same method should be applied when using a general register for addressing.

3. At least 5 NOP instructions must be inserted after executing a store instruction to the PSC register to set software STOP or IDLE mode.

4. Do not perform a write operation to the PRCMD and specific registers using DMA transfer.

To write data to the PSC register, use the store instruction (ST/SST) and bit manipulation instruction (SET1/CLR1/NOT1).

The contents of this register can be read in the normal sequence.

8.6.2 Power Save Mode Register (PSM)

This is a 8-bit register that control the power save mode.
This register can be read or written in 8- or 1-bit units.

	7	6	5	4	3	2	1	0	Address	R/W	At Reset
PSM	0	0	0	0	0	0	PSM1	PSM0	FFFFF820H	R/W	00H

<table>
<tr><th>Bit name</th><th>Function</th></tr>
<tr><td>PSM1, PSM0</td><td>Standby mode specification after STB bit (PSC.1) set to “1”.
Sets the standby mode.
<table>
<tr><th>PSM1</th><th>PSM0</th><th>Standby Mode</th></tr>
<tr><td>0</td><td>0</td><td>IDLE</td></tr>
<tr><td>0</td><td>1</td><td>STOP</td></tr>
<tr><td>1</td><td>0</td><td>WATCH</td></tr>
<tr><td>1</td><td>1</td><td>reserved</td></tr>
</table>
<b>Note:</b> The setting is automatically reset to “00” when STOP mode is released.</td></tr>
</table>

The contents of this register can be read in the normal sequence.

8.7 Securing Oscillation Stabilization Time

8.7.1 Oscillation stabilization time security specification

Two methods can be used to secure the required stabilization times from when watch mode or software STOP mode is released.

(1) Securing the time using an on-chip time base counter

Watch mode and software STOP mode are released when a valid edge is input to the NMI pin or a maskable interrupt request is input (INTPn). Valid edge input to the pin causes the time base counter (TBC) to start counting, and the time until the clock output from the oscillator stabilizes is secured during that counting time.

Oscillation stabilization time = TBC counting time

After a fixed time, internal system clock output begins, and processing branches to the NMI interrupt or maskable interrupt (INTPn) handler address.

Figure 8-4: WATCH mode release by NMI or INT

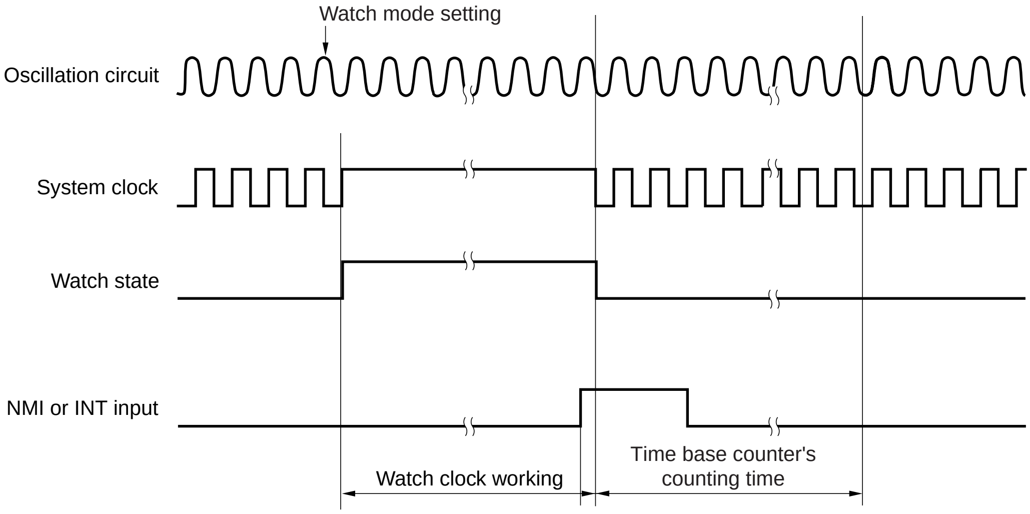
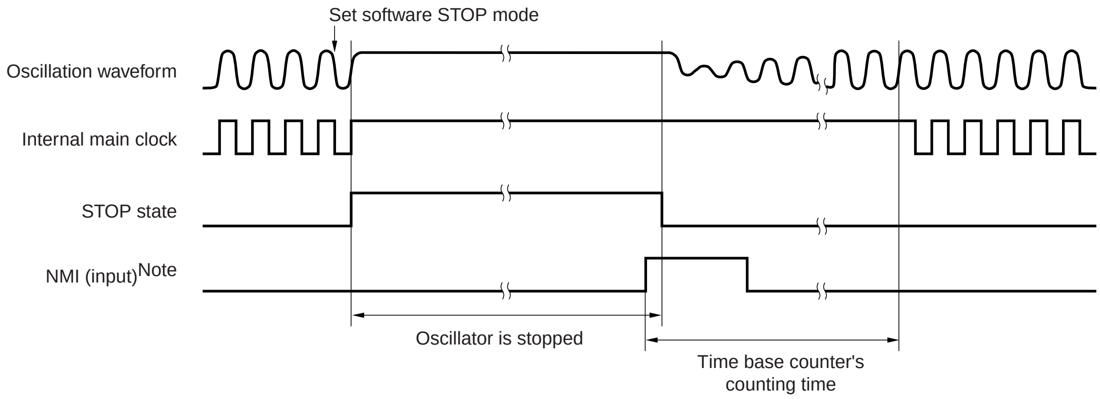


Figure 8-5: STOP mode release by NMI or INT



Note: Valid edge: When specified as the rising edge.

The NMI pin should usually be set to an inactive level (for example, high level when the valid edge is specified as the falling edge) in advance.
Watch mode and software STOP mode are immediately released if an operation is performed according to NMI valid edge input or maskable interrupt request input (INTPn) timing in which STOP mode is set until the CPU acknowledges the interrupt.
If direct mode (CESEL bit of CKC register = 1) is used, stabilization stage will be skipped.
If PLL mode and resonator connection mode (CESEL bit of CKC register = 0) are used, program execution begins after the oscillation stabilization time or the flash stabilization time is secured according to the time base counter, which begins counting due to NMI pin valid edge input.

(2) Securing the time according to the signal level width ($\overline{\text{RESET}}$ pin input)

Watch mode and software STOP mode are released due to falling edge input to the $\overline{\text{RESET}}$ pin. The time until the clock output from the oscillator stabilizes is secured according to the low level width of the signal that is input to the pin.

The supply of internal system clocks begins after a rising edge is input to the $\overline{\text{RESET}}$ pin, and processing branches to the handler address used for a system reset.

Figure 8-6: WATCH mode release by reset or watchdog timer

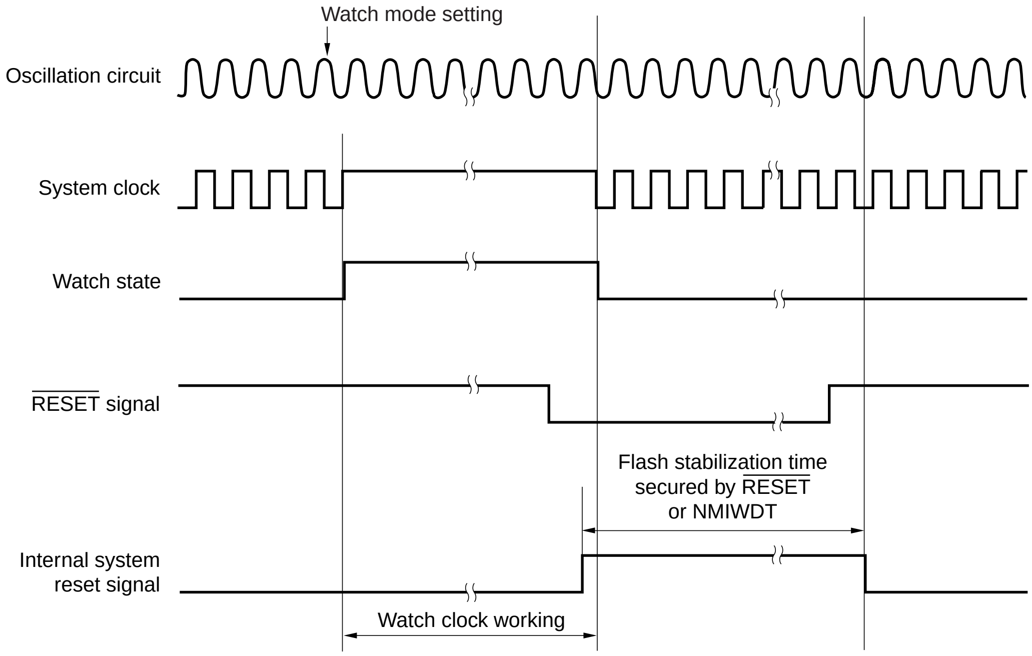
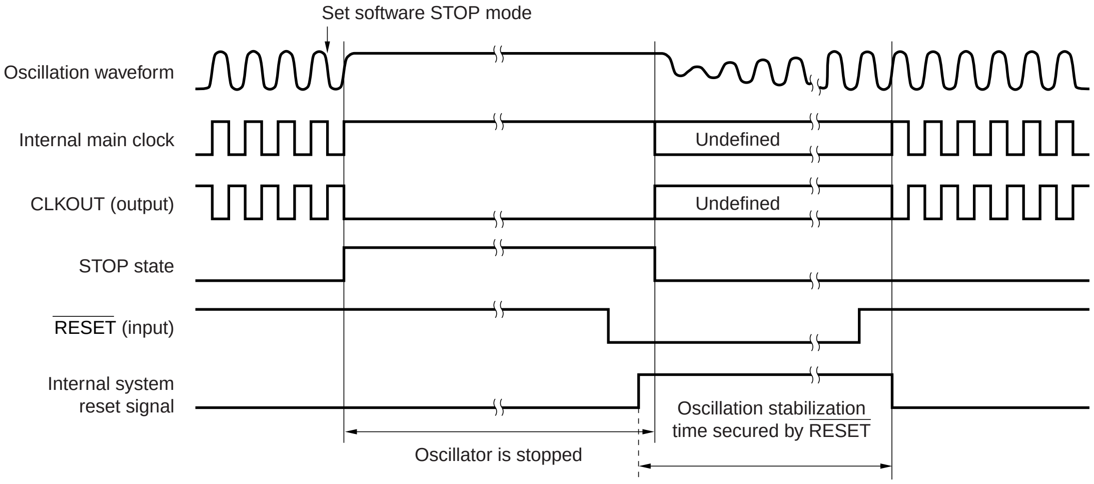


Figure 8-7: STOP mode release by $\overline{\text{RESET}}$ pin input



8.7.2 Time base counter (TBC)

The time base counter (TBC) is used to secure the oscillator’s oscillation stabilization time when software STOP mode is released. It also is used to secure the flash stabilization time when software WATCH mode is released.
The TBC count clock is selected according to the TBCS bit of the CKC register, and the next counting time can be set (reference).

Table 8-12: Counting Time Examples

TBCS Bit	Counting Time	
	f_{XX} = 4.0000 MHz	f_{XX} = 5.0000 MHz
0	12.5 ms	10.0 ms
1	25.0 ms	20.0 ms

Remark: fxx: External oscillation frequency

Chapter 9 Timer / Counter (Real Time Pulse Unit)

9.1 Timer D

9.1.1 Features (timer D)

Timer D (TMD) functions as a 16-bit interval timer.

9.1.2 Function overview (timer D)

- 16-bit interval timer: 2 channels
- Compare registers: 2
- Count clock selected from divisions of internal system clock
  (maximum frequency of count clock: 10 MHz @ f_{CPU} = 20 MHz)
- Prescaler division ratio
  The following division ratios can be selected related to the internal system clock (f_{CPU}).

Division Ratio	Count Clock
1/2	$f_{CPU}/2$
1/4	$f_{CPU}/4$
1/8	$f_{CPU}/8$
1/16	$f_{CPU}/16$
1/32	$f_{CPU}/32$
1/64	$f_{CPU}/64$
1/128	$f_{CPU}/128$
1/256	$f_{CPU}/256$

- Interrupt request sources: 2
  - Compare match interrupt
    TINTCMDn generated with CMDn match signal
- Timer clear
  TMDn register can be cleared by CMDn register match.

Remarks: **1.** f_{CPU}: Internal system clock.
2. n = 0, 1

9.1.3 Basic configuration

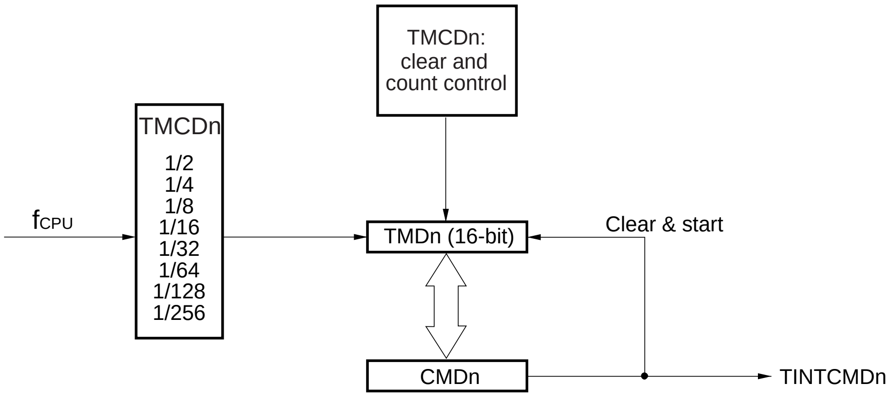
Table 9-1: Timer D Configuration List

Timer	Count Clock	Register	R/W	Generated Interrupt Signal	Capture Trigger	Timer Output S/R	Other Functions
Timer D	$f_{CPU}/2$, $f_{CPU}/4$, $f_{CPU}/8$, $f_{CPU}/16$, $f_{CPU}/32$,$f_{CPU}/64$, $f_{CPU}/128$, $f_{CPU}/256$	TMD0	R	–	–	–	–
		CMD0	R/W	TINTCMD0	–	–	–
		TMCD0	R/W	–	–	–	–
		TMD1	R	–	–	–	–
		CMD1	R/W	TINTCMD1	–	–	–
		TMCD1	R/W	–	–	–	–

Remarks: **1.** f_{CPU}: Internal system clock
2. S/R: Set/Reset

Figure 9-1 shows the block diagram of the channel of timer D.

Figure 9-1: Block Diagram of Timer D



Remark: n = 0, 1

(1) Timers D Registers 0, 1 (TMD0, TMD1)

TMDn is a 16-bit timer. It is mainly used as an interval timer for software (n = 0, 1).
Starting and stopping TMDn is controlled by the CE bit of the timer D control register n (TMCDn).
A division by the prescaler can be selected for the count clock from among $f_{CPU}/2$, $f_{CPU}/4$, $f_{CPU}/8$, $f_{CPU}/16$, $f_{CPU}/32$, $f_{CPU}/64$, $f_{CPU}/128$, and $f_{CPU}/256$ by the CS2 to CS0 bits of the TMCDn register.
TMDn is read-only in 16-bit units.

Figure 9-2: Timer D Registers 0, 1 (TMD0, TMD1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
TMD0																	FFFFF540H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
TMD1																	FFFFF550H	0000H

The conditions for which the TMDn register becomes 0000H are shown below.

- Reset input
- CAE bit = 0
- CE bit = 0
- Match of TMDn register and CMDn register
- Overflow

Cautions: 1. If the CAE bit of the TMCDn register is cleared (0), a reset is performed asynchronously.
2. If the CE bit of the TMCDn register is cleared (0), a reset is performed, synchronized with the internal clock. Similarly, a synchronized reset is performed after a match with the CMDn register and after an overflow.
3. The count clock must not be changed during a timer operation. If it is to be overwritten, it should be overwritten after the CE bit is cleared (0).
4. Up to $f_{CPU}/2$ clocks are required after a value is set in the CE bit until the set value is transferred to internal units. When a count operation begins, the count cycle from 0000H to 0001H differs from subsequent count cycles.
5. After a compare match is generated, the timer is cleared at the next count clock. Therefore, if the division ratio is large, the timer value may not be zero even if the timer value is read immediately after a match interrupt is generated.

(2) Timer D compare registers 0, 1 (CMD0 to CMD1)

CMDn and the TMDn register count value are compared, and an interrupt request signal (TINTCMDn) is generated when a match occurs. TMDn is cleared, synchronized with this match. If the CAE bit of the TMCDn register is set to 0, a reset is performed asynchronously, and the registers are initialized (n = 0, 1).

The CMDn register is configured with a master/slave configuration. When a write operation to a CMDn register is performed, data is first written to the master register and then the master register data is transferred to the slave register. In a compare operation, the slave register value is compared with the count value of the TMDn register. When a read operation to a CMDn register is performed, data in the master side is read out.

CMDn can be read/written in 16-bit units.

Figure 9-3: Timer D Compare Registers 0, 1 (CMD0 to CMD1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CMD0																	FFFFF542H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CMD1																	FFFFF552H	0000H

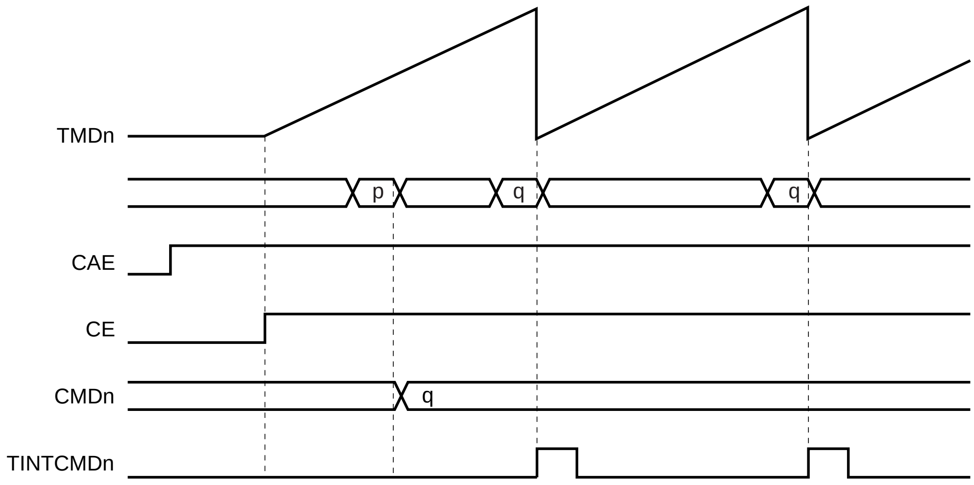
Cautions: 1. A write operation to the a CMDn register requires f_{CPU}/2 clocks until the value that was set in the CMDn register is transferred to internal units. When writing continuously to the CMDn register, be sure to reserve a time interval of at least f_{CPU}/2 clocks.

2. The CMDn register can be overwritten only once in a single TMDn register cycle (from 0000H until an TINTCMDn interrupt is generated due to a match of the TMDn register and CMDn register). If this cannot be secured by the application, make sure that the CMDn register is not overwritten during timer operation.

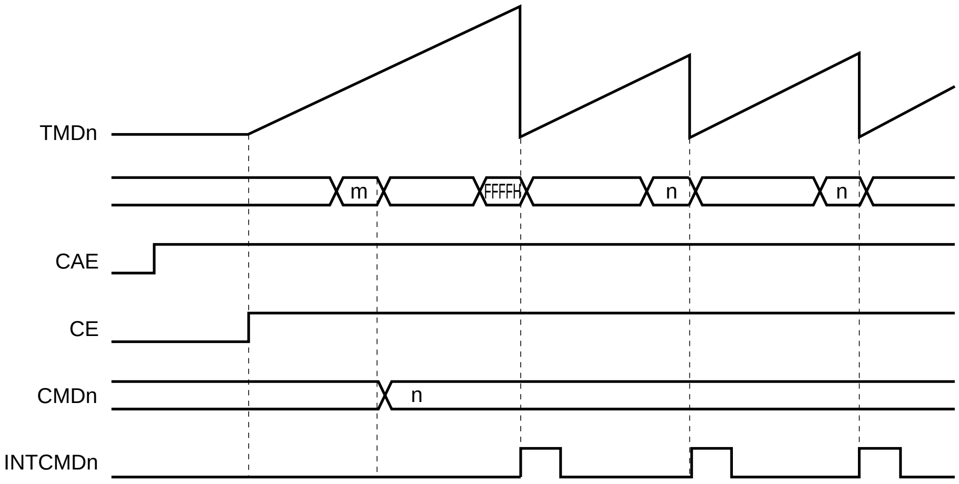
3. Note that an TINTCMDn interrupt will be generated after an overflow if a value less than the counter value is written in the CMDn register during TMDn register operation (Figure 9-4).

Figure 9-4: Example of Timing During TMD Operation

(a) When TMDn < CMDn



(b) When TMDn > CMDn



Remarks: **1.** p = TMDn value when overwritten
2. q = CMDn value when overwritten
3. n = 0, 1

9.1.4 Control register

(1) Timer D control registers 0, 1 (TMCD0 to TMCD1)

The TMCDn register controls the operation of timer D (n = 0, 1).
This register can be read/written in 8- or 1-bit units.

Figure 9-5: Timer D Control Register 0, 1 (TMCD0 to TMCD1)

	7	6	5	4	3	2	1	0	Address	Initial value
TMCD0	0	CS2	CS1	CS0	0	0	CE	CAE	FFFFF544H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
TMCD1	0	CS2	CS1	CS0	0	0	CE	CAE	FFFFF554H	00H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>6 to 4</td><td>CS2 to CS0</td><td>Selects the TMDn count clock (n = 0, 1).
<table>
<tr><th>CS2</th><th>CS1</th><th>CS0</th><th>Count Clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>f_{CPU} /2</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>f_{CPU} /4</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>f_{CPU} /8</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>f_{CPU} /16</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>f_{CPU} /32</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>f_{CPU} /64</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>f_{CPU} /128</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>f_{CPU} /256</td></tr>
</table>
<b>Caution: Do not change the CS2 to CS0 bits during timer operation. If they are to be changed, they must be changed after setting the CE bit to 0. If the CS2 to CS0 bits are overwritten during timer operation, the operation is not guaranteed.</b></td></tr>
<tr><td>1</td><td>CE</td><td>Count Enable: Controls the operation of TMDn (n = 0, 1).
0: Disable count (timer stopped at 0000H and does not operate)
1: Perform count operation
<b>Caution: CE bit is not cleared even if a match is detected by the compare operation. To stop the count operation, clear the CE bit.</b></td></tr>
<tr><td>0</td><td>CAE</td><td>Count Action Enable: Controls the internal count clock.
0: Asynchronously reset entire TMDn unit. Stop clock supply to TMDn unit.
1: Supply clock to TMDn unit (n = 0, 1).
<b>Cautions: 1. When CAE = 0 is set, the TMDn unit can be reset asynchronously.</b>
<b>2. When CAE = 0, the TMDn unit is in a reset state. To operate TMDn, first set CAE = 1.</b>
<b>3. When the CAE bit is changed from 1 to 0, all the registers of the TMDn unit are initialized. When again setting CAE = 1, be sure to then again set all the registers of the TMDn unit.</b></td></tr>
</table>

Caution: The CAE bit and CE bit cannot be set at the same time. Be sure to set the CAE bit prior to setting the CE bit.

9.1.5 Operation

(1) Compare operation

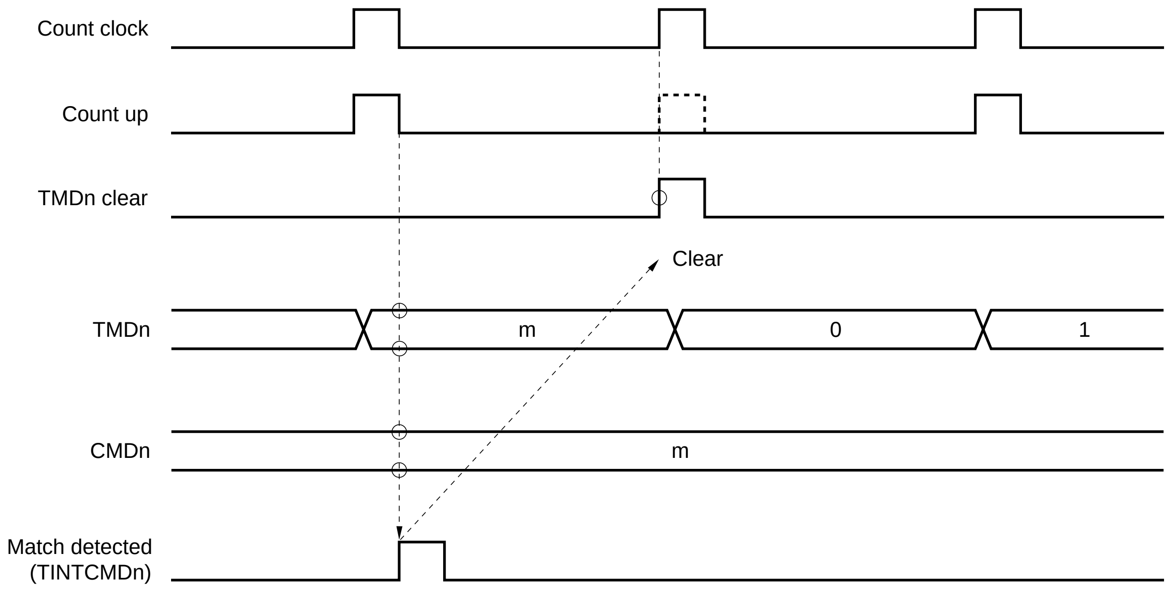
TMDn can be used for a compare operation in which the value that was set in a compare register (CMDn) is compared with the TMDn count value (n = 0, 1).

If a match is detected by the compare operation, an interrupt (TINTCMDn) is generated. The generation of the interrupt causes TMDn to be cleared (0) at the next count timing. This function enables timer D to be used as an interval timer.

CMDn can also be set to 0. In this case, when an overflow occurs and TMDn becomes 0, a match is detected and TINTCMDn is generated. Although the TMDn value is cleared (0) at the next count timing, TINTCMDn is not generated according to this match.

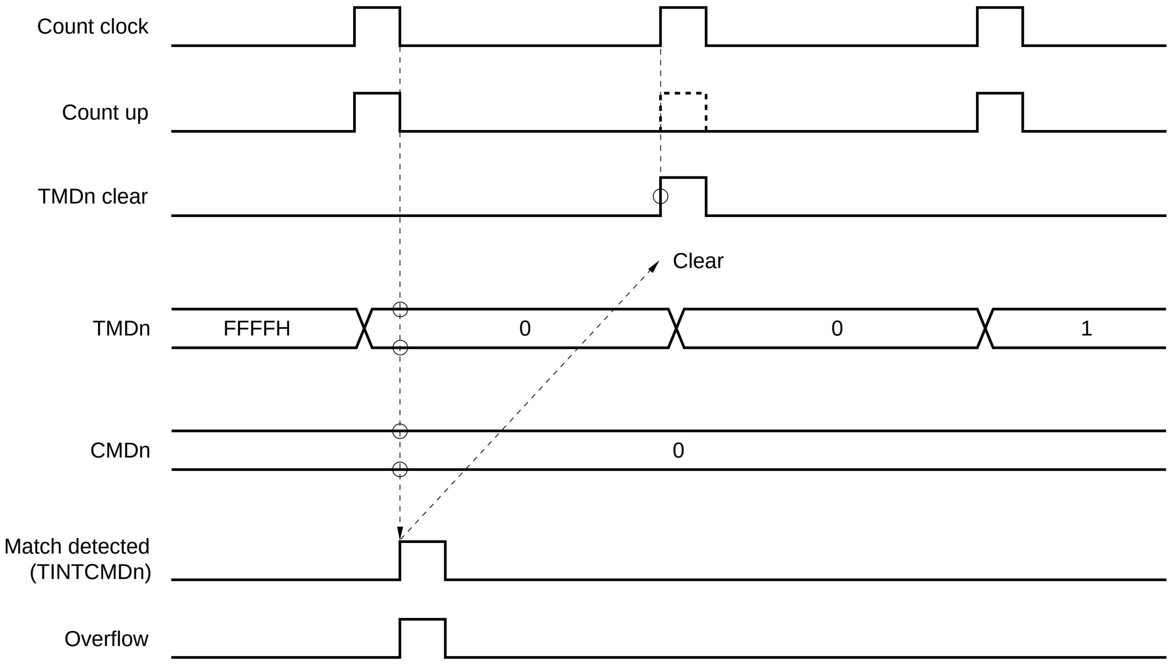
Figure 9-6: TMD Compare Operation Example

(a) When CMDn is set to m (non-zero)



Remarks: **1.** Interval time = (m + 1) × Count clock cycle
2. m = 1 to 65536 (FFFFH)
3. n = 0, 1

(b) When CMDn is set to 0



Remark: Interval time = (FFFFH + 2) × Count clock cycle

9.1.6 Application example

(1) Interval timer

This section explains an example in which timer D is used as an interval timer with 16-bit precision. Interrupt requests (TINTCMDn) are output at equal intervals (refer to **Figure 9-6: TMD Compare Operation Example**). The setup procedure is shown below (n = 0, 1).

<1> Set (1) the CAE bit.
<2> Set each register.
- Select the count clock using the CS2 to CS0 bits of the TMCDn register.
- Set the compare value in the CMDn register.

<3> Start counting by setting (1) the CE bit.
<4> If the TMDn register and CMDn register values match, an TINTCMDn interrupt is generated.
<5> TINTCMDn interrupts are generated thereafter at equal intervals.

9.1.7 Precautions

Various precautions concerning timer D are shown below.

(1) To operate TMDn, first set (1) the CAE bit of the TMCDn register.

(2) Up to $f_{CPU}/2$ clocks are required after a value is set in the CE bit of the TMCDn register until the set value is transferred to internal units. When a count operation begins, the count cycle from 0000H to 0001H differs from subsequent count cycles.

(3) To initialize the TMDn register status and start counting again, clear (0) the CE bit and then set (1) the CE bit after an interval of $f_{CPU}/2$ clocks has elapsed.

(4) Up to $f_{CPU}/2$ clocks are required until the value that was set in the CMDn register is transferred to internal units. When writing continuously to the CMDn register, be sure to secure a time interval of at least $f_{CPU}/2$ clocks.

(5) The CMDn register can be overwritten only once during a timer/counter operation (from 0000H until an TINTCMDn interrupt is generated due to a match of the TMDn register and CMDn register). If this cannot be secured, make sure that the CMDn register is not overwritten during a timer/counter operation.

(6) The count clock must not be changed during a timer operation. If the clock selection by CS2 to CS0 bits is going to be changed, it should be overwritten after the CE bit is cleared (0). If the count clock is changed during a timer operation, operation cannot be guaranteed.

(7) An TINTCMDn interrupt will be generated after an overflow if a value less than the counter value is written in the CMDn register during TMDn register operation.

Remark: n = 0, 1

9.2 Timer E

9.2.1 Features (timer E)

The 3 x 6 channels 16/32-bit multi purpose timers En (TMEn) (n = 0 to 2) operate as

- Pulse interval and frequency measurement counter
- Up/down event counter
- Interval timer
- Programmable pulse output
- PWM output timer

9.2.2 Function overview (timer E)

- 16-bit timer/counter (TBASE0n, TBASE1n): 2 channels each unit n (n = 0 to 2)
- Bit length
  - Timer En registers (TBASE0n, TBASE1n): 16 bits
  - During cascade operation: 32 bits (higher 16 bits:TBASE1n, lower 16 bits: TBASE0n)
- Capture/compare register
  - In 16-bit mode: 6
  - In 32-bit: 4 (capture mode only)
- Count clock division selectable by prescaler (max. frequency of count clock: 10 MHz @ f_{CPU} = 20 MHz)
- Interrupt request sources
  - Compare-match interrupt requests: 6 types
    Perform comparison with sub-channel n capture/compare register and generate the TINTCCExn interrupt upon compare match.
  - Timer counter overflow interrupt requests: 2 types
    The TINTOVE0n (TINTOVE1n) interrupt is generated when the count value of TBASE0n (TBASE1n) becomes FFFFH.
- Capture request
  - Count values of TBASE0n, TBASE1n can be latched using external pin (INTPExn)[Note 1, Note 2]
- PWM output function
  - Control of the outputs of pins TOE1n to TOE4n in the compare mode and PWM output can be performed using the compare match timing of sub-channels 1 to 4 and the zero count signal of the timer counter.
- Timer count operation with external clock input[Note 2]
  - Timer count operation can be performed with the pin TIEn clock input signal.
- Timer count enable operation[Note 3] with external pin input[Note 2]
  - Timer count enable operation can be performed with the TCLREn pin input signal.
- Timer counter clear control[Note 3, Note 4] with external pin input[Note 2]
  - Timer counter clear operation can be performed with the TCLREn pin input signal.

- Up/down count control**Note 3, Note 5** with external pin input**Note 2**
  - Up/down count operation in the compare mode can be controlled with the TCLREn pin input signal.

- Output delay operation
  - A clock-synchronized output delay can be added to the output signal of pins TOE1n to TOE4n.
  - This is effective as an EMI counter measure.

- Input filter
  - An input filter can be inserted at the input stage of external pins (TIEn, INTPE0n to INTPE5n, TCLREn) (refer to cross reference to Input Filter Mode Registers FEMn0 to FEMn5).

Notes: **1.** For the registers used to specify the valid edge for external interrupt requests (INTPE0n through INTPE5n) to timer E, refer to the chapter Timer E input filter mode registers 0 to 5 (FEM0n to FEM5n).

2. The pairs TIEn and INTPE0n, TOE1n and INTPE1n, TOE2n and INTPE2n, TOE3n and INTPE3n, TOE4n and INTPE4n,TCLREn and INTPE5n are each alternate function pins.

3. The count enable operation for the timer counter through external pin input, timer counter clear operation, and up/down count control cannot be performed combined all at the same time.

4. In the case of 32-bit cascade connection, clear operation by external pin input (TCLREn) cannot be performed.

5. Up/down count control using 32-bit cascade connection cannot be performed.

Remarks: **1.** f_{CPU}: Internal system clock

2. x = 0 to 5

3. n = 0 to 2

9.2.3 Basic configuration

The basic configuration is shown below.

Table 9-2: Timer E Configuration List

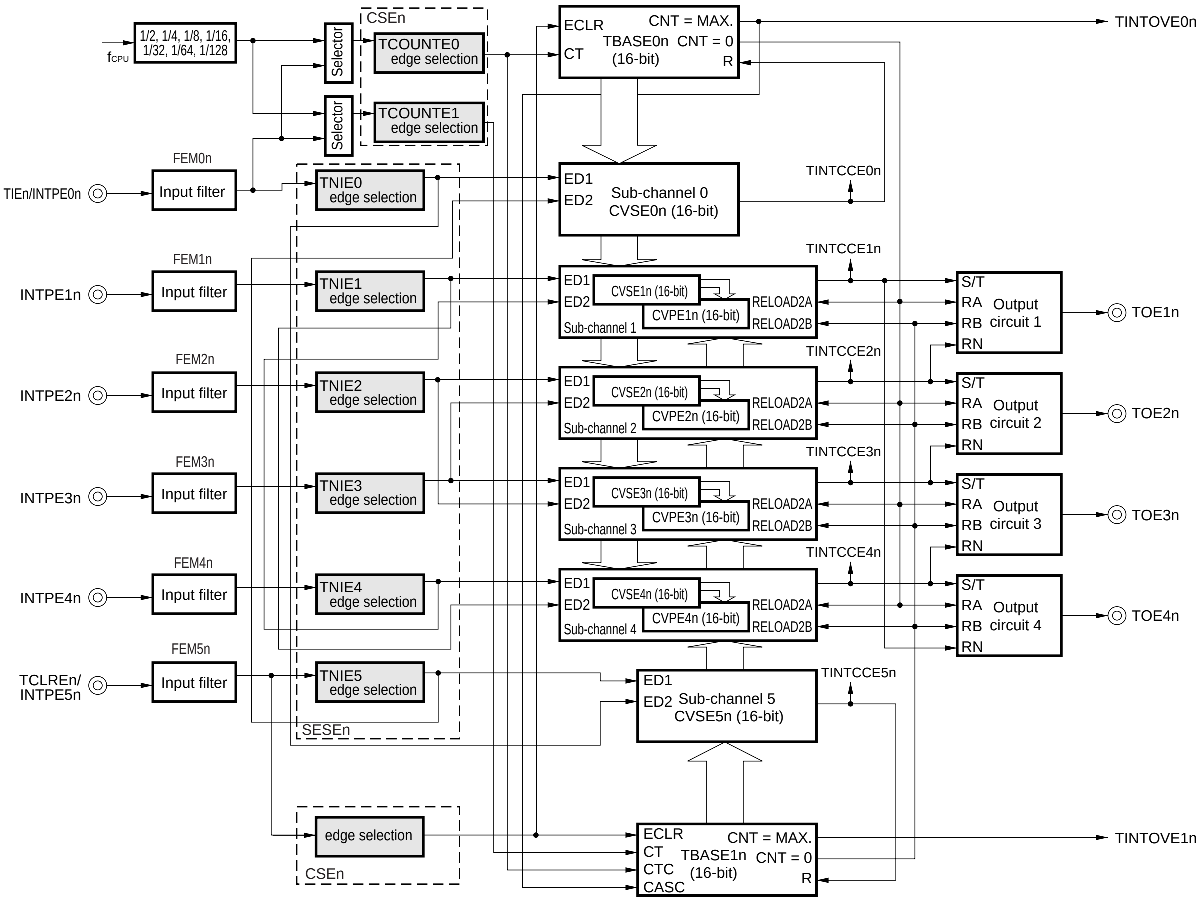
Timer	Count Clock	Register	R/W	Generated Interrupt Signal	Capture Trigger	Timer Output Note 1 S/R	Other Functions
Timer En	f_{CPU} /2, f_{CPU} /4, f_{CPU} /8, f_{CPU} /16, f_{CPU} /32, f_{CPU} /64, f_{CPU} /128, TIEn pin	TBASE0n	–	TINTOVE0n	–	**Note 2**	**Note 3**
		TBASE1n	–	TINTOVE1n	–	**Note 2**	**Note 3**
		CVSE0n	R/W	TINTCCE0n	INTPE0n/ INTPE5n	–	–
		CVSE1n	R/W	–	INTPE1n/ INTPE4n	TOE1n**Note 4**/ TOE4n**Note 4**	Buffer**Note 5**
		CVSE2n	R/W	–	INTPE2n/ INTPE3n	TOE2n**Note 4**/ TOE3n**Note 4**	Buffer**Note 5**
		CVSE3n	R/W	–	INTPE3n/ INTPE2n	TOE3n**Note 4**/ TOE2n**Note 4**	Buffer**Note 5**
		CVSE4n	R/W	–	INTPE4n/ INTPE1n	TOE4n**Note 4**/ TOE1n**Note 4**	Buffer**Note 5**
		CVSE5n	R/W	TINTCCE5n	INTPE5n/ INTPE0n	–	–
		CVPE4n	R	TINTCCE4n	INTPE4n/ INTPE1n	TOE4n/TOE3n	**Note 5**
		CVPE3n	R	TINTCCE3n	INTPE3n/ INTPE2n	TOE3n/TOE2n	**Note 5**
		CVPE2n	R	TINTCCE2n	INTPE2n/ INTPE3n	TOE2n/TOE1n	**Note 5**
		CVPE1n	R	TINTCCE1n	INTPE1n/ INTPE4n	TOE1n/TOE43n	**Note 5**

Notes:
1. Refer to OTMEx1, OTMEx0 bits in (5)“Timer E output control registers 0 to 2 (OCTLE0 to OCTLE2)” on page 271
2. Reset operation by zero count signal is enabled.
3. Cascade operation with TBASE0n and TBASE1n is enabled.
4. Only in buffer-less mode (BFEEx) bit of CMSEmn register = 0)
5. Cascade operation using the CVSExn register and CVPExn register is enabled.

Remarks:
1. f_{CPU}: Internal system clock
2. S/R: Set/Reset
3. m = 12, 34; x = 1 to 4; n = 0 to 2)

Figure 9-7 shows the block diagram of one timer E unit.

Figure 9-7: Block Diagram of Timer E



Remarks: 1. f_{CPU}: Internal system clock
2. n = 0 to 2

Table 9-3: Meaning of Signals in Block Diagram

Signal Name	Meaning
CASC[Note 1]	TBASE1n count signal input in 32-bit mode
CNT	Count value of timer En (CNT = MAX.: Maximum value count signal output of timer En (generated when TBASE0n, TBASE1n = FFFFH), CNT = 0: Zero count signal output of timer (generated when TBASE0n, TBASE1n = 0000H))
CT	TBASE0n, TBASE1n count signal input in 16-bit mode
CTC	TBASE1n count signal input in 32-bit mode
ECLR	External control signal input from TCLREn input
ED1, ED2	Capture event signal input from edge selection circuit
R[Note 2]	Compare match signal input (sub-channel 0/5)
RA	TBASE0n zero count signal input (reset signal of output circuit)
RB	TBASE1n zero count signal input (reset signal of output circuit)
RELOAD2A	TBASE0n zero count signal input (generated when TBASE0n = 0000H)
RELOAD2B	TBASE1n zero count signal input (generated when TBASE1n = 0000H)
RN	Sub-channel x interrupt signal input (reset signal of output circuit)
S/T	Sub-channel x interrupt signal input (set signal of output circuit)
TCOUNTE0, TCOUNTE1	Timer En count enable signal input
TNIEm	Timer En sub-channel m capture event signal input

Notes: **1.** TBASE1n performs count operation when CASC (CNT = MAX. for TBASE0n) is generated and the rising edge of CTC is detected in the 32-bit mode.

2. TBASE0n/TBASE1n clear by sub-channel 0/5 compare match or count direction can be controlled.

Remarks: **1.** m = 0 to 5

2. n = 0 to 2

3. x = 1 to 4

(1) Timer E time base counters 0, 1 registers 0 to 2 (TBASE0n, TBASE1n (n = 0 to 2))

The features of time base counters TBASE0n, TBASE1n are listed below.

- Free-running counter that enables counter clearing by compare match of sub-channel 0 and sub-channel 5
- Can be used as a 32-bit capture timer when TBASE0n and TBASE1n are connected in cascade.
- Up/down control, counter clear, and count operation enable/disable can be controlled with external pin (TCLREn)
- Counter up/down and clear operation control method can be set by software.
- Stop upon occurrence of count value 0 and count operation start/stop can be controlled by software.

Figure 9-8: Timer E Time Base Counter 0 Registers 0 to 2 (TBASE00 to TBASE02)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
TBASE00																	FFFFF670H	0000H
TBASE01																	FFFFF6B0H	0000H
TBASE02																	FFFFF6F0H	0000H

Figure 9-9: Timer E Time Base Counter 1 Registers 0 to 2 (TBASE10 to TBASE12)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
TBASE10																	FFFFF672H	0000H
TBASE11																	FFFFF6B2H	0000H
TBASE12																	FFFFF6F2H	0000H

(2) Timer E sub-channel 0 capture/compare registers 0 to 2 (CVSE00 to CVSE02)

The CVSE0n register is the 16-bit sub-channel 0 capture/compare register of timer TMEn (n = 0 to 2).
In the capture register mode, it captures the TBASE0n count value.
In the compare register mode, it detects match with TBASE0n.
This register can be read/written in 16-bit units.

Figure 9-10: Timer E Sub-Channel 0 Capture/Compare Registers 0 to 2 (CVSE00 to 02)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CVSE00																	FFFFF660H	0000H
CVSE01																	FFFFF6A0H	0000H
CVSE02																	FFFFF6E0H	0000H

(3) Timer E sub-channel x main capture/compare registers 0 to 2 (CVPEx0 to CVPEx2) (x = 1 to 4)

The CVPExn register is a 16-bit sub-channel x main capture/compare register of timer TMEn (x = 1 to 4) (n = 0 to 2).
In the capture register mode, this register captures the value of TBASE1n when the BFEEx bit of the CMSEmn register is zero (m = 12, 34). When the BFEEx bit = 1, this register holds the value of TBASE0n or TBASE1n.
If the capture register mode is selected in the 32-bit mode (value of TB1Ex, TB0Ex bits of CMSEmn register = 11B), this register captures the contents of TBASE1n (higher 16 bits).
This register is read-only in 16-bit units.
In compare mode, this register represents the actual compare value. To write a compare value, the registers CVSExn have to be used. This double register structure refers to the buffered operations in compare mode.

Figure 9-11: Timer E Sub-Channel x Main Capture/Compare Registers 0 to 2 (CVPEx0 to CVPEx2) (x = 1 to 4)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CVPE10																	FFFFF652H	0000H
CVPE11																	FFFFF692H	0000H
CVPE12																	FFFFF6D2H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CVPE20																	FFFFF656H	0000H
CVPE21																	FFFFF696H	0000H
CVPE22																	FFFFF6D6H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CVPE30																	FFFFF65AH	0000H
CVPE31																	FFFFF69AH	0000H
CVPE32																	FFFFF6DAH	

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CVPE40																	FFFFF65EH	0000H
CVPE41																	FFFFF69EH	0000H
CVPE42																	FFFFF6DEH	0000H

(4) Timer E sub-channel x sub capture/compare registers 0 to 2 (CVSEx0 to CVSEx2) (x = 1 to 4)

The CVSExn register is a 16-bit sub channel x sub-capture/compare register of timer TMEn (x = 1 to 4) (n = 0 to 2).

In the compare register mode, this register can be used as a buffer. In the capture register mode, this register captures the value of TBASE0n when the BFEEx bit of the CMSEmn register is cleared (BFEEx bit = 0) (m = 12, 34).

If the capture register mode is selected in the 32-bit mode (TB1Ex, TB0Ex bits of CMSEmn register = 11B), this register captures the contents of TBASE0n (lower 16 bits).

The CVSExn register can be written only in the compare register mode. If this register is written in the capture register mode, the contents written to CVSExn register will be lost.

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

Figure 9-12: Timer E Sub-Channel x Sub Capture/Compare Registers 0 to 2 (CVSEx0 to CVSEx2) (x = 1 to 4)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CVSE10																	FFFFF650H	0000H
CVSE11																	FFFFF690H	0000H
CVSE12																	FFFFF6D0H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CVSE20																	FFFFF654H	0000H
CVSE21																	FFFFF694H	0000H
CVSE22																	FFFFF6D4H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CVSE30																	FFFFF658H	0000H
CVSE31																	FFFFF698H	0000H
CVSE32																	FFFFF6D8H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CVSE40																	FFFFF65CH	0000H
CVSE41																	FFFFF69CH	0000H
CVSE42																	FFFFF6DCH	0000H

(5) Timer E sub-channel 5 capture/compare registers 0 to 2 (CVSE50 to CVSE52)

The CVSE5n register is the 16-bit sub-channel 5 capture/compare register of timer TMEn (n = 0 to 2).
In the capture register mode, it captures the count value of TBASE1n.
In the compare register mode, it detects match with TBASE1n.
This register can be read/written in 16-bit units.

Figure 9-13: Timer E Sub-Channel 5 Capture/Compare Registers (CVSE50 to CVSE52)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CVSE50																	FFFFF662H	0000H
CVSE51																	FFFFF6A2H	0000H
CVSE52																	FFFFF6E2H	0000H

9.2.4 Control Registers

(1) Timer E clock stop registers 0 to 2 (STOPTE0 to STOPTE2)

The STOPTEn register is used to stop the operation clock input to timer TMEn (n = 0 to 2).
This register can be read/written in 16-, 8-, or 1-bit units.

Figure 9-14: Timer E Clock Stop Registers 0 to 2 (STOPTE0 to STOPTE2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
STOPTE0	STFTE0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FFFFF640H	0000H
STOPTE1	STFTE1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FFFFF680H	0000H
STOPTE2	STFTE2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FFFFF6C0H	0000H

Bit Position	Bit Name	Function
15	STFTEn	Stops the operation clock to TMEn. 0: Normal operation 1: Stop operation clock to TMEn

Cautions: 1. Initialize TMEn when the STFTEn bit is cleared (0). TMEn cannot be initialized when the STFTEn bit is set (1).
2. If the STFTEn bit is set (1) after initialization, the initialized state is maintained.

Remark: n = 0 to 2

(2) Timer E count clock/control edge selection registers 0 to 2 (CSE0 to CSE2)

The CSEn register is used to specify the timer TMEn count clock and the control valid edge (n = 0 to 2).
This register can be read/written in 16-, 8-, or 1-bit units.

Figure 9-15: Timer E Count Clock/Control Edge Selection Registers 0 to 2 (CSE0 to CSE2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CSE0	0	0	0	0	TES1E1	TES1E0	TES0E1	TES0E0	CESE1	CESE0	CSE12	CSE11	CSE10	CSE02	CSE01	CSE00	FFFFF642H	0000H
CSE1	0	0	0	0	TES1E1	TES1E0	TES0E1	TES0E0	CESE1	CESE0	CSE12	CSE11	CSE10	CSE02	CSE01	CSE00	FFFFF682H	0000H
CSE2	0	0	0	0	TES1E1	TES1E0	TES0E1	TES0E0	CESE1	CESE0	CSE12	CSE11	CSE10	CSE02	CSE01	CSE00	FFFFF6C2H	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>11, 10, 9, 8</td><td>TESyE1,
TESyE0</td><td>Specifies the valid edge of the corresponding timer base counter TBASEyn count clock signal (TCOUNTEy).
<table>
<tr><th>TESyE1</th><th>TESyE0</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>
<tr><td>7, 6</td><td>CESE1,
CESE0</td><td>Specifies the valid edge of the external clear input signal (TCLREn).
<table>
<tr><th>CESE1</th><th>CESE0</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>Through input</td></tr>
<tr><td>1</td><td>1</td><td>Both rising and falling edges</td></tr>
</table></td></tr>
<tr><td>5 to 3,
2 to 0</td><td>CSEy0,
CSEy1,
CSEy2</td><td>Selects internal count clock of the time base counter TBASEyn.
<table>
<tr><th>CSEy2</th><th>CSEy1</th><th>CSEy0</th><th>Count Clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>$f_{CPU}/2$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$f_{CPU}/4$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$f_{CPU}/8$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$f_{CPU}/16$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$f_{CPU}/32$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>$f_{CPU}/64$</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>$f_{CPU}/128$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Selects input signal from external clock input pin (TIEn) as clock.</td></tr>
</table></td></tr>
</table>

Remark: y = 0, 1
n = 0 to 2

(3) Timer E sub-channel input event edge selection registers 0 to 2 (SESE0 to SESE2)

The SESEn register specifies the valid edge of the external capture signal input (TIExn) for the sub-channel x capture/compare register performing capture (x = 0 to 5, n = 0 to 2).
This register can be read/written in 16-, 8-, or 1-bit units.

Figure 9-16: Timer E Sub-Channel Input Event Edge Selection Register 0 to 2 (SESE0 to SESE2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SESE0	0	0	0	0	IESE51	IESE50	IESE41	IESE40	IESE31	IESE30	IESE21	IESE20	IESE11	IESE10	IESE01	IESE00	FFFFF644H	0000H
SESE1	0	0	0	0	IESE51	IESE50	IESE41	IESE40	IESE31	IESE30	IESE21	IESE20	IESE11	IESE10	IESE01	IESE00	FFFFF684H	0000H
SESE2	0	0	0	0	IESE51	IESE50	IESE41	IESE40	IESE31	IESE30	IESE21	IESE20	IESE11	IESE10	IESE01	IESE00	FFFFF6C4H	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>11 to 0</td><td>IESEx1,
IESEx0</td><td>Specifies the valid edge of external capture signal input (TIExn) for sub-channel x capture/compare register performing capture.
<table>
<tr><th>IESEx1</th><th>IESEx0</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>

Remark: x = 0 to 5
n = 0 to 2

(4) Timer E time base control registers 0 to 2 (TCRE0 to TCRE2)

The TCREn register controls the operation of timer TMEn (n = 0 to 2).
This register can be read/written in 16-, 8-, or 1-bit units.

Figure 9-17: Timer E Time Base Control Registers 0 to 2 (TCRE0 to TCRE2) (1/2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
TCRE0	CASE1	CLRE1	CEE1	ECRE1	ECEE1	OSTE1	UDSE11	UDSE10	0	CLRE0	CEE0	ECRE0	ECEE0	OSTE0	UDSE01	UDSE00	FFFFF646H	0000H
TCRE1	CASE1	CLRE1	CEE1	ECRE1	ECEE1	OSTE1	UDSE11	UDSE10	0	CLRE0	CEE0	ECRE0	ECEE0	OSTE0	UDSE01	UDSE00	FFFFF686H	0000H
TCRE2	CASE1	CLRE1	CEE1	ECRE1	ECEE1	OSTE1	UDSE11	UDSE10	0	CLRE0	CEE0	ECRE0	ECEE0	OSTE0	UDSE01	UDSE00	FFFFF6C6H	0000H

Bit Position	Bit Name	Function
15	CASE1	Specifies 32-bit cascade operation mode for TBASE1n (TBASE1n counts upon overflow of TBASE0n (carry count)). 0: Don't connect in cascade[Note 1] 1: 32-bit cascade operation mode[Notes 2, 3] **Notes:** **1.** TBASE1n counts with CT signal input in the count enabled state **2.** TBASE1n counts with CTC and CASC signal inputs in the count enabled state. **3.** Only the capture register mode can be used for the capture/compare register. **Caution:** **In the 32-bit cascade operation mode (CASE1 bit = 1), only the 32-bit capture function is permitted: set TB1Ex and TB0Ex bits of the CMSEmn registers to 11B (m = 12, 34, x: when m = 12, x = 1, 2, and when m = 34, x = 3, 4).**
14, 6	CLREy	Specifies software clear for TBASEyn. 0: Continue TBASEyn operation 1: Clear (0) TBASEyn count value **Cautions:** **1. Setting the CLREy bit stops and clears the concerned timebase. This bit has to be cleared before starting the count operations by setting the CEEy bit again.** **2. To acknowledge the clear and stop operation it's mandatory to keep the CLREy bit set (1) for at least one timer clock period.** **Remark:** Set/clear operation of CLREy: 1. CLREy = 1 2. CSEy2-0 = 000B 3. CSEy2-0 = "old value" 4. CLREY = 0
13, 5	CEEy	Specifies TBASEyn count operation enable/disable. 0: Stop count operation 1: Enable count operation
12, 4	ECREy	Specifies TBASEyn external clear (TCLREn) operation enable/disable through ECLR signal input. 0: Don't enable TBASEyn external clear (TCLREn) operation 1: Enable TBASEyn external clear (TCLREn) operation **Cautions:** **1. In the 32-bit cascade operation mode (CASE1 bit = 1), TMEn external clear operation does not work.** **2. If the ECLR signal is input when ECREy = 1, TMEn clear operation is performed after 1 internal count clock set with the corresponding CSEy2 to CSEy0 bits of the CSEn register.**

Remark: y = 0, 1
n = 0 to 2

Figure 9-17: Timer E Time Base Control Registers 0 to 2 (TCRE0 to TCRE2) (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>11, 3</td><td>ECEEy</td><td>Specifies TBASEyn count operation enable/disable through ECLR signal input.
0: Don’t enable TBASEyn count operation
1: Enable TBASEyn count operation
<b>Cautions: 1. In the 32-bit cascade operation mode (CASE1 bit = 1), control of the TBASEyn count operation using ECLR signal input is not enabled.</b>
<b>2. When the ECEEy bit = 1, always set the CESE1 and CESE0 bits of the CSEn register to 10B (through input).</b></td></tr>
<tr><td>10, 2</td><td>OSTEy</td><td>Specifies stop mode.
0: Don’t stop TBASEyn count when count value is 0.
1: Stop TBASEyn count when count value is 0.
<b>Caution: When TBASEyn count stop is cancelled when the OSTE1y bit = 1 (TBASEyn count is stopped when the count value is 0), TBASEyn counts up except when the UDSEy1, UDSEy0 bits = 10B. The count direction when the UDSEy1, UDSEy0 bits = 10B is determined by the value of the ECLR signal input.</b></td></tr>
<tr><td>9, 8, 1, 0</td><td>UDSEy1,
UDSEy0</td><td>Specifies TBASEyn up/down count.
<table>
<tr><th>UDSEy1</th><th>UDSEy0</th><th>Count</th></tr>
<tr><td>0</td><td>0</td><td>Perform only up count.
Clear TBASEyn with compare match signal.</td></tr>
<tr><td>0</td><td>1</td><td>Count up after TBASEyn has become “0”, and count down after a compare match occurs for sub-channels 0, 5 (triangular wave up/down count).</td></tr>
<tr><td>1</td><td>0</td><td>Selects up/down count according to the ECLR signal input.
Up count when ECLR = 1
Down count when ECLR = 0</td></tr>
<tr><td>1</td><td>1</td><td>Setting prohibited</td></tr>
</table>
<b>Cautions: 1. In the 32-bit cascade operation mode (CASE1 bit = 1), set the UDSEy1, UDSEy0 bits to 00B.</b>
<b>2. When the UDSEy1, UDSEy0 bits = 10B, be sure to set the CESE1, CESE0 bits of the CSE0n register to 10B (through input).</b>
<b>3. When the UDSEy1, UDSEy0 bits = 10B, compare match between TBASEyn and CVSEmn has no effect on the TBASEyn count operation (m: 0 when y = 0, 5 when y = 1).</b></td></tr>
</table>

Cautions: 1. If there is no external count clock, when this is selected by the prescaler setting, clear operations (external or by software) of the timebase counter does not work.

2. When clearing is performed with the ECLR signal, the TBASEyn counter is cleared with a delay of (1 internal count clock set with bits CSEy2 to CSEy0 of the CSEn register) + 2 base clocks. Therefore, if external clock input is selected as the internal count clock, the counter is not cleared until the external clock (TIEn) is input.

3. The ECREy bit and the ECEEy bit must not be set to 1 at the same time.

4. If either ECEEy bit or ECREy bit is set to 1, up-/down operation with external control via ECLR signal cannot be performed (UDSEy1, UDSEy0 = 10B).

5. When UDSEy1, UDSEy0 = 01B and OSTEy = 1, the counter does not count up when the counter value is “0”. Therefore, when the counter value is “0”, set OSTEy = 0, and after the value of the counter ceases to be “0”, set OSTEy = 1.

6. If there is no external count clock, when this is selected by the prescaler setting, clear operations (external or by software) of the timebase counter does not work.

Remark: y = 0, 1
n = 0 to 2

(5) Timer E output control registers 0 to 2 (OCTLE0 to OCTLE2)

The OCTLEn register controls timer output from the TOExn pin (x = 1 to 4, n = 0 to 2).
This register can be read/written in 16-, 8-, or 1-bit units.

Figure 9-18: Timer E Output Control Registers 0 to 2 (OCTLE0 to OCTLE2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
OCTLE0	SWFE4	ALVE4	OTME41	OTME40	SWFE3	ALVE3	OTME31	OTME30	SWFE2	ALVE2	OTME21	OTME20	SWFE1	ALVE1	OTME11	OTME10	FFFFF648H	0000H
OCTLE1	SWFE4	ALVE4	OTME41	OTME40	SWFE3	ALVE3	OTME31	OTME30	SWFE2	ALVE2	OTME21	OTME20	SWFE1	ALVE1	OTME11	OTME10	FFFFF688H	0000H
OCTLE2	SWFE4	ALVE4	OTME41	OTME40	SWFE3	ALVE3	OTME31	OTME30	SWFE2	ALVE2	OTME21	OTME20	SWFE1	ALVE1	OTME11	OTME10	FFFFF6C8H	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15, 11, 7, 3</td><td>SWFEx</td><td>Fixes the TOExn pin output level according to the setting of ALVEx bit.
0: Don’t fix output level.
1: When ALVE x = 0, fix output level to low level.
When ALVEx = 1, fix output level to high level.</td></tr>
<tr><td>14, 10, 6, 2</td><td>ALVEx</td><td>Specifies the active level of the TOExn pin output.
0: Active level is high level
1: Active level is low level</td></tr>
<tr><td>13, 12, 9, 8, 5, 4, 1, 0</td><td>OTMEx1, OTMEx0</td><td>Specifies toggle mode.
<table>
<tr><th>OTMEx1</th><th>OTMEx0</th><th>Toggle Mode</th></tr>
<tr><td>0</td><td>0</td><td>Toggle mode 0:
Reverse output level of TOExn output every time a sub-channel x compare match occurs.</td></tr>
<tr><td>0</td><td>1</td><td>Toggle mode 1:
Upon sub-channel x compare match, set TOExn output to active level, and when TBASE0n is cleared (0), set TOExn output to inactive level.</td></tr>
<tr><td>1</td><td>0</td><td>Toggle mode 2:
Upon sub-channel x compare match, set TOExn output to active level, and when TBASE1n is cleared (0), set TOExn output to inactive level.</td></tr>
<tr><td>1</td><td>1</td><td>Toggle mode 3:
Upon sub-channel x compare match, set TOExn output to active level, and upon sub-channel [x + 1] compare match, set TOxn output to inactive level (when x = 4, [x + 1] becomes 1).</td></tr>
</table>
<b>Cautions: 1. When the OTMEx1, OTMEx0 bits = 11B (toggle mode 3), and if the same output delay operation settings are made by setting bits ODLEx2 to ODLEx0 of the ODELEn register, two outputs change simultaneously upon 1 sub-channel x compare match.</b>
<b>2. If two or more signals are input simultaneously to the same output circuit, S/T signal input has a higher priority than RA, RB, and RN signal inputs.</b></td></tr>
</table>

Remark: x = 1 to 4
n = 0 to 2

(6) Timer E sub-channel 0, 5 capture/compare control registers 0 to 2 (CMSE050 to CMSE052)

The CMSE05n register controls timer TMEn sub-channel 0 capture/compare register (CVSE0n) and timer TMEn sub-channel 5 capture/compare register (CVSE5n) (n = 0 to 2).
This register can be read/written in 16-bit units.

Figure 9-19: Timer E Sub-Channel 0, 5 Capture/Compare Control Registers 0 to 2 (CMSE050 to CMSE052)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CMSE050	0	0	EEVE5	0	LNKE5	CCSE5	0	0	0	0	EEVE0	0	LNKE0	CCSE0	0	0	FFFFF64AH	0000H
CMSE051	0	0	EEVE5	0	LNKE5	CCSE5	0	0	0	0	EEVE0	0	LNKE0	CCSE0	0	0	FFFFF68AH	0000H
CMSE052	0	0	EEVE5	0	LNKE5	CCSE5	0	0	0	0	EEVE0	0	LNKE0	CCSE0	0	0	FFFFF6CAH	0000H

Bit Position	Bit Name	Function
13, 5	EEVEx	Enables/disables capture event detection by sub-channel x capture/compare register. 0: Don’t detect events 1: Detect events
11, 3	LNKEx	Specifies capture event signal input from edge selection to ED1 or ED2 of timer TMEn, sub-channel x. 0: In capture register mode, select ED1 signal input. In compare register mode, LNKEx bit has no influence. 1: In capture register mode, select ED2 signal input. In compare register mode, LNKEx bit has no influence.
10, 2	CCSEx	Selects capture/compare register operation mode of timer TMEn, sub-channel x. 0: Operate in capture register mode. The TBASE0n and TBASE1n count statuses can be read with sub-channel 0 and sub-channel 5, respectively. 1: Operate in compare register mode. TBASE0n, TBASE1n is cleared upon detection of match between sub-channel x and TBASE0n, TBASE1n.

Remark: x = 0, 5
n = 0 to 2

(7) Timer E sub-channel 1, 2 capture/compare control register 0 to 2 (CMSE120 to CMSE122)

The CMSE12n register controls the timer TMEn sub-channel x sub capture/compare register (CVSExn) and the timer TMEn sub-channel x main capture/compare register (CVPExn) (x = 1, 2) (n = 0 to 2).
This register can be read/written in 16-bit units.

Figure 9-20: Timer E Sub-Channel 1, 2 Capture/Compare Control Registers 0 to 2 (CMSE120 to CMSE122) (1/2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CMSE120	0	0	EEVE2	BFEE2	LNKE2	CCSE2	TB1E2	TB0E2	0	0	EEVE1	BFEE1	LNKE1	CCSE1	TB1E1	TB0E1	FFFFF64CH	0000H
CMSE121	0	0	EEVE2	BFEE2	LNKE2	CCSE2	TB1E2	TB0E2	0	0	EEVE1	BFEE1	LNKE1	CCSE1	TB1E1	TB0E1	FFFFF68CH	0000H
CMSE122	0	0	EEVE2	BFEE2	LNKE2	CCSE2	TB1E2	TB0E2	0	0	EEVE1	BFEE1	LNKE1	CCSE1	TB1E1	TB0E1	FFFFF6CCH	0000H

Bit Position	Bit Name	Function
13, 5	EEVEx	Enables/disables capture event detection by sub-channel x capture/compare register. 0: Don’t detect events 1: Detect events
12, 4	BFEEx	Specifies the buffer operation of sub-channel x sub capture/compare register (CVSExn). 0: Don’t use sub-channel x sub capture/compare register (CVSExn) as buffer. 1: Use sub-channel x sub capture/compare register (CVSExn) as buffer. **Remarks:** **1.** The operations in the capture register mode and compare register mode when the sub-channel x sub capture/compare register (CVSExn) is not used as a buffer are shown below. (BFEEx = 0) • In capture register mode: The CPU can read both the master register (CVPExn) and slave register (CVSExn). The next event is ignored until the CPU finishes reading the master register. TME0n capture is performed by the slave register, and TME1n capture is performed by the master register. • In compare register mode: The CPU writes to the slave register (CVSExn), and immediately after, the same contents as those of the slave register are written to the master register (CVPExn). **2.** The operations in the capture register mode and compare register mode when the sub-channel x sub capture/compare register (CVSExn) is used as a buffer are shown below. (BFEEx = 1) • In capture register mode: When the CPU reads the master register (CVPExn), the master register updates the value held by the slave register (CVSExn) immediately and once only, after the CPU read operation. When a capture event occurs, the timer counter value at that time is always saved in the slave register. • In compare register mode: The CPU writes to the slave register (CVSExn) and these contents are transferred to the master register (CVPExn) specified by the corresponding LNKEx bit.

Remark: x = 1, 2
n = 0 to 2

Figure 9-20: Timer E Sub-Channel 1, 2 Capture/Compare Control Registers 0 to 2 (CMSE120 to CMSE122) (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>11, 3</td><td>LNKEx</td><td>Selects capture event signal input from edge selection and specifies transfer operation in compare register mode.
0: Select ED1 signal input in capture register mode.
In the compare register mode, the data of the CVSExn register is transferred to the CVPExn register upon occurrence of TBASE0n, TBASE1n[Note] compare match.
1: Select ED2 signal input in capture register mode.
In the compare register mode, the data of the CVSExn register is transferred to the CVPExn register when the TBASE0n, TBASE1n[Note] count value becomes zero.
Note: TBASE0n, TBASE1n = time base counter specified by the corresponding TB1Ex, TB0Ex bits.</td></tr>
<tr><td>10, 2</td><td>CCSEx</td><td>Selects capture/compare register operation mode.
0: Capture register mode
1: Compare register mode</td></tr>
<tr><td>9, 8, 1, 0</td><td>TB1Ex
TB0Ex</td><td>Specifies time base counter of sub-channel x.
<table>
<tr><th>TB1Ex</th><th>TB0Ex</th><th>Time Base Counter of Sub-channel x</th></tr>
<tr><td>0</td><td>0</td><td>Don’t use sub-channel x.</td></tr>
<tr><td>0</td><td>1</td><td>Set TBASE0n to sub-channel x.</td></tr>
<tr><td>1</td><td>0</td><td>Set TBASE1n to sub-channel x.</td></tr>
<tr><td>1</td><td>1</td><td>32-bit mode[Note] (Select both TBASE0n and TBASE1n.)</td></tr>
</table>
Note: In the 32-bit mode, setting of the BFEEx bit is ignored, and the CVSExn register cannot be used as a buffer.</td></tr>
</table>

Remark: x = 1, 2
n = 0 to 2

(8) Timer E sub-channel 3, 4 capture/compare control registers 0 to 2 (CMSE340 to CMSE342)

The CMSE34n register controls the timer TMEn sub-channel x sub capture/compare register (CVSExn) and the timer TMEn sub-channel x main capture/compare register (CVPExn) (x = 3, 4) (n = 0 to 2).
This register can be read/written in 16-bit units.

Figure 9-21: Timer E Sub-Channel 3, 4 Capture/Compare Control Registers 0 to 2 (CMSE340 to CMSE342) (1/2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CMSE340	0	0	EEVE4	BFEE4	LNKE4	CCSE4	TB1E4	TB0E4	0	0	EEVE3	BFEE3	LNKE3	CCSE3	TB1E3	TB0E3	FFFFF64EH	0000H
CMSE341	0	0	EEVE4	BFEE4	LNKE4	CCSE4	TB1E4	TB0E4	0	0	EEVE3	BFEE3	LNKE3	CCSE3	TB1E3	TB0E3	FFFFF68EH	0000H
CMSE342	0	0	EEVE4	BFEE4	LNKE4	CCSE4	TB1E4	TB0E4	0	0	EEVE3	BFEE3	LNKE3	CCSE3	TB1E3	TB0E3	FFFFF6CEH	0000H

Bit Position	Bit Name	Function
13, 5	EEVEx	Enables/disables capture event detection by sub-channel x capture/compare register. 0: Don’t detect events 1: Detect events
12, 4	BFEEx	Specifies the buffer operation of sub-channel x sub capture/compare register (CVSExn). 0: Don’t use sub-channel x sub capture/compare register (CVSExn) as buffer. 1: Use sub-channel x sub capture/compare register (CVSExn) as buffer. **Remarks:** **1.** The operations in the capture register mode and compare register mode when the sub-channel x sub capture/compare register (CVSExn) is not used as a buffer are shown below. (BFEEx = 0) • In capture register mode: The CPU can read both the master register (CVPExn) and slave register (CVSExn). The next event is ignored until the CPU finishes reading the master register. TME0n capture is performed by the slave register, and TME1n capture is performed by the master register. • In compare register mode: The CPU writes to the slave register (CVSExn), and immediately after, the same contents as those of the slave register are written to the master register (CVPExn). **2.** The operations in the capture register mode and compare register mode when the sub-channel x sub capture/compare register (CVSExn) is used as a buffer are shown below. (BFEEx = 1) • In capture register mode: When the CPU reads the master register (CVPExn), the master register updates the value held by the slave register (CVSExn) immediately and once only, after the CPU read operation. When a capture event occurs, the timer counter value at that time is always saved in the slave register. • In compare register mode: The CPU writes to the slave register (CVSExn) and these contents are transferred to the master register (CVPExn) specified by the corresponding LNKEx bit.

Remark: x = 3, 4
n = 0 to 2

Figure 9-21: Timer E Sub-Channel 3, 4 Capture/Compare Control Registers 0 to 2 (CMSE340 to CMSE342) (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>11, 3</td><td>LNKEx</td><td>Selects capture event signal input from edge selection and specifies transfer operation in compare register mode.
0: Select ED1 signal input in capture register mode.
In the compare register mode, the data of the CVSExn register is transferred to the CVPExn register upon occurrence of TBASE0n, TBASE1n[Note] compare match.
1: Select ED2 signal input in capture register mode.
In the compare register mode, the data of the CVSExn register is transferred to the CVPExn register when the TBASE0n, TBASE1n[Note] count value becomes zero.
Note: TBASE0n, TBASE1n = time base counter specified by the corresponding TB1Ex, TB0Ex bits.</td></tr>
<tr><td>10, 2</td><td>CCSEx</td><td>Selects capture/compare register operation mode.
0: Capture register mode
1: Compare register mode</td></tr>
<tr><td>9, 8, 1, 0</td><td>TB1Ex,
TB0Ex</td><td>Specifies time base counter of sub-channel x.
<table>
<tr><th>TB1Ex</th><th>TB0Ex</th><th>Time Base Counter of Sub-channel x</th></tr>
<tr><td>0</td><td>0</td><td>Don’t use sub-channel x.</td></tr>
<tr><td>0</td><td>1</td><td>Set TBASE0n to sub-channel x.</td></tr>
<tr><td>1</td><td>0</td><td>Set TBASE1n to sub-channel x.</td></tr>
<tr><td>1</td><td>1</td><td>32-bit mode[Note] (Select both TBASE0n and TBASE1n.)</td></tr>
</table>
Note: In the 32-bit mode, setting of the BFEEx bit is ignored, and the CVSExn register cannot be used as a buffer.</td></tr>
</table>

Remark: x = 3, 4
n = 0 to 2

(9) Timer E time base status registers 0 to 2 (TBSTATE0 to TBSTATE2)

The TBSTATEn register indicates the status of the time base counter TBASEyn (y = 0, 1) (n = 0 to 2).

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

Figure 9-22: Timer E Time Base Status Register (TBSTATE0 to TBSTATE2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
TBSTATE0	0	0	0	0	OVFE1	ECFE1	RSFE1	UDFE1	0	0	0	0	OVFE0	ECFE0	RSFE0	UDFE0	FFFFF664H	0000H
TBSTATE1	0	0	0	0	OVFE1	ECFE1	RSFE1	UDFE1	0	0	0	0	OVFE0	ECFE0	RSFE0	UDFE0	FFFFF6A4H	0000H
TBSTATE2	0	0	0	0	OVFE1	ECFE1	RSFE1	UDFE1	0	0	0	0	OVFE0	ECFE0	RSFE0	UDFE0	FFFFF6E4H	0000H

Bit Position	Bit Name	Function
11, 3	OVFEy	Indicates TBASEyn overflow status. 0: No overflow 1: Overflow
10, 2	ECFEy	Indicates the ECLR signal input status. 0: Low level 1: High level
9, 1	RSFEy	Indicates the TBASEyn count status. 0: TBASEyn is not counting. 1: TBASEyn is counting (either up or down)
8, 0	UDFEy	Indicates the TBASEyn up/down count status. 0: TBASEyn is in the down count mode. 1: TBASEyn is in the up count mode.

Cautions: 1. The OVFEy bits are cleared, if a "1" is written into these bits.
2. The ECFEy, RSFEy, and UDFEy bits are read-only bits.

Remark: y = 0, 1
n = 0 to 2

(10) Timer E capture/compare status registers 0 to 2 (CCSTATE0 to CCSTATE2)

The CCSTATEn register indicates the status of the timer TMEn sub-channel x sub capture/compare register (CVSExn) and the timer TMEn sub-channel x main capture/compare register (CVPExn) (x = 1 to 4) (n = 0 to 2).
This register can be read/written in 16-, 8-bit units.

Figure 9-23: Timer E Capture/Compare Status Registers 0 to 2 (CCSTATE0 to CCSTATE2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CCSTATE0	0	CEFE4	BFFE41	BFFE40	0	CEFE3	BFFE31	BFFE30	0	CEFE2	BFFE21	BFFE20	0	CEFE1	BFFE11	BFFE10	FFFFF666H	0000H
CCSTATE1	0	CEFE4	BFFE41	BFFE40	0	CEFE3	BFFE31	BFFE30	0	CEFE2	BFFE21	BFFE20	0	CEFE1	BFFE11	BFFE10	FFFFF6A6H	0000H
CCSTATE2	0	CEFE4	BFFE41	BFFE40	0	CEFE3	BFFE31	BFFE30	0	CEFE2	BFFE21	BFFE20	0	CEFE1	BFFE11	BFFE10	FFFFF6E6H	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>14, 10, 6, 2</td><td>CEFEx</td><td>Indicates the capture/compare event occurrence status.
0: In capture register mode: No capture operation has occurred.
In compare register mode: No compare match has occurred.
1: In capture register mode: At least one capture operation has occurred.
In compare register mode: At least one compare match has occurred.

<b>Caution: The CEFEx bit can be cleared (0) by performing write access to the CCSTATEn register while no capture operation or compare match occurs. Bit manipulation instructions are not allowed.</b></td></tr>
<tr><td>13, 12, 9, 8, 5, 4, 1, 0</td><td>BFFEx1, BFFEx0</td><td>Indicates the capture buffer status.
<table>
<tr><th>BFFEx1</th><th>BFFEx0</th><th>Capture Buffer Status</th></tr>
<tr><td>0</td><td>0</td><td>No value in buffer</td></tr>
<tr><td>0</td><td>1</td><td>Sub-channel x master register (CVPExn) contains a capture value. Slave register (CVSExn) does not contain a value.</td></tr>
<tr><td>1</td><td>0</td><td>Both sub-channel x master register (CVPExn) and slave register (CVSExn) contain a capture value.</td></tr>
<tr><td>1</td><td>1</td><td>Unused</td></tr>
</table>
<b>Caution: The BFFEx1 and BFFEx0 bits return a value only when sub-channel x sub capture/compare register (CVSExn) buffer operation (bit BFEEx of CMSEmn register = 1) is selected or when capture register mode (bit CCSEx of CMSEmn register = 0) is selected. Zero is read when the compare register mode (CCSEx bit = 1) is selected.</b></td></tr>
</table>

Caution: The BFFEx1 and BFFEx0 bits are read-only bits.

Remark: x = 1 to 4
n = 0 to 2
m = 12, 34

(11) Timer E output delay registers 0 to 2 (ODELE0 to ODELE2)

The ODELEn register sets the output delay operation synchronized with the clock to the TOExn pin's output delay circuit (x = 1 to 4) (n = 0 to 2).
This register can be read/written in 16-, 8-, or 1-bit units.

Figure 9-24: Timer E Output Delay Registers 0 to 2 (ODELE0 to ODELE2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ODELE0	0	ODLE42	ODLE41	ODLE40	0	ODLE32	ODLE31	ODLE30	0	ODLE22	ODLE21	ODLE20	0	ODLE12	ODLE11	ODLE10	FFFFF668H	0000H
ODELE1	0	ODLE42	ODLE41	ODLE40	0	ODLE32	ODLE31	ODLE30	0	ODLE22	ODLE21	ODLE20	0	ODLE12	ODLE11	ODLE10	FFFFF6A8H	0000H
ODELE2	0	ODLE42	ODLE41	ODLE40	0	ODLE32	ODLE31	ODLE30	0	ODLE22	ODLE21	ODLE20	0	ODLE12	ODLE11	ODLE10	FFFFF6E8H	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>14 to 12,
10 to 8,
6 to 4,
2 to 0</td><td>ODLEx2,
ODLEx1,
ODLEx0</td><td>Specifies output delay operation of TOExn
<table>
<tr><th>ODLEx2</th><th>ODLEx1</th><th>ODLEx0</th><th>Set Output Delay Operation</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Don’t perform output delay operation.</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Set output delay of 1 system clock.</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Set output delay of 2 system clocks.</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Set output delay of 3 system clocks.</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Set output delay of 4 system clocks.</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Set output delay of 5 system clocks.</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Set output delay of 6 system clocks.</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Set output delay of 7 system clocks.</td></tr>
</table>
<b>Remark:</b> The ODLEx2 to ODLEx0 bits are used for EMI counter measures.</td></tr>
</table>

Remark: x = 1 to 4
n = 0 to 2

(12) Timer E software event capture registers 0 to 2 (CSCE0 to CECE2)

The CSCE0n register sets capture operation by software in the capture register mode (n = 0 to 2). This register can be read/written in 16-bit units.

Figure 9-25: Timer E Software Event Capture Registers 0 to 2 (CSCE0 to CSCE2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CSCE0	0	0	0	0	0	0	0	0	0	0	SEVE5	SEVE4	SEVE3	SEVE2	SEVE1	SEVE0	FFFFF66AH	0000H
CSCE1	0	0	0	0	0	0	0	0	0	0	SEVE5	SEVE4	SEVE3	SEVE2	SEVE1	SEVE0	FFFFF6AAH	0000H
CSCE2	0	0	0	0	0	0	0	0	0	0	SEVE5	SEVE4	SEVE3	SEVE2	SEVE1	SEVE0	FFFFF6EAH	0000H

Bit Position	Bit Name	Function
5 to 0	SEVEx	Specifies capture operation by software in capture register mode of sub-channel x of the corresponding timer TMEn. 0: Continue normal operation. 1: Perform capture operation.

Cautions: 1. The SEVEx bit ignores the settings of the EEVEx and the LNKEx bits of the CMSEmn register.

2. The SEVEx bit is automatically cleared (0) at the end of an event.

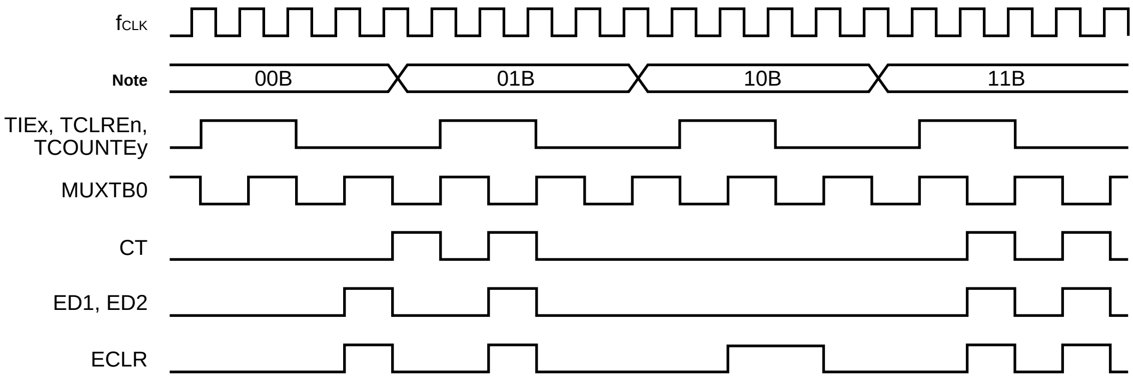
Remark: x = 0 to 5
n = 0 to 2
m = 12, 34, 05

9.2.5 Operation

(1) Edge detection

The edge detection timing is shown below.

Figure 9-26: Edge Detection Timing



Note: The set values of the TESyE1, TESyE0 bits and the CESE1, CESE0 bits of the CSEn register, and the IESEx1, IESEx0 bits of the SESEn register are shown.

Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock

2. CT: TBASEyn count signal input in the 16-bit mode
ECLR: External control signal input from TCLREn pin input
ED1, ED2: Capture event signal input from edge selection circuit
MUXTB0: TBASE0n multiplex signal
TCOUNTEy: Timer En count enable signal input of time base TBASEyn
TIEx: Timer En sub-channel x capture event signal pin input
TCLREn: Timer En clear signal pin input

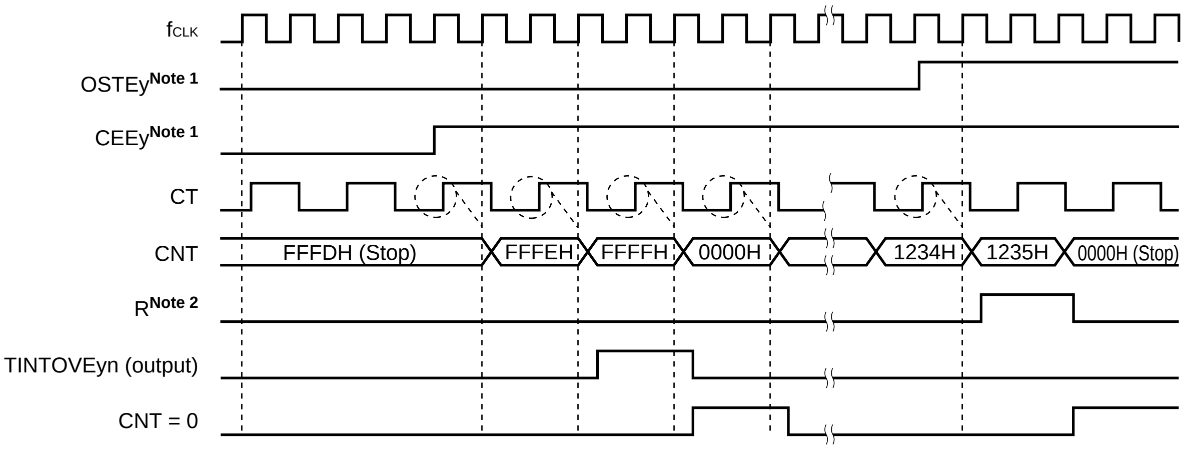
3. x = 0 to 5
y = 0, 1
n = 0 to 2

(2) Basic operation of timer E

Figures 9-27 to 9-30 show the basic operation of timer E.

Figure 9-27: Timer E Up Count Timing

(When TCREn Register’s UDSEy1, UDSEy0 Bits = 00B, ECEEy Bit = 0, ECREy Bit = 0, CLREy Bit = 0, CASE1 Bit = 0)

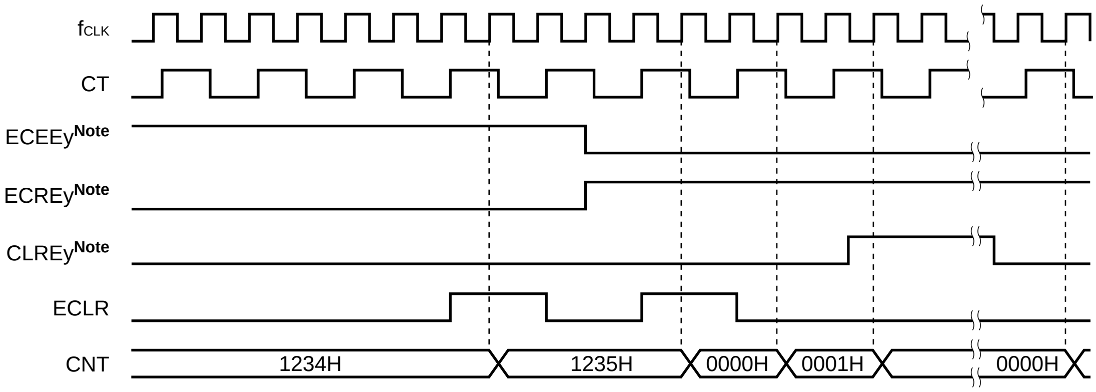


Notes: **1.** Bits OSTEy, CEEy of TCREn register
2. Controls TBASE0n/TBASE1n clear by sub-channel 0/5 compare match or count direction.

Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock
2. CNT: Count value of time base TBASEyn
CT: TBASEyn count signal input in the 16-bit mode
R: Compare match signal input (sub-channel 0/5)
3. y = 0, 1
n = 0 to 2

Figure 9-28: External Control Timing of Timer E

(When TCREn Register's UDSEy1, UDSEy0 Bits = 00B, OSTEy Bit = 0, CEEy Bit = 1, CASE1 Bit = 0)

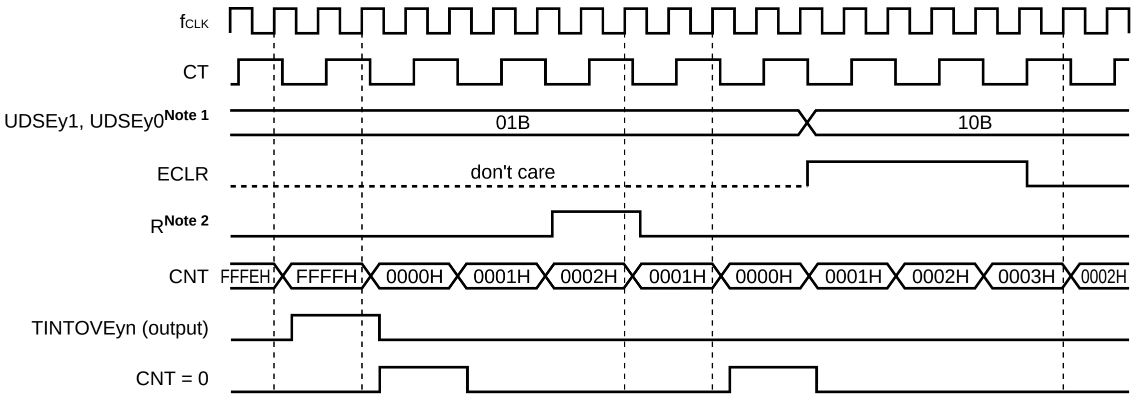


Note: Bits ECEEy, ECREy, CLREy of TCREn register.

Remarks:	**1.** f_{CLK} = f_{CPU}:	Base clock
	2. CNT:	Count value of time base TBASEyn
	CT:	TBASEyn count signal input in the 16-bit mode
	ECLR:	External control signal input from TCLREn pin input
	3. y = 0, 1	
	n = 0 to 2	

Figure 9-29: Operation in Timer E Up/Down Count Mode

(When TCREn Register’s ECEEy bit = 0, ECREy Bit = 0, CLREy Bit = 0, OSTEy Bit = 0, CEEy Bit = 1, CASE1 Bit = 0)

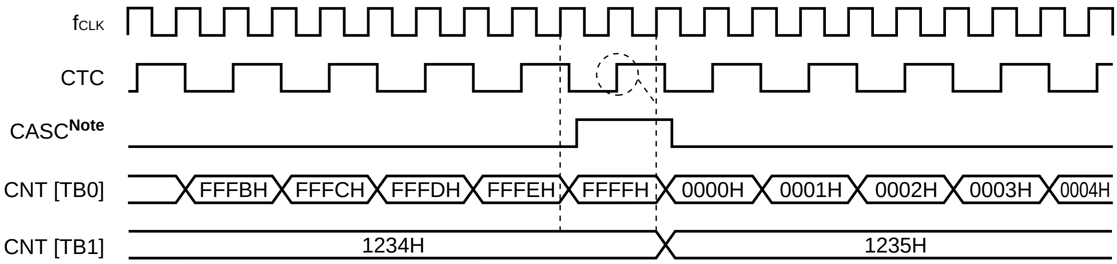


Notes: **1.** UDSEy1, UDSEy0 bits of TCREn register
2. Controls TBASE0n/TBASE1n clear by sub-channel 0/5 compare match or count direction.

Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock
2. CNT: Count value of time base TBASEyn
CT: TBASEyn count signal input in 16-bit mode
ECLR: External control signal input from TCLREn pin input
R: Compare match signal input (sub-channel 0/5)
3. y = 0, 1
n = 0 to 2

Figure 9-30: Timer E Timing in 32-Bit Cascade Operation Mode

(When TCREn Register’s UDSEy1, UDSEy0 Bits = 00B, ECEEy Bit = 0, ECREy Bit = 0, CLREy Bit = 0, OSTEy Bit = 0, CEEy Bit = 1, CASE1 Bit = 1)



Note: If, in the 32-bit mode, CASC (CNT = MAX for TBASE0n) is input to TBASE1n and the CTC rising edge is detected, TBASE1n performs count operation.

Remarks: 1. $f_{CLK} = f_{CPU}$: Base clock

2. CASC: TBASE1n count signal input in 32-bit mode
CNT (TBy): Count value of time base TBASEyn
CTC: TBASE1n count signal input in 32-bit mode

3. y = 0, 1
n = 0 to 2

(3) Operation of capture/compare register (sub-channels 1 to 4)

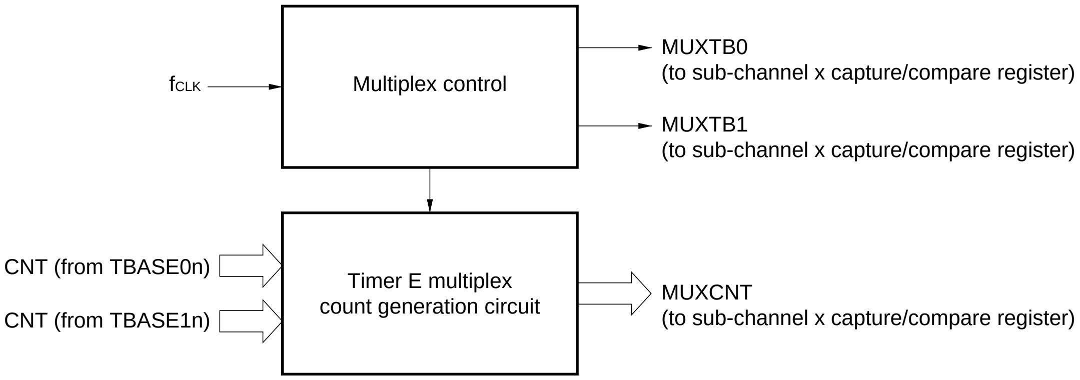
Sub-channels 1 to 4 receive the count value of the timer TMEn multiplex count generation circuit (n = 0 to 2).

The multiplex count generation circuit is an internal unit of the time base counters TBASE0n, TBASE1n that supplies the multiplex count value MUXCNT to sub-channels 1 to 4. The count value of TBASE0n is output to sub-channels 1 to 4 at the rising edge of MUXTB0, and the count value of TBASE1n is output to sub-channels 1 to 4 at the rising edge of MUXTB1.

Figure 9-31 shows the block diagram of the timer TMEn multiplex count generation circuit, and Figure 9-32 shows the multiplex count timing.

Figures 9-33 to 9-38 show the operation of the capture/compare register (sub-channels 1 to 4).

Figure 9-31: Block Diagram of Timer E Multiplex Count Generation Circuit

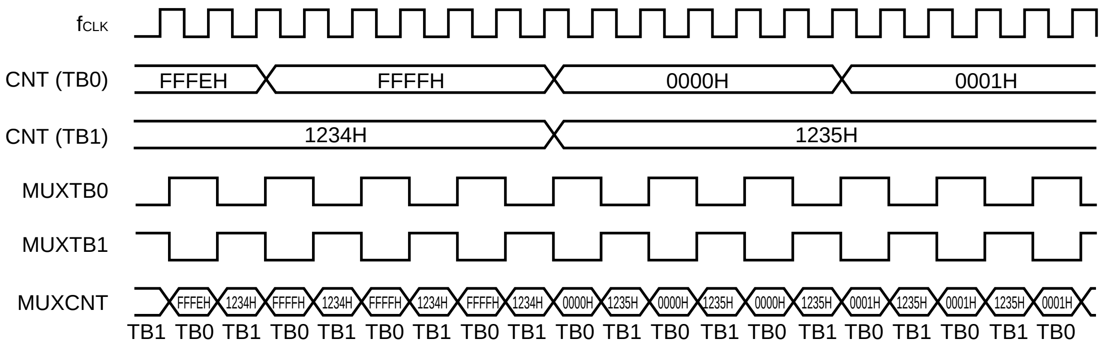

fCLK
Multiplex control
MUXTB0
(to sub-channel x capture/compare register)
MUXTB1
(to sub-channel x capture/compare register)
CNT (from TBASE0n)
CNT (from TBASE1n)
Timer E multiplex
count generation circuit
MUXCNT
(to sub-channel x capture/compare register)

Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock

2. CNT: Count value of time base TBASE0n/TBASE1n
MUXTB0: Multiplex signal of TBASE0n
MUXTB1: Multiplex signal of TBASE1n
MUXCNT: Count value to sub-channel x

3. x = 1 to 4
n = 0 to 2

Figure 9-32: Timer E Multiplex Count Timing



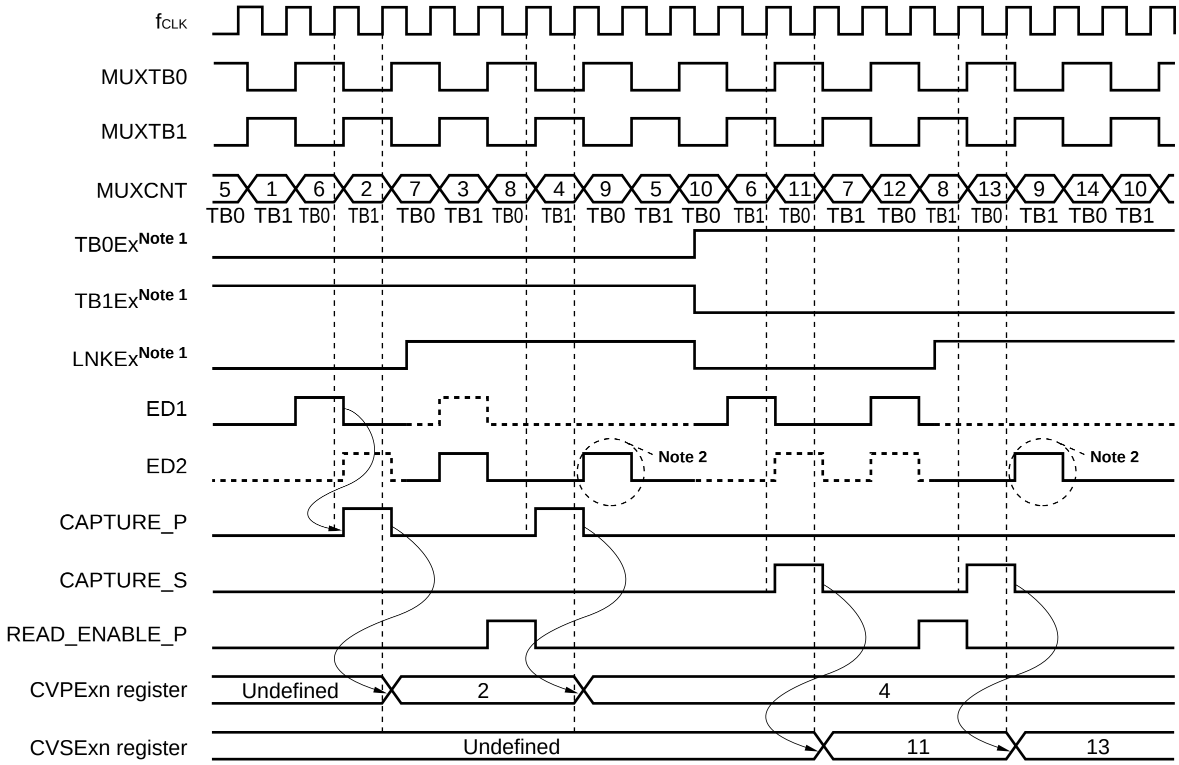
Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock

2. CNT (TB0): Count value of time base TBASE0n
CNT (TB1): Count value of time base TBASE1n
MUXTB0: Multiplex signal of TBASE0n
MUXTB1: Multiplex signal of TBASE1n
MUXCNT: Count value to sub-channel x
TB0, TB1: Time base TBASE0n, TBASE1n

3. x = 1 to 4
n = 0 to 2

Figure 9-33: Timer E Capture Operation: 16-Bit Buffer-Less Mode

(When Operation Is Delayed Through Setting of LNKEx Bit of CMSEmn Register, and CMSEmn Register's CCSEx Bit = 0, BFEEx Bit = 0, EEVEx Bit = 1, and CSCEn Register's SEVEx Bit = 0)



Notes: **1.** Bits TB0Ex, TB1Ex of CMSEmn register
2. If an event occurs in this timing, it is ignored.

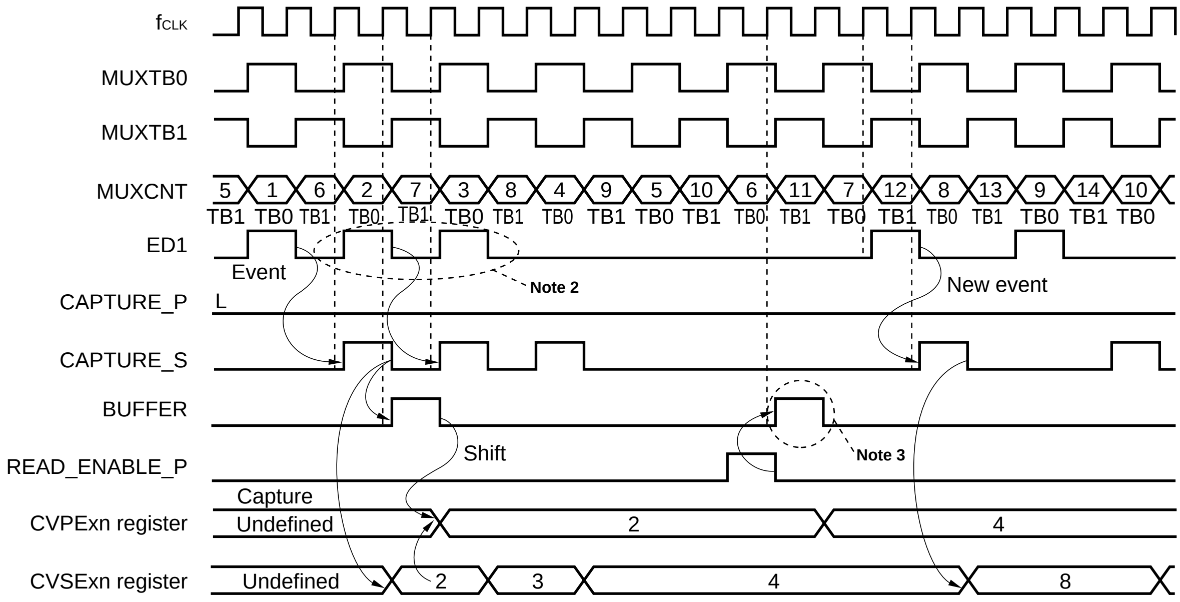
Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock

2. CAPTURE_P: Capture trigger signal of main capture register
CAPTURE_S: Capture trigger signal of sub capture register
ED1, ED2: Capture event signal input from edge selection circuit
MUXCNT: Count value to sub-channel x
MUXTB0: Multiplex signal of TBASE0n
MUXTB1: Multiplex signal of TBASE1n
READ_ENABLE_P: Read timing for CVPExn register
TB0, TB1: Time base TBASE0n, TBASE1n

3. x = 1 to 4
m = 12, when x = 1 or 2
m = 34, when x = 3 or 4
n = 0 to 2

Figure 9-34: Timer E Capture Operation: Mode with 16-Bit Buffer **Note 1**

(When CMSEmn Register's TB1Ex Bit = 0, TB0Ex Bit = 1, CCSEx Bit = 0, LNKEx Bit = 0, BFEEx Bit = 1, EEVEx Bit = 1, and CSCEn Register's SEVEx Bit = 0)



Notes:

1. To operate TBASE0n, TBASE1n in this mode, perform capture at least twice at the start of operation and read the CVPExn register. Also, read the CVPExn register after performing capture at least once.

2. Write operation to the CVPExn register is not performed at these signal inputs because the CVSExn register operates as a buffer.

3. After this timing, write operation from the CVSExn register to the CVPExn register is enabled.

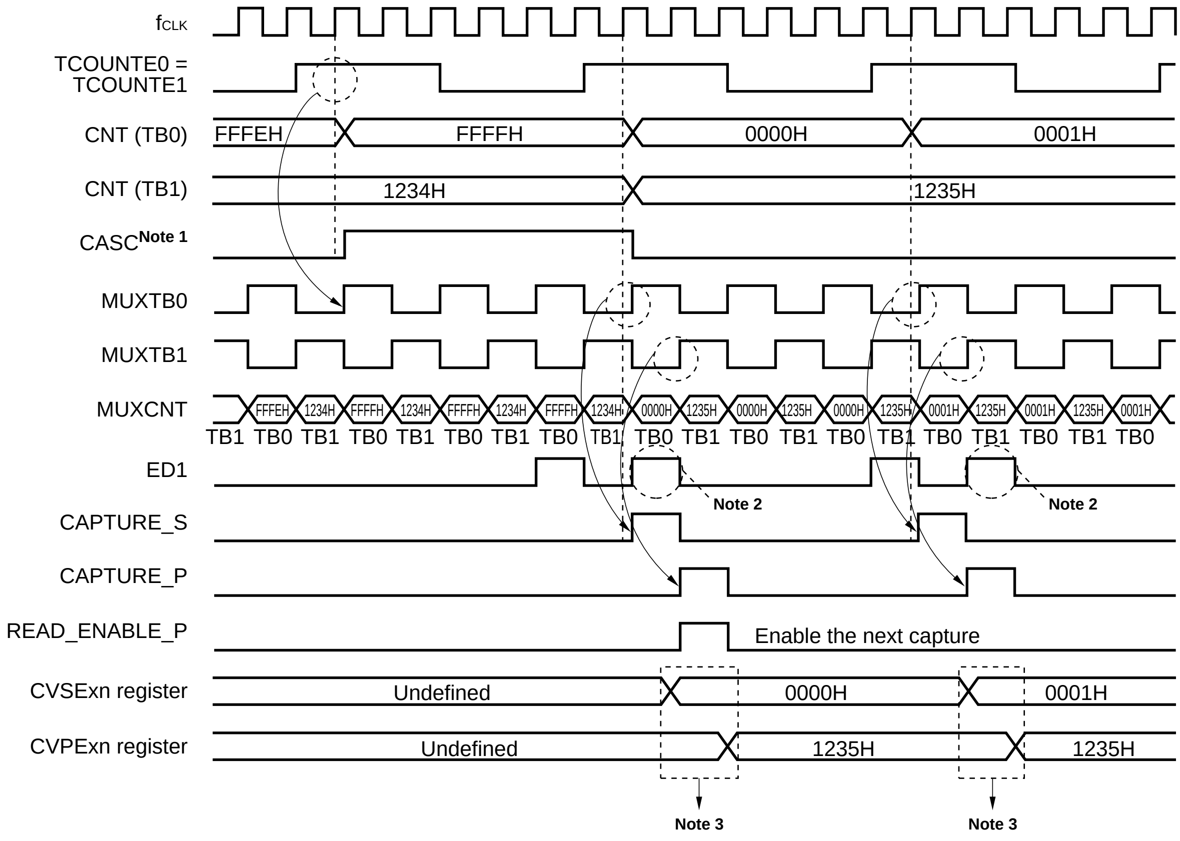
Remarks:

1.	$f_{CLK} = f_{CPU}$:	Base clock
2.	BUFFER:	Timing of write operation from CVSExn register to CVPExn register
	CAPTURE_P:	Capture trigger signal of main capture register
	CAPTURE_S:	Capture trigger signal of sub capture register
	ED1:	Capture event signal input from edge selection circuit
	MUXCNT:	Count value to sub-channel x
	MUXTB0:	Multiplex signal of TBASE0n
	MUXTB1:	Multiplex signal of TBASE1n
	READ_ENABLE_P:	Read timing of CVPExn register
	TB0, TB1:	Time base TBASE0n, TBASE1n

3. x = 1 to 4
m = 12, when x = 1 or 2
m = 34, when x = 3 or 4
n = 0 to 2

Figure 9-35: Timer E Capture Operation: 32-Bit Cascade Operation Mode

(When CMSEmn Register’s TB1Ex Bit = 1, TB0Ex Bit = 1, CCSEx Bit = 0, LNKEx Bit = 0, BFEEx Bit = Arbitrary, EEVEx Bit = 1, and CSCEn Register’s SEVEx Bit = 0)



Notes: **1.** TBASE1n performs count operation when, in the 32-bit mode, CASC (CNT = MAX. for TBASE0n) is input to TBASE1n and the rising edge of CTC is detected.

2. If an event occurs during this timing, it is ignored.

3. CPU read access is not performed in this timing (wait status).

Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock

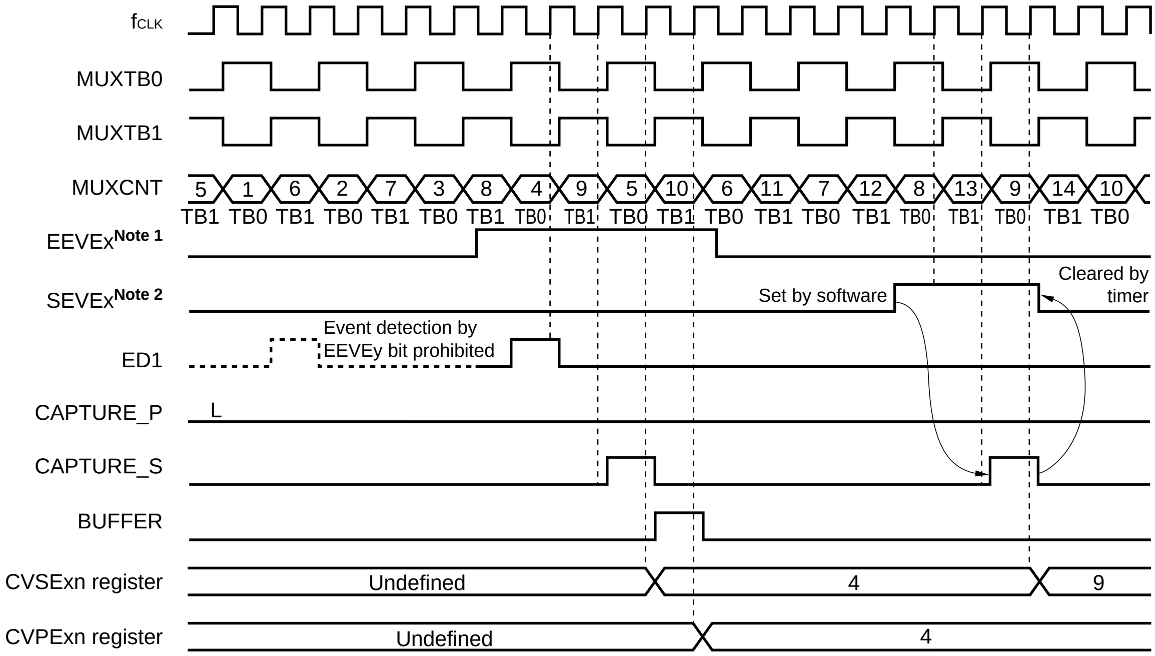
2.

CAPTURE_P:	Capture trigger signal of main capture register
CAPTURE_S:	Capture trigger signal of sub capture register
CASC:	TBASE1n count signal in 32-bit mode
CNT (TB0):	Count value of time baser TBASE0n
CNT (TB1):	Count value of time baser TBASE1n
ED1:	Capture event signal input from edge selection circuit
MUXCNT:	Count value to sub-channel x
MUXTB0:	Multiplex signal of TBASE0n
MUXTB1:	Multiplex signal of TBASE1n
READ_ENABLE_P:	Read timing of CVPExn register
TB0, TB1:	Time base TBASE0n, TBASE1n
TCOUNTE0:	Timer TMEn count enable signal input of time base TBASE0n
TCOUNTE1:	Timer TMEn count enable signal input of time base TBASE1n

3. x = 1 to 4
m = 12, when x = 1 or 2
m = 34, when x = 3 or 4
n = 0 to 2

Figure 9-36: Timer E Capture Operation: Capture Control by Software and Trigger Timing

(When CMSEmn Register's TB1Ex Bit = 0, TB0Ex Bit = 1, CCSEx Bit = 0, LNKEx Bit = 0, BFEEx Bit = 1)



Notes: **1.** EEVEx bit of CMSEmn register
2. SEVEx bit of CSCEn register

Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock

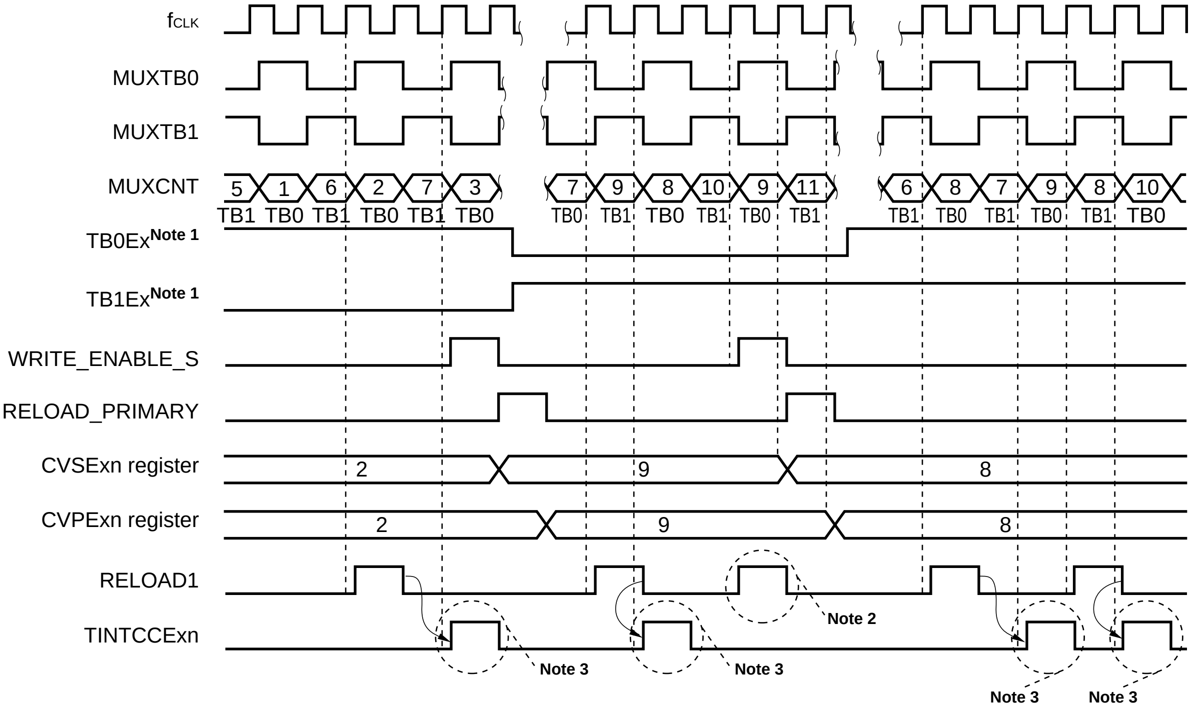
2.

BUFFER:	Timing of write operation from CVSExn register to CVPExn register
CAPTURE_P:	Capture trigger signal of main capture register
CAPTURE_S:	Capture trigger signal of sub capture register
ED1:	Capture event signal input from edge selection circuit
MUXCNT:	Count value to sub-channel x
MUXTB0:	Multiplex signal of TBASE0n
MUXTB1:	Multiplex signal of TBASE1n
TB0, TB1:	Time base TBASE0n, TBASE1n

3. x = 1 to 4
m = 12, when x = 1 or 2
m = 34, when x = 3 or 4
n = 0 to 2

Figure 9-37: Timer E Compare Operation: Buffer-Less Mode

(When CMSEmn Register's CCSEx Bit = 1, LNKEx Bit = Arbitrary, BFEEx Bit = 0)



Notes: **1.** TB1Ex, TB0Ex bits of CMSEmn register

2. No interrupt is generated due to compare match with counter differing from TB1Ex, TB0Ex bit settings.

3. TINTCCExn is generated to match the cycle from rising edge to falling edge of MUXTB0.

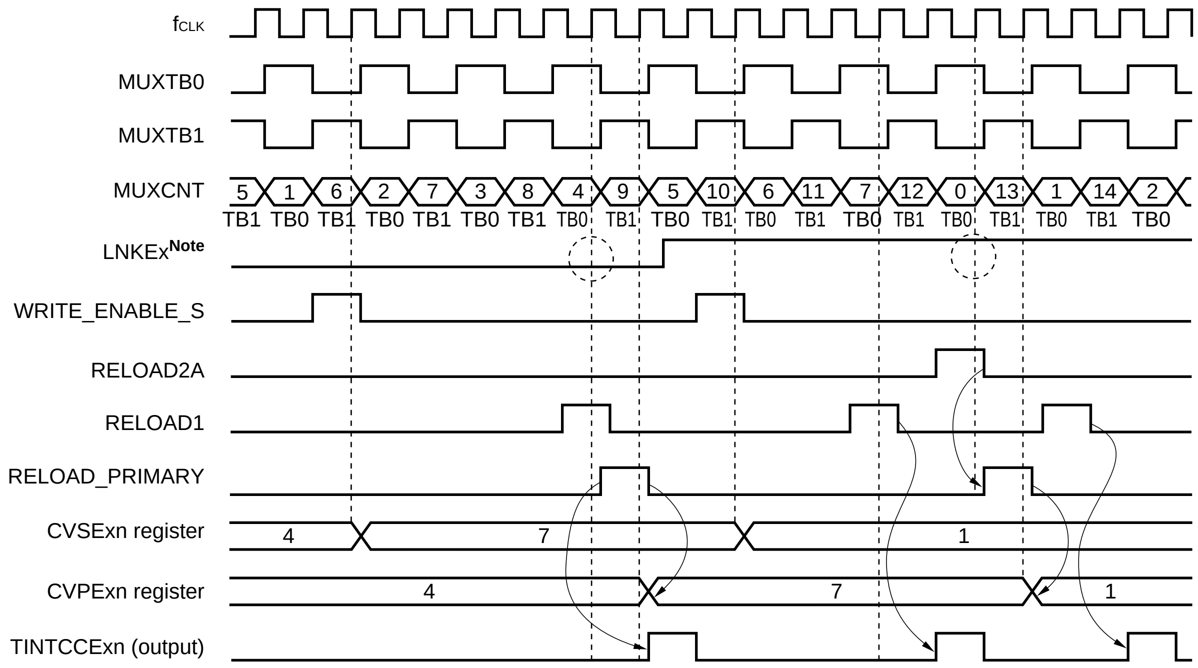
Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock

2. MUXCNT: Count value to sub-channel x
MUXTB0: Multiplex signal of TBASE0n
MUXTB1: Multiplex signal of TBASE1n
RELOAD1: Compare match signal
RELOAD_PRIMARY: Timing of write operation from CVSExn register to CVPExn register
WRITE_ENABLE_S: Timing of CVSExn register write operation
TB0, TB1: Time base TBASE0n, TBASE1n

3. x = 1 to 4
m = 12, when x = 1 or 2
m = 34, when x = 3 or 4
n = 0 to 2

Figure 9-38: Timer E Compare Operation: Mode with Buffer

(When CMSEmn Register's CCSEx Bit = 1, BFEEx Bit = 1, and Operation Is Delayed Through Setting of LNKEx Bit)



Note: LNKEx bit of CMSEmn register

Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock

2. MUXCNT: Count value to sub-channel x
MUXTB0: Multiplex signal of TBASE0n
MUXTB1: Multiplex signal of TBASE1n
RELOAD1: Compare match signal
RELOAD2A: Zero count signal input of TBASE0n (occurs when TBASE0n = 0000H)
RELOAD_PRIMARY: Timing of write operation from CVSExn register to CVPExn register
WRITE_ENABLE_S: Timing of CVSExn register write operation
TB0, TB1: Time base TBASE0n, TBASE1n

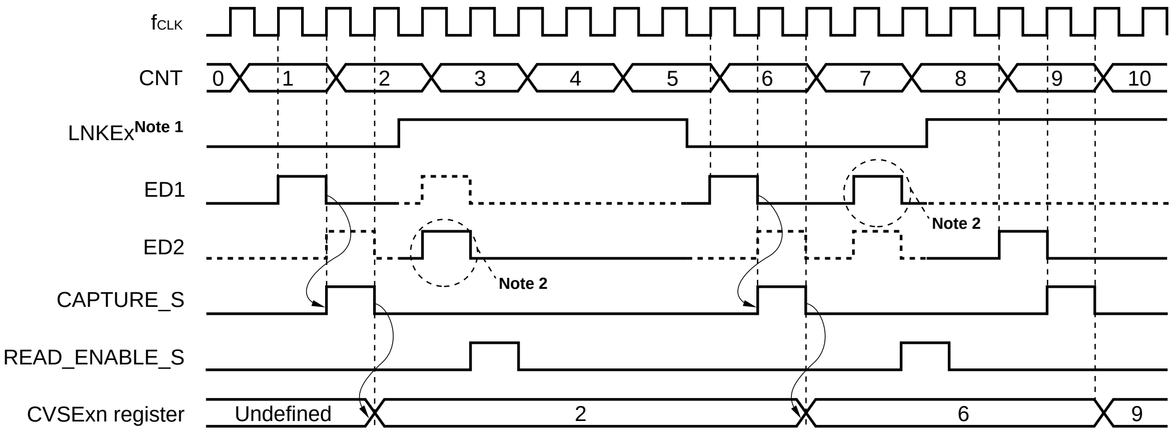
3. x = 1 to 4
m = 12, when x = 1 or 2
m = 34, when x = 3 or 4
n = 0 to 2

(4) Operation of capture/compare register (sub-channels 0, 5)

Figures 9-39 and 9-40 show the operation of the capture/compare register (sub-channels 0, 5).

Figure 9-39: Timer E Capture Operation: Count Value Read Timing

(When CMSE05n Register's CCSEx Bit = 0, EEVEx Bit = 1, and CSCEn Register's SEVEx Bit = 0)



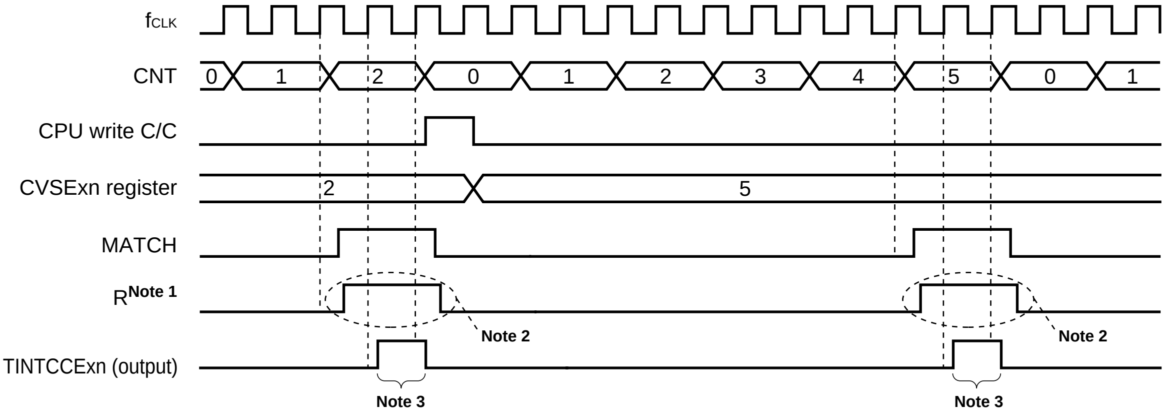
Notes: **1.** LNKEx bit of CMSE05n register
2. If an event occurs in this timing, it is ignored.

Remarks:	**1.** $f_{CLK} = f_{CPU}$:	Base clock
	2. CNT:	Count value of time base TBASEyn
	CAPTURE_S:	Capture trigger signal of sub capture register
	ED1, ED2:	Capture event signal inputs from edge selection circuit
	READ_ENABLE_S:	Read timing for CVSExn register

3. x = 0, 5
y = 0, when x = 0
y = 1, when x = 5
n = 0 to 2

Figure 9-40: Timer E Compare Operation: Timing of Compare Match and Write Operation to Register

(When CMSE05n Register's CCSEx Bit = 1, EEVEx Bit = Arbitrary, and CSCE0n Register's SEVEx Bit = Arbitrary)



Notes: **1.** Controls TBASEyn clear by sub-channel x compare match and count direction.
2. MATCH is forwarded to the R input of the timebase(s).
3. The pulse width is always 1 clock.

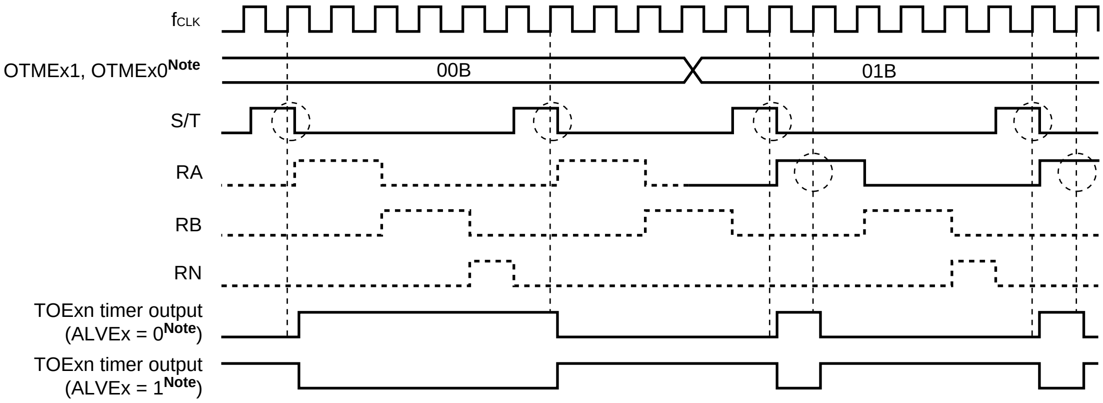
Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock
2. CNT: Count value of time base TBASEyn
MATCH: CVSExn register compare match timing
R: Compare match input (sub-channel x)
3. x = 0, 5
y = 0, when x = 0
y = 1, when x = 5
n = 0 to 2

(5) Operation of output circuit

Figures 9-41 to 9-44 show the output circuit operation.

Figure 9-41: Timer E Signal Output Operation: Toggle Mode 0 and Toggle Mode 1

(When OCTLEn Register's SWFEx Bit = 0, and ODELEn Register's ODLEx2 to ODLEx0 Bits = 0)


fCLK
OTMEx1, OTMEx0Note
00B
01B
S/T
RA
RB
RN
TOExn timer output (ALVEx = 0Note)
TOExn timer output (ALVEx = 1Note)

Note: ALVEx, OTMEx1, OTMEx0 bits of OCTLEn register

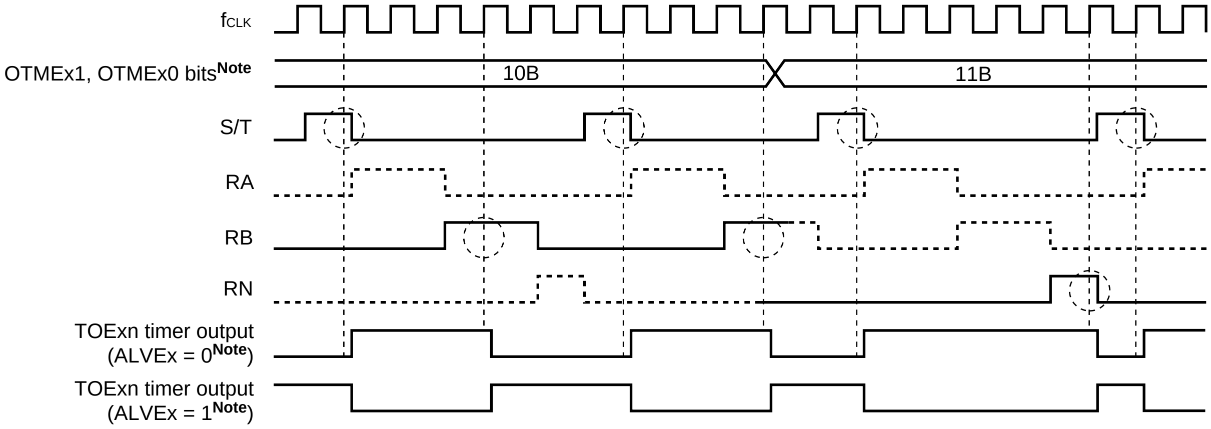
Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock

2. RA: Zero count signal input of TBASE0n (output circuit reset signal)
RB: Zero count signal input of TBASE1n (output circuit reset signal)
RN: Interrupt signal input of sub-channel x (output circuit reset signal)
S/T: Interrupt signal input of sub-channel x (output circuit set signal)

3. x = 1 to 4
n = 0 to 2

Figure 9-42: Timer E Signal Output Operation: Toggle Mode 2 and Toggle Mode 3

(When OCTLEn Register's SWFEx Bit = 0, and ODELEn Register's ODLEx2 to ODLEx0 Bits = 0)

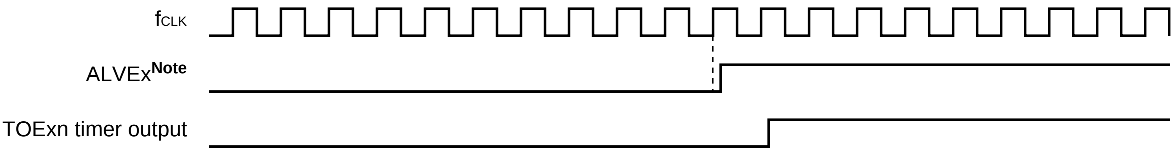


Note: ALVEx, OTMEx1, OTMEx0 bits of OCTLEn register

Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock
2. RA: Zero count signal input of TBASE0n (output circuit reset signal)
RB: Zero count signal input of TBASE1n (output circuit reset signal)
RN: Interrupt signal input of sub-channel x (output circuit reset signal)
S/T: Interrupt signal input of sub-channel x (output circuit set signal)
3. x = 1 to 4
n = 0 to 2

Figure 9-43: Timer E Signal Output Operation: During Software Control

(When OCTLEn Register’s OTMEx1, OTMEx0 Bits = Arbitrary, SWFEx Bit = 1, and ODELEn Register’s ODLEx2 to ODLEx0 Bits = 0)

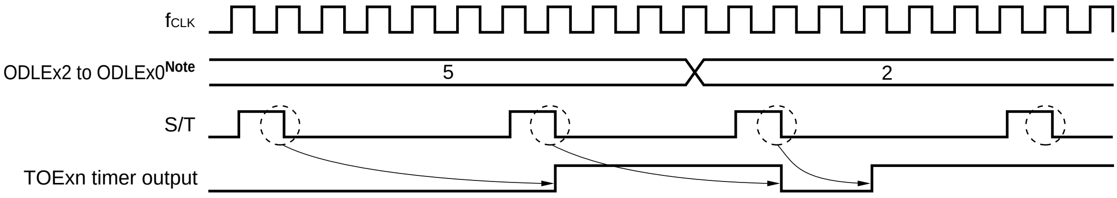


Note: ALVEx bit of OCTLEn register

Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock
2. x = 1 to 4
n = 0 to 2

Figure 9-44: Timer E Signal Output Operation: During Delay Output Operation

(When OCTLEn Register’s OTMEx1, OTMEx0 Bits = 0, ALVEx = 0, SWFEx Bit = 0)



Note: Refer to (11)“Timer E output delay registers 0 to 2 (ODELE0 to ODELE2)” on page 279.

Remarks: **1.** $f_{CLK} = f_{CPU}$: Base clock
2. x = 1 to 4
n = 0 to 2

Chapter 10 Watch Timer

10.1 Function

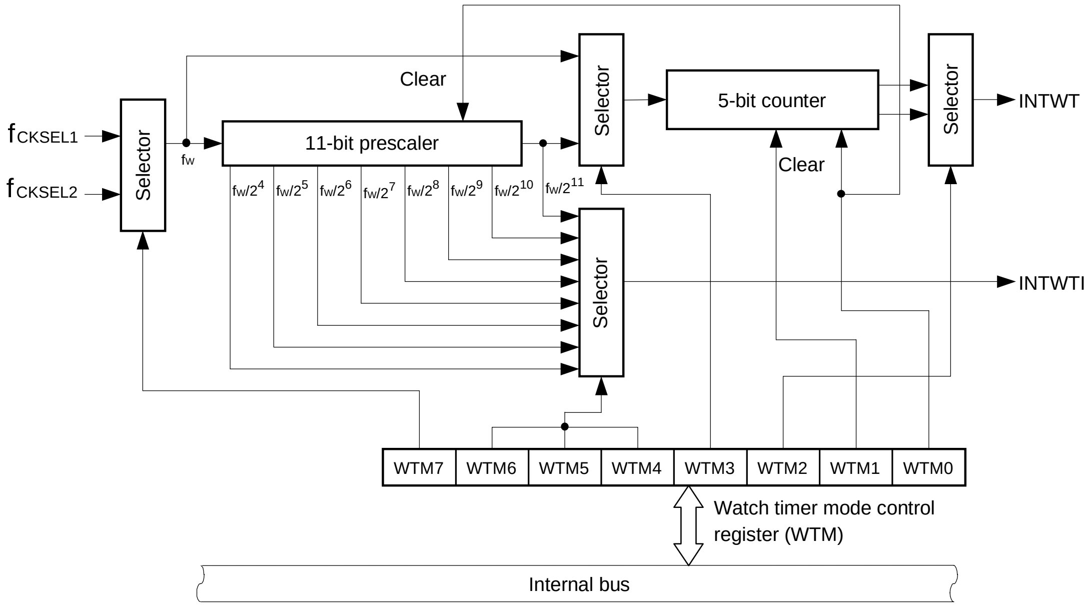
The watch timer has the following functions:

- Watch timer
- Interval timer

The watch timer and interval timer functions can be used at the same time.

Figure 10-1 shows the block diagram of the watch timer.

Figure 10-1: Block Diagram of Watch Timer



(1) Watch timer

The watch timer generates an interrupt request (INTWT) at time intervals of 512 µs or 2.097 s by using the subsystem clock f_{CKSEL1} or f_{CKSEL2} or the derived 11-bit prescaler clock from f_{CKSEL1} or f_{CKSEL2} (refer to Figure 8-1: “Block Diagram of the Clock Generator” on page 225).

(2) Interval timer

The interval timer generates an interrupt request (INTWTI) at time intervals specified in advance.

Table 10-1: Interval Time of Interval Timer (f_{SUB} = 4 MHz)

Interval Time	fw = f_{CKSEL2}	fw = f_{CKSEL1}
2^4 × 1/fw	2.408 ms	512 µs
2^5 × 1/fw	4.096 ms	1.024 ms
2^6 × 1/fw	8.192 ms	2.048 ms
2^7 × 1/fw	16.384 ms	4.096 ms
2^8 × 1/fw	32.768 ms	8.192 ms
2^9 × 1/fw	65.536 ms	16.384 ms
2^{10} × 1/fw	131.072 ms	32.768 ms
2^{11} × 1/fw	262.144 ms	65.536 ms

Remarks: **1.** fw: Watch timer clock frequency
2. interval times change accordingly if f_{SUB} = 5 MHz

10.2 Configuration

The watch timer consists of the following hardware:

Table 10-2: Configuration of Watch Timer

Item	Configuration
Counter	5 bits × 1
Prescaler	11 bits × 1
Control register	Watch timer mode control register (WTM)

10.3 Watch Timer Control Register

The watch timer mode control register (WTM) controls the watch timer.

(1) Watch timer mode control register (WTM)

This register enables or disables the count clock and operation of the watch timer, sets the interval time of the prescaler, controls the operation of the 5-bit counter, and sets the set time of the watch flag.

WTM is set by a 1-bit or 8-bit memory manipulation instruction.

Figure 10-2: Watch Timer Mode Control Register (WTM)

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
WTM	WTM7	WTM6	WTM5	WTM4	WTM3	WTM2	WTM1	WTM0	FFFFF560H	R/W	00H

WTM6	WTM5	WTM4	Selects Interval Time of Prescaler (f_{SUB} = 4 MHz)
0	0	0	2^4/fw (2.408 ms, 512 µs)
0	0	1	2^5/fw (4.096 ms, 1.024 ms)
0	1	0	2^6/fw (8.192 ms, 2.048 ms)
0	1	1	2^7/fw (16.384 ms, 4.096 ms)
1	0	0	2^8/fw (32.768 ms, 8.192 ms)
1	0	1	2^9/fw (65.536 ms, 16.384 ms)
1	1	0	2^{10}/fw (131.072 ms, 32.768 ms)
1	1	1	2^{11}/fw (262.144 ms, 62.536 ms)

WTM3	WTM2	Selects Set Time of Watch Flag
0	0	2^{14}/fw (2.097152 s, 524.188 ms)
0	1	2^{13}/fw (1.048576 ms, 262.144 ms)
1	0	2^5/fw (4.096 ms, 1.024 ms)
1	1	2^4/fw (2.048 ms, 512 µs)

WTM1	Controls Operation of 5-bit Counter
0	Clears after operation stops
1	Starts

WTM0	Enables Operation of Watch Timer
0	Stops operation (clears both prescaler and timer)
1	Enables operation

WTM7	Selects main input frequency from prescaler
0	Clock input f_{CKSEL1} is selected
1	Clock input f_{CKSEL2} is selected

Remarks: **1.** fw: Watch timer clock frequency

2. Values in parentheses apply when fxx = 4 MHz (Refer to Chapter 8 “Clock Generator” on page 225.

Sub Clock f_{SUB}	Input Clock	fw
4 MHz	f_{CKSEL1}	31250 Hz
4 MHz	f_{CKSEL2}	7812.5 Hz
5 MHz	f_{CKSEL1}	39062.5 Hz
5 MHz	f_{CKSEL2}	9765.625 Hz

10.4 Operations

10.4.1 Operation as watch timer

The watch timer operates with time intervals from 2.09715 s to 512 µs$^{\text{Note}}$ with fw of 31250 Hz / 7812.5 Hz.

The watch timer generates an interrupt request at fixed time intervals.

The count operation of the watch timer is started when bits 0 (WTM0) and 1 (WTM1) of the watch timer mode control register (WTM) are set to 1. When these bits are cleared to 0, the 11-bit prescaler and 5-bit counter are cleared, and the watch timer stops the count operation.

When the interval timer function is started at the same time, the watch timer can be started from 0 second by resetting WTM1 to 0. However, an error of up to 2.09715 s to 512 µs may occur when the watch timer overflows (INTWT).

Notes: **1.** f_{SUB} = 4 MHz with fw of 31250 Hz [f_{CKSEL1}] / 7812.5 Hz [f_{CKSEL2}]
2. f_{SUB} = 5 MHz with fw of 39062.5 Hz [f_{CKSEL1}] / 9765.625 Hz [f_{CKSEL2}]

10.4.2 Operation as interval timer

The watch timer can also be used as an interval timer that repeatedly generates an interrupt at intervals specified by a count value set in advance.

The interval time can be selected by bits 4 through 6 (WTM4 through WTM6) of the watch timer mode control register (WTM).

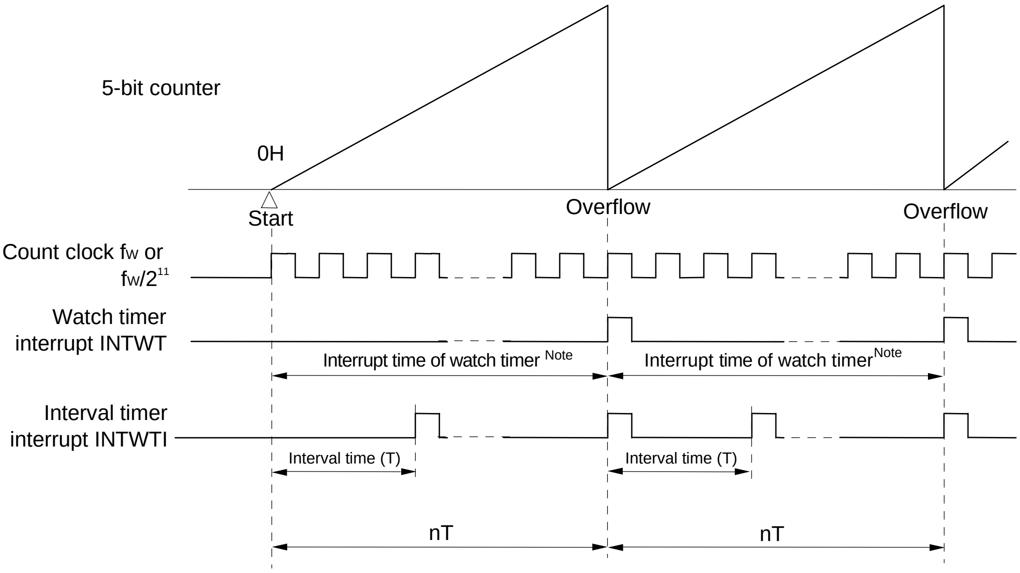
Table 10-3: Interval Time of Interval Timer

WTM6	WTM5	WTM4	Interval Time	fw = $f_{CKSEL2}$$^{\text{Note}}$	fw = $f_{CKSEL1}$$^{\text{Note}}$
0	0	0	2^4 × 1/fw	2.408 ms	512 µs
0	0	1	2^5 × 1/fw	4.096 ms	1.024 ms
0	1	0	2^6 × 1/fw	8.192 ms	2.048 ms
0	1	1	2^7 × 1/fw	16.384 ms	4.096 ms
1	0	0	2^8 × 1/fw	32.768 ms	8.192 ms
1	0	1	2^9 × 1/fw	65.536 ms	16.384 ms
1	1	0	2^{10} × 1/fw	131.072 ms	32.768 ms
1	1	1	2^{11} × 1/fw	262.144 ms	65.536 ms

Note: f_{SUB} = 4 MHz

Remark: fw: Watch timer clock frequency

Figure 10-3: Operation Timing of Watch Timer/Interval Timer


5-bit counter
0H
Start
Overflow
Overflow
Count clock fw or fw/2^11
Watch timer interrupt INTWT
Interrupt time of watch timer Note
Interrupt time of watch timer Note
Interval timer interrupt INTWTI
Interval time (T)
Interval time (T)
nT
nT

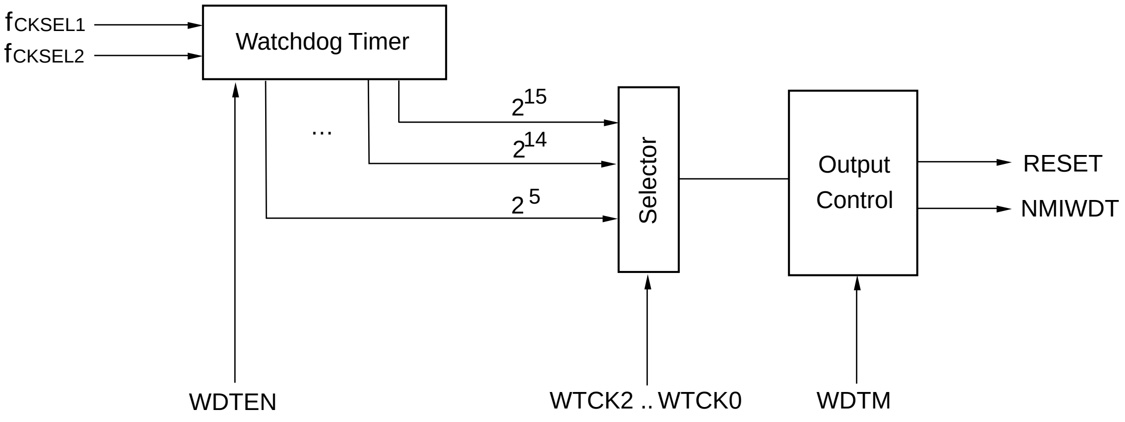
Notes: **1.** f_{SUB} = 4 MHz with fw of 31250 Hz [f_{CKSEL1}] / 7812.5 Hz [f_{CKSEL2}]
2. f_{SUB} = 5 MHz with fw of 39062.5 Hz [f_{CKSEL1}] / 9765.625 Hz [f_{CKSEL2}]

Remarks: **1.** fw: Watch timer clock frequency
2. n: Interval timer operation numbers

Chapter 11 Watchdog Timer

- Features:
  - Generates reset or NMI (selectable)
  - Have to be started once by software control (afterwards protected)
  - Will operate at system frequency divided by 4 to get a lower watch limit of 1ms.

Figure 11-1: Block Diagram of Watchdog Timer Unit



11.1 Watchdog timer mode

This mode detects program runaway. When runaway is detected, a non-maskable interrupt can be generated.

Table 11-1: Runaway Detection Time by Watchdog Timer

Clock	Runaway detection time	
	f_{SUB} = 4 MHz	f_{SUB} = 5 MHz
$2^{15}/f_{CKSEL1}$	1.04 s	0.839 s
$2^{14}/f_{CKSEL1}$	0.52 s	0.419 s
$2^{6}/f_{CKSEL1}$	0.002 s	0.0016 s
$2^{5}/f_{CKSEL1}$	0.001 s	0.0008 s
$2^{15}/f_{CKSEL2}$	4.19 s	3.355 s
$2^{14}/f_{CKSEL2}$	2.097 s	1.677 s
$2^{6}/f_{CKSEL2}$	0.008 s	0.006 s
$2^{5}/f_{CKSEL2}$	0.004 s	0.003

Note: f_{SUB} = 4 MHz with fw of 31250 Hz[f_{CKSEL1}] / 7812.5 Hz[f_{CKSEL2}]
f_{SUB} = 5 MHz with fw of 39062.5 Hz[f_{CKSEL1}] / 9765.625 Hz[f_{CKSEL2}]

11.2 Control Register

11.2.1 Watchdog timer mode register (WDTM)

This register sets the operating mode of the watchdog timer, and enables and disables counting.
This register sets the overflow times of the watchdog timer.
WDTM is set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets WDTM to 00H.

	7	6	5	4	3	2	1	0	Address	R/W	At Reset
WDTM	WDTEN	WDTM	0	0	0	WDCK2	WDCK1	WDCK0	FFFFF570H	R/W	00H

<table>
<tr><th>Bit name</th><th>Function</th></tr>
<tr><td>WDTEN</td><td>Watch Dog Timer enable
Starts/Stops and clears Watch Dog Timer
0: Watch Dog Timer not started
1: clear counter and start counting/continue counting
<b>Note:</b> Once set, only a reset signal clears this bit!</td></tr>
<tr><td>WDTM</td><td>Watch Dog Timer Mode
Selects event type:
0: watch dog timer generates NMI
1: watch dog timer generates reset (once set to 1, only reset can clear)</td></tr>
<tr><td>WTCK2 to WTCK0</td><td>Count Enable Select
<table>
<tr><th>WTCK2</th><th>WTCK1</th><th>WTCK0</th><th>Watchdog Time</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>$2^{15}/f_{CKSEL1}$ (4 MHz: → 1.04 s)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$2^{14}/f_{CKSEL1}$ (4 MHz: → 0.52 s)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$2^{6}/f_{CKSEL1}$ (4 MHz: → 0.002 s)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$2^{5}/f_{CKSEL1}$ (4 MHz: → 0.001 s)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$2^{15}/f_{CKSEL2}$ (4 MHz: → 4.19 s)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>$2^{14}/f_{CKSEL2}$</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>$2^{6}/f_{CKSEL2}$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>$2^{5}/f_{CKSEL2}$</td></tr>
</table>
<b>Notes:</b> <b>1.</b> $f_{CKSEL1} = f_{SUB} / 128$; $f_{CKSEL2} = f_{SUB} / 512$
<b>2.</b> for f_{CKSEL1}/f_{CKSEL2} refer to Chapter 8.2 “Configuration” on page 225.</td></tr>
</table>

Note: If WDTEN is once set to 1, the register cannot be cleared to 0 by software. Therefore, when the count starts, the count cannot be stopped except by $\overline{\text{RESET}}$ input.

Caution: If WDTEN is set to 1, the register cannot be cleared. The actual overflow time is a maximum of 0.5% less than the set time.

11.3 Operation

11.3.1 Operating as watchdog timer

Set bit 6 (WDTM) of the watchdog timer mode register (WDTM) to 1 to operate as a watchdog timer to detect program runaway and generate an $\overline{\text{RESET}}$ signal.
Setting bit 7 (WDTEN) of WDTM to 1 starts the count. After counting starts, if WDTEN is set to 1 again within the set time interval for runaway detection, the watchdog timer is cleared and counting starts again.
If WDTEN is not set to 1 and the runaway detection time has elapsed, a non-maskable interrupt (NMI-WDT) or a $\overline{\text{RESET}}$ is generated.
The watchdog timer stops running in the STOP mode. Consequently, set WDTEN to 1 and clear the watchdog timer before entering the STOP mode.

Caution: Sometimes, the actual runaway detection time is a maximum of 0.5% less than the set time.

Table 11-2: Runaway Detection Time by Watchdog Timer

Clock	Runaway detection time	
	f_{SUB} = 4 MHz	f_{SUB} = 5 MHz
$2^{15}/f_{CKSEL1}$	1.04 s	0.839 s
$2^{14}/f_{CKSEL1}$	0.52 s	0.419 s
$2^{6}/f_{CKSEL1}$	0.002 s	0.0016 s
$2^{5}/f_{CKSEL1}$	0.001 s	0.0008 s
$2^{15}/f_{CKSEL2}$	4.19 s	3.355 s
$2^{14}/f_{CKSEL2}$	2.097 s	1.677 s
$2^{6}/f_{CKSEL2}$	0.008 s	0.006 s
$2^{5}/f_{CKSEL2}$	0.004 s	0.003 s

[MEMO]

Chapter 12 Serial Interface Function

12.1 Features

The serial interface function provides three types of serial interfaces combining a total of eight transmit/ receive channels. All channels can be used simultaneously.
The three interface formats are as follows.

(1) Asynchronous serial interfaces (UART0 to UART2): 3 channels
(2) Clocked serial interfaces (CSI0, CSI1): 2 channels
(3) FCAN controller: 3 channels

Remark: For details about the FCAN controller, refer to **Chapter 13 FCAN Interface Function**.

UART0 to UART2, transmit/receive 1-byte serial data following a start bit and support full-duplex communication.
CSI0 and CSI1 perform data transfer according to three types of signals, namely serial clocks ($\overline{\text{SCK0}}$, $\overline{\text{SCK1}}$), serial inputs (SI0, SI1), and serial outputs (SO0, SO1) (3-wire serial I/O).
FCAN conforms to CAN specification Ver. 2.0 Part B, and provides 64-message buffers.

12.2 Asynchronous Serial Interfaces 0 to 2 (UART0, UART1, UART2)

12.2.1 Features

- Transfer rate: 300 bps to 625 Kbps
  (using a dedicated baud rate generator and an internal system clock of 20 MHz)

- Full-duplex communications
  - On-chip reception buffer register (RXBn)
  - On-chip transmission buffer register (TXBn)

- Two-pin configuration
  - TXDn: Transmit data output pin
  - RXDn: Receive data input pin

- Reception error detection functions
  - Parity error
  - Framing error
  - Overrun error

- Interrupt sources: 3 types
  - Reception error interrupt (INTSERn): Interrupt is generated according to the logical OR of the three types of reception errors.
  - Reception completion interrupt (INTSRn): Interrupt is generated when receive data is transferred from the shift register to the reception buffer register after serial transfer is completed during a reception enabled state.
  - Transmission completion interrupt (INTSTn): Interrupt is generated when the serial transmission of transmit data (8 or 7 bits) from the shift register is completed.

- Character length: 7 or 8 bits

- Parity functions: Odd, even, 0, or none

- Transmission stop bits: 1 or 2 bits

- On-chip dedicated baud rate generator

Remark: n = 0 to 2

12.2.2 Configuration

UARTn is controlled by the asynchronous serial interface mode register (ASIMn), asynchronous serial interface status register (ASISn), and asynchronous serial interface transmission status register (ASIFn). Receive data is maintained in the reception buffer register (RXBn), and transmit data is written to the transmission buffer register (TXBn).
Figure 12-1, “Asynchronous Serial Interfaces 0 to 2 Block Diagram,” on page 312 shows the configuration of the asynchronous serial interface (UARTn) (n = 0 to 2).

(1) Asynchronous serial interface mode registers 0 to 2 (ASIM0 to ASIM2)

The ASIMn register is an 8-bit register for specifying the operation of the asynchronous serial interface.

(2) Asynchronous serial interface status registers 0 to 2 (ASIS0 to ASIS2)

The ASISn register consists of a set of flags that indicate the error contents when a reception error occurs. The various reception error flags are set (1) when a reception error occurs and are reset (0) when the ASISn register is read.

(3) Asynchronous serial interface transmission status registers 0 to 2 (ASIF0 to ASIF2)

The ASIFn register is an 8-bit register that indicates the status when a transmit operation is performed.
This register consists of a transmission buffer data flag, which indicates the hold status of TXBn data, and the transmission shift register data flag, which indicates whether transmission is in progress.

(4) Reception control parity check

The receive operation is controlled according to the contents set in the ASIMn register. A check for parity errors is also performed during a receive operation, and if an error is detected, a value corresponding to the error contents is set in the ASISn register.

(5) Reception shift register

This is a shift register that converts the serial data that was input to the RXDn pin to parallel data. One byte of data is received, and if a stop bit is detected, the receive data is transferred to the reception buffer register (RXBn).
This register cannot be directly manipulated.

(6) Reception buffer registers 0 to 2 (RXB0 to RXB2)

RXBn is an 8-bit buffer register for holding receive data. When 7 characters are received, 0 is stored in the MSB.
During a reception enabled state, receive data is transferred from the reception shift register to the RXBn, synchronized with the end of the shift-in processing of one frame.
Also, the reception completion interrupt request (INTSRn) is generated by the transfer of data to the RXBn.

(7) Transmission shift register

This is a shift register that converts the parallel data that was transferred from the transmission buffer register (TXBn) to serial data.
When one byte of data is transferred from the TXBn, the shift register data is output from the TXDn pin.
This register cannot be directly manipulated.

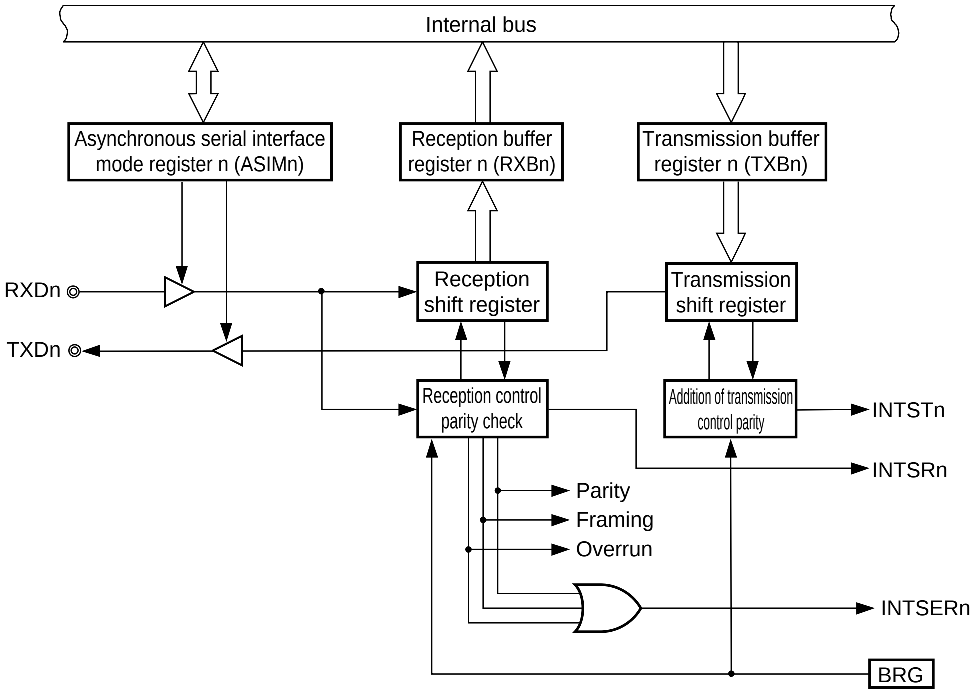
(8) Transmission buffer registers 0 to 2 (TXB0 to TXB2)

TXBn is an 8-bit buffer for transmit data. A transmit operation is started by writing transmit data to TXBn. The transmission completion interrupt request (INTSTn) is generated synchronized with the completion of transmission of one frame.

(9) Addition of transmission control parity

A transmit operation is controlled by adding a start bit, parity bit, or stop bit to the data that is written to the TXBn register, according to the contents that were set in the ASIMn register.

Figure 12-1: Asynchronous Serial Interfaces 0 to 2 Block Diagram


Internal bus
Asynchronous serial interface mode register n (ASIMn)
Reception buffer register n (RXBn)
Transmission buffer register n (TXBn)
RXDn
TXDn
Reception shift register
Transmission shift register
Reception control parity check
Addition of transmission control parity
INTSTn
INTSRn
Parity
Framing
Overrun
INTSERn
BRG

Remark: n = 0 to 2

12.2.3 Control registers

(1) Asynchronous serial interface mode registers 0 to 2 (ASIM0 to ASIM2)

The ASIMn register is an 8-bit register that controls the UARTn transfer operation.
This register can be read/written in 8 bit or 1-bit units (n = 0 to 2).

Figure 12-2: Asynchronous Serial Interface Mode Registers 0 to 2 (ASIM0 to ASIM2) (1/3)

	7	6	5	4	3	2	1	0	Address	Initial value
ASIM0	CAE	TXE	RXE	PS1	PS0	CL	SL	ISRM	FFFFFA00H	01H

	7	6	5	4	3	2	1	0	Address	Initial value
ASIM1	CAE	TXE	RXE	PS1	PS0	CL	SL	ISRM	FFFFFA10H	01H

	7	6	5	4	3	2	1	0	Address	Initial value
ASIM2	CAE	TXE	RXE	PS1	PS0	CL	SL	ISRM	FFFFFA20H	01H

Bit Position	Bit Name	Function
7	CAE	Enables/disables clock operation. 0: Disable clock operation (reset internal circuit asynchronously.) 1: Enable clock operation UARTn operation clock control and asynchronous reset of the internal circuit are performed with the CAE bit. When the CAE bit is set to 0, the UARTn operation clock stops (fixed to low level), and an asynchronous reset is applied to internal UARTn latch. The TXDn pin output is low level when the CAE bit = 0, and high level when the CAE bit = 1. Therefore, perform CAE setting in combination with port mode register (PM1, PM2, PM6) so as to avoid malfunction on the other side at start-up (Set the port to the output mode after setting the CAE bit to 1). Input from the RXDn pin is fixed to high level with CAE bit = 0.
6	TXE	Enables/disables transfer. 0: Disable transfer (Perform synchronized reset of transfer circuit.) 1: Enable transfer **Cautions: 1. Set the TXE bit to 1 after setting the CAE bit to 1 when starting transfer. Set the CAE bit to 0 after setting the TXE bit to 0 when stopping transfer.** **2. To initialize the transfer unit, clear (0) the TXE bit, and after letting 2 Clock cycles (base clock) elapse, set (1) the TXE bit again. If the TXE bit is not set again, initialization may not be successful. (For details about the base clock, refer to 12.2.6 Dedicated baud rate generators (BRG) of UARTn (n = 0 to 2).)**

Figure 12-2: Asynchronous Serial Interface Mode Registers 0 to 2 (ASIM0 to ASIM2) (2/3)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>5</td><td>RXE</td><td>Enables/disables reception.
0: Disable reception (Perform synchronous reset of reception circuit)
1: Enable reception
<b>Cautions: 1. Set the RXE bit to 1 after setting the CAE bit to 1 when starting transfer. Set the CAE bit to 0 after setting the RXE bit to 0 when stopping transfer.</b>
<b>2. To initialize the reception unit status, clear (0) the RXE bit, and after letting 2 Clock cycles (base clock) elapse, set (1) the RXE bit again. If the RXE bit is not set again, initialization may not be successful. (For details about the base clock, refer to 12.2.6 Dedicated baud rate generators 1 to 3 (BRG1 to BRG3).)</b></td></tr>
<tr><td>4, 3</td><td>PS1, PS0</td><td>Controls parity bit.
<table>
<tr><th>PS1</th><th>PS0</th><th>Transmit Operation</th><th>Receive Operation</th></tr>
<tr><td>0</td><td>0</td><td>Don’t output parity bit</td><td>Receive with no parity</td></tr>
<tr><td>0</td><td>1</td><td>Output 0 parity</td><td>Receive as 0 parity</td></tr>
<tr><td>1</td><td>0</td><td>Output odd parity</td><td>Judge as odd parity</td></tr>
<tr><td>1</td><td>1</td><td>Output even parity</td><td>Judge as even parity</td></tr>
</table>
<b>Cautions: 1. To overwrite the PS1 and PS0 bits, first clear (0) the TXE and RXE bits.</b>
<b>2. If “0 parity” is selected for reception, no parity judgment is performed. Therefore, no error interrupt is generated because the PE bit of the ASISn register is not set.</b>
• Even parity
If the transmit data contains an odd number of bits with the value “1”, the parity bit is set (1). If it contains an even number of bits with the value “1”, the parity bit is cleared (0). This controls the number of bits with the value “1” contained in the transmit data and the parity bit so that it is an even number.
During reception, the number of bits with the value “1” contained in the receive data and the parity bit is counted, and if the number is odd, a parity error is generated.
• Odd parity
In contrast to even parity, odd parity controls the number of bits with the value “1” contained in the transmit data and the parity bit so that it is an odd number. During reception, the number of bits with the value “1” contained in the receive data and the parity bit is counted, and if the number is even, a parity error is generated.</td></tr>
</table>

Remark: When reception is disabled, the reception shift register does not detect a start bit. No shift-in processing or transfer processing to the reception buffer register (RXBn) is performed, and the contents of the RXBn register are retained.
When reception is enabled, the reception shift operation starts, synchronized with the detection of the start bit, and when the reception of one frame is completed, the contents of the reception shift register are transferred to the RXBn register. A reception completion interrupt (INTSRn) is also generated in synchronization with the transfer to the RXBn register.

Figure 12-2: Asynchronous Serial Interface Mode Registers 0 to 2 (ASIM0 to ASIM2) (3/3)

Bit Position	Bit Name	Function
4, 3	PS1, PS0	• 0 parity During transmission, the parity bit is cleared (0) regardless of the transmit data. During reception, no parity error is generated because no parity bit is checked. • No parity No parity bit is added to transmit data. During reception, the receive data is considered to have no parity bit. No parity error is generated because there is no parity bit.
2	CL	Specifies character length of transmit/receive data. 0: 7 bits 1: 8 bits **Caution: To overwrite the CL bit, first clear (0) the TXE and RXE bits.**
1	SL	Specifies stop bit length of transmit data. 0: 1 bit 1: 2 bits **Caution: To overwrite the SL bit, first clear (0) the TXE bit. Since reception is always done using a single stop bit, the SL bit setting does not affect receive operations.**
0	ISRM	Enables/disables generation of reception completion interrupt requests when an error occurs. 0: Generate a reception error interrupt request (INTSERn) as an interrupt when an error occurs. In this case, no reception completion interrupt request (INTSRn) is generated. 1: Generate a reception completion interrupt request (INTSRn) as an interrupt when an error occurs. In this case, no reception error interrupt request (INTSERn) is generated. **Caution: To overwrite the ISRM bit, first clear (0) the RXE bit.**

(2) Asynchronous serial interface status registers 0 to 2 (ASIS0 to ASIS2)

The ASISn register, which consists of 3-bit error flags (PE, FE and OVE), indicates the error status when UARTn reception is completed.

The status flag, which indicates a reception error, always indicates the status of the error that occurred most recently. That is, if the same error occurred several times before the receive data was read, this flag would hold only the status of the error that occurred last.

The ASISn register is cleared to 00H by a read operation. When a reception error occurs, the reception buffer register (RXBn) should be read and the error flag should be cleared after the ASISn register is read.

This register is read-only in 8-bit or 1-bit units (n = 0 to 2).

Caution: When the CAE bit or RXE bit of the ASIMn register is set to 0, or when the ASIS0 register is read, the PE, FE, and OVE bits of the ASISn register are cleared (0).

Figure 12-3: Asynchronous Serial Interface Status Registers 0 to 2 (ASIS0 to ASIS2)

	7	6	5	4	3	2	1	0	Address	Initial value
ASIS0	0	0	0	0	0	PE	FE	OVE	FFFFFA03H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
ASIS1	0	0	0	0	0	PE	FE	OVE	FFFFFA13H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
ASIS2	0	0	0	0	0	PE	FE	OVE	FFFFFA23H	00H

Bit Position	Bit Name	Function
2	PE	This is a status flag that indicates a parity error. 0: When the ASIMn register's CAE and RXE bits are both set to 0, or when the ASISn register has been read 1: When reception was completed, the transmit data parity did not match the parity bit **Caution: The operation of the PE bit differs according to the settings of the PS1 and PS0 bits of the ASIMn register.**
1	FE	This is a status flag that indicates a framing error. 0: When the ASIMn register's CAE and RXE bits are both set to 0, or when the ASISn register has been read 1: When reception was completed, no stop bit was detected **Caution: For receive data stop bits, only the first bit is checked regardless of the number of stop bits.**
0	OVE	This is a status flag that indicates an overrun error. 0: When the ASIMn register's CAE and RXE bits are both 0, or when the ASISn register has been read. 1: UARTn completed the next receive operation before reading the RXBn receive data. **Caution: When an overrun error occurs, the next receive data value is not written to the RXBn register and the data is discarded.**

(3) Asynchronous serial interface transmission status registers 0 to 2 (ASIF0 to ASIF2)

The ASIFn register, which consists of 2-bit status flags, indicates the status during transmission. By writing the next data to the TXBn register after data is transferred from the TXBn register to the transmission shift register, transmit operations can be performed continuously without suspension even during an interrupt interval. When transmission is performed continuously, data should be written after referencing the ASIFn register to prevent writing to the TXBn register by mistake.
This register is read-only in 8-bit or 1-bit units (n = 0 to 2).

Figure 12-4: Asynchronous Serial Interface Transmit Status Registers 0 to 2 (ASIF0 to ASIF2)

	7	6	5	4	3	2	1	0	Address	Initial value
ASIF0	0	0	0	0	0	0	TXBF0	TXSF0	FFFFFA05H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
ASIF1	0	0	0	0	0	0	TXBF0	TXSF0	FFFFFA15H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
ASIF2	0	0	0	0	0	0	TXBF0	TXSF0	FFFFFA25H	00H

Bit Position	Bit Name	Function
1	TXBF	This is a transmission buffer data flag. 0: When the ASIMn register's CAE or TXE bits is 0, or when data has been transferred to the transmission shift register (Data to be transferred next to TXBn register does not exist). 1: Data exists in TXBn register when the TXBn register has been written to (Data to be transferred next exists in TXBn register).
0	TXSF	This is a transmission shift register data flag. It indicates the transmission status of UARTn. 0: When the ASIMn register's CAE or TXE bits is set to 0, or when following transfer completion, the next data transfer from the TXBn register is not performed (waiting transmission) 1: When data has been transferred from the TXBn register (Transmission in progress)

The following table shows relationships between the transmission status and write operations to TXBn register.

TXBF	TXSF	Transmission Status	Write Operation to TXBn
0	0	Initial status or transmission completed	Writing is permitted
0	1	Transmission in progress (no data is in TXBn)	Writing is permitted
1	0	Waiting transmission (data is in TXBn)	Writing is not permitted
1	1	Transmission in progress (data is in TXBn)	Writing is not permitted

Caution: When transmission is performed continuously, data should be written to TXBn register after confirming the TXBF bit value. If writing is not permitted, transmit data cannot be guaranteed when data is written to TXBn register.

(4) Reception buffer registers 0 to 2 (RXB0 to RXB2)

The RXBn register is an 8-bit buffer register for storing parallel data that had been converted by the reception shift register.
When reception is enabled (RXE bit = 1 in the ASIMn register), receive data is transferred from the reception shift register to the RXBn register, synchronized with the completion of the shift-in processing of one frame. Also, a reception completion interrupt request (INTSRn) is generated by the transfer to the RXBn register. For information about the timing for generating this interrupt request, refer to 12.2.5 (4)“Receive operation” on page 326.
If reception is disabled (RXE bit = 0 in the ASIMn register), the contents of the RXBn register are retained, and no processing is performed for transferring data to the RXBn register even when the shift-in processing of one frame is completed. Also, no reception completion interrupt is generated.
When 7 bits is specified for the data length, bits 6 to 0 of the RXBn register are transferred for the receive data and the MSB (bit 7) is always 0. However, if an overrun error (OVE) occurs, the receive data at that time is not transferred to the RXBn register.
Except when a reset is input, the RXBn register becomes FFH even when CAE bit = 0 in the ASIMn register.
This register is read-only in 8-bit or 1-bit units (n = 0 to 2).

Figure 12-5: Reception Buffer Registers 0 to 2 (RXB0 to RXB2)

	7	6	5	4	3	2	1	0	Address	Initial value
RXB0	RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0	FFFFFA02H	FFH

	7	6	5	4	3	2	1	0	Address	Initial value
RXB1	RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0	FFFFFA12H	FFH

	7	6	5	4	3	2	1	0	Address	Initial value
RXB2	RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0	FFFFFA22H	FFH

Bit Position	Bit Name	Function
7 to 0	RXB7 to RXB0	Stores receive data. 0 can be read for RXB7 when 7-bit or character data is received.

(5) Transmission buffer registers 0 to 2 (TXB0 to TXB2)

The TXBn register is an 8-bit buffer register for setting transmit data.
When transmission is enabled (TXE bit = 1 in the ASIMn register), the transmit operation is started by writing data to TXBn register.
When transmission is disabled (TXE bit = 0 in the ASIMn register), even if data is written to TXBn register, the value is ignored.
The TXBn register data is transferred to the transmission shift register, and a transmission completion interrupt request (INTSTn) is generated, synchronized with the completion of the transmission of one frame from the transmission shift register. For information about the timing for generating this interrupt request, refer to 12.2.5 (2)“Transmit operation” on page 322.
When TXBF bit = 1 in the ASIFn register, writing must not be performed to TXBn register.
This register can be read or written in 8-bit or 1-bit units (n = 0 to 2).

Figure 12-6: Transmission Buffer Registers 0 to 2 (TXB0 to TXB2)

	7	6	5	4	3	2	1	0	Address	Initial value
TXB0	TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0	FFFFFA04H	FFH

	7	6	5	4	3	2	1	0	Address	Initial value
TXB1	TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0	FFFFFA14H	FFH

	7	6	5	4	3	2	1	0	Address	Initial value
TXB2	TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0	FFFFFA24H	FFH

Bit Position	Bit Name	Function
7 to 0	TXB7 to TXB0	Writes transmit data.

12.2.4 Interrupt requests

The following three types of interrupt requests are generated from UART0 to UART2.

- Reception error interrupt (INTSERn)
- Reception completion interrupt (INTSRn)
- Transmission completion interrupt (INTSTn)

The default priorities among these three types of interrupt requests is, from high to low, reception error interrupt, reception completion interrupt, and transmission completion interrupt (n = 0 to 2).

Table 12-1: Generated Interrupts and Default Priorities

Interrupt	Priority
Reception error	1
Reception completion	2
Transmission completion	3

(1) Reception error interrupt (INTSER0 to INTSER2)

When reception is enabled, a reception error interrupt is generated according to the logical OR of the three types of reception errors explained for the ASISn register. Whether a reception error interrupt (INTSERn) or a reception completion interrupt (INTSRn) is generated when an error occurs can be specified according to the ISRM bit of the ASIMn register.
When reception is disabled, no reception error interrupt is generated.

(2) Reception completion interrupt (INTSR0 to INTSR2)

When reception is enabled, a reception completion interrupt is generated when data is shifted in to the reception shift register and transferred to the reception buffer register (RXBn).
A reception completion interrupt request can be generated in place of a reception error interrupt according to the ISRM bit of the ASIMn register even when a reception error has occurred.
When reception is disabled, no reception completion interrupt is generated.

(3) Transmission completion interrupt (INTST0 to INTST2)

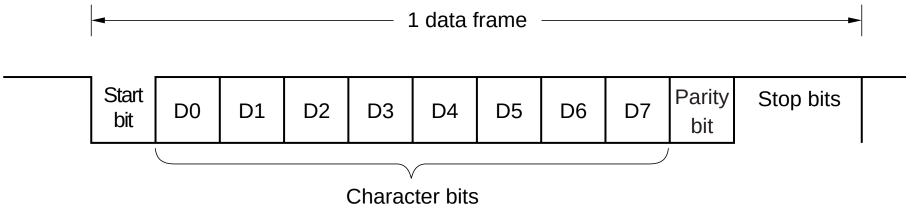
A transmission completion interrupt is generated when one frame of transmit data containing 7-bit or 8-bit characters is shifted out from the transmission shift register.

12.2.5 Operation

(1) Data format

Full-duplex serial data transmission and reception can be performed.
The transmit/receive data format consists of one data frame containing a start bit, character bits, a parity bit, and stop bits as shown in Figure 12-7.
The character bit length within one data frame, the type of parity, and the stop bit length are specified according to the asynchronous serial interface mode register (ASIMn) (n = 0 to 2).
Also, data is transferred with LSB first.

Figure 12-7: Asynchronous Serial Interface Transmit/Receive Data Format



- Start bit ···1 bit
- Character bits ···7 bits or 8 bits
- Parity bit ···Even parity, odd parity, 0 parity, or no parity
- Stop bits ···1 bit or 2 bits

(2) Transmit operation

When CAE bit is set to 1 in the ASIMn register, a high level is output from the TXDn pin.
Then, when TXE bit is set to 1 in the ASIMn register, transmission is enabled, and the transmit operation is started by writing transmit data to transmission buffer register (TXBn) (n = 0 to 2).

(a) Transmission enabled state

This state is set by the TXE bit in the ASIMn register.

- TXE = 1: Transmission enabled state
- TXE = 0: Transmission disabled state

Since UARTn does not have a CTS (transmission enabled signal) input pin, a port should be used to confirm whether the destination is in a reception enabled state.

(b) Starting a transmit operation

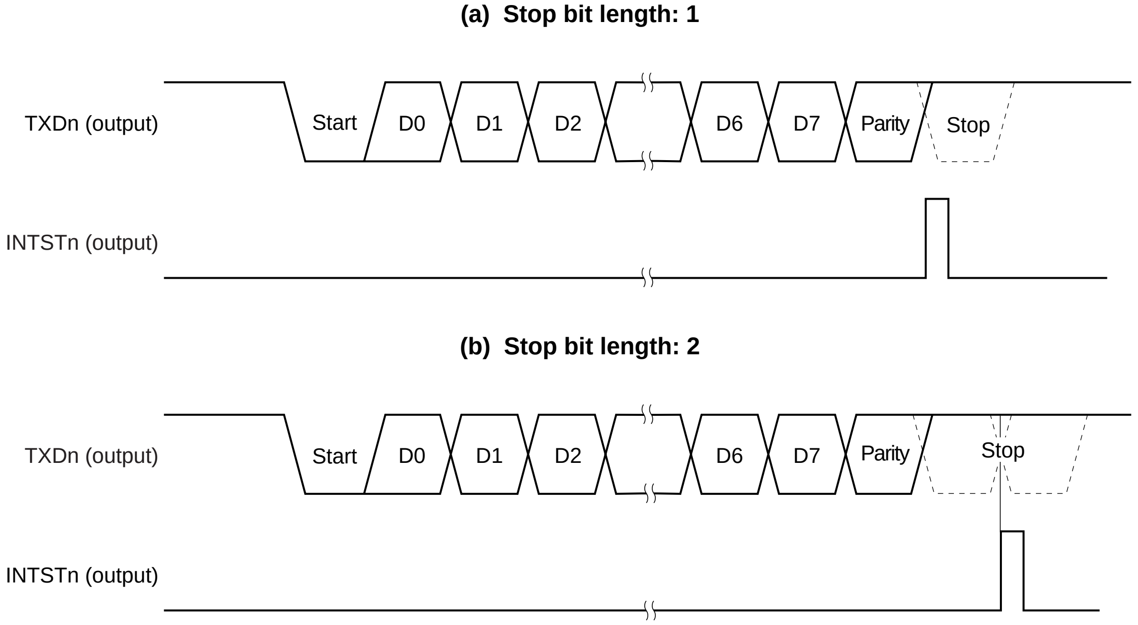
In transmission enabled state, a transmit operation is started by writing transmit data to transmission buffer register (TXBn). When a transmit operation is started, the data in TXBn is transferred to transmission shift register. Then, the transmission shift register outputs data to the TXDn pin (the transmit data is transferred sequential starting with the start bit). The start bit, parity bit, and stop bits are added automatically.

(c) Transmission interrupt request

When the transmission shift register becomes empty, a transmission completion interrupt request (INTSTn) is generated. The timing for generating the INTSTn interrupt differs according to the specification of the number of stop bits. The INTSTn interrupt is generated at the same time that the last stop bit is output.
If the data to be transmitted next has not been written to the TXBn register, the transmit operation is suspended.

Caution: Normally, when the transmission shift register becomes empty, a transmission completion interrupt (INTSTn) is generated. However, no transmission completion interrupt (INTSTn) is generated if the transmission shift register becomes empty due to the input of a $\overline{\text{RESET}}$.

Figure 12-8: Asynchronous Serial Interface Transmission Completion Interrupt Timing



(a) Stop bit length: 1

(b) Stop bit length: 2

(3) Continuous transmission operation

UARTn can write the next transmit data to the TXBn register at the time that the transmission shift register starts the shift operation. This enables an efficient transmission rate to be realized by continuously transmitting data even during the INTSTn interrupt service after the transmission of one data frame.
When continuous transmission is performed, data should be written after referencing the ASIFn register to confirm the transmission status and whether or not data can be written to the TXBn register (n = 0 to 2).

Caution: Transmit data should be written when the TXBF bit is 0. The transmission unit should be initialized when the TXSF bit is 0. If these actions are performed at other times, the transmit data cannot be guaranteed.

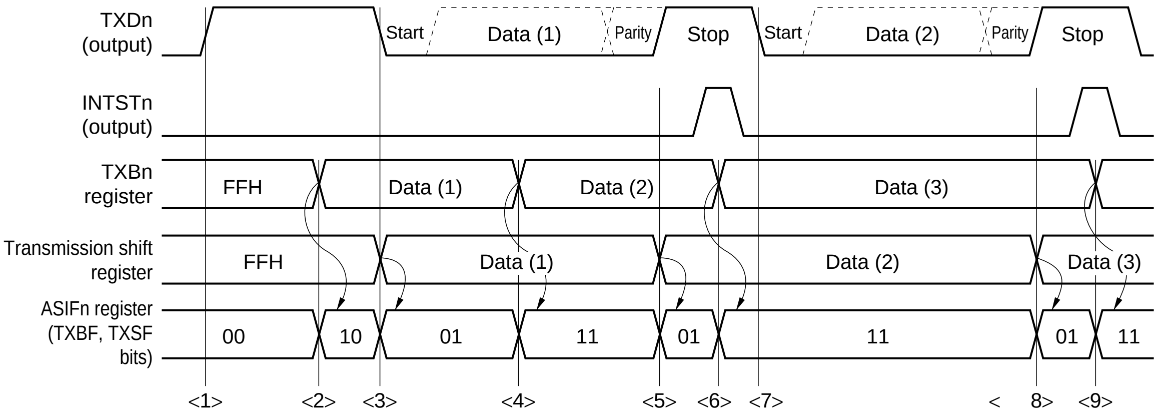
Table 12-2: Transmission Status and Whether or Not Writing Is Enabled

TXBF	TXSF	Transmission Status	Whether or not Write Operation to TXBn is Enabled
0	0	Initial status or transmission completed	Writing is enabled
0	1	Transmission in progress (no data is in TXBn register)	Writing is enabled
1	0	Awaiting transmission (data is in TXBn register)	Writing is not enabled
1	1	Transmission in progress (data is in TXBn register)	Writing is not enabled

(a) Starting procedure

Figure 12-9 shows the procedure to start continuous transmission.

Figure 12-9: Continuous Transmission Starting Procedure

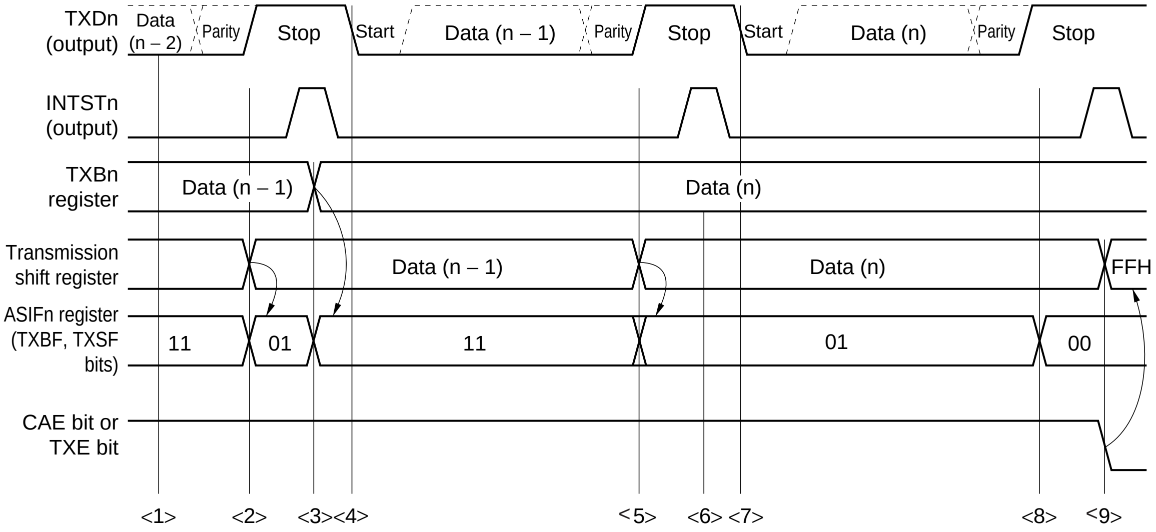


Transmission Starting Procedure	Internal Operation	ASIFn Register	
		TXBF bit	TXSF bit
<1> Set transmission mode		0	0
<2> Write data (1) to TXBn register		1	0
<4> Read ASIFn register (confirm that TXBF bit = 0)	<3> Generate start bit Start data (1) transmission[Note]	0	1
Write data (2)		1	1
<6> Read ASIFn register (confirm that TXBF bit = 0)	<<Transmission in progress>> <5> Generate transmission completion interrupt (INTSTn)	0	1
Write data (3)	<7> Generate start bit Start data (2) transmission	1	1
<9> Read ASIFn register (confirm that TXBF bit = 0)	<<Transmission in progress>> <8> Generate transmission completion interrupt (INTSTn)	0	1
Write data (4)		1	1

Note: For a certain time it may happen that the bit combinations of TXBF and TXSF bits 00B or 11B can be read.

(b) Ending procedure

Figure 12-10: Continuous Transmission End Procedure



Transmission End Procedure	Internal Operation	ASIFn Register	
		TXBF Bit	TXSF Bit
	<1> Transmission of data (n - 2) is in progress	1	1
	<2> Generate transmission completion interrupt (INTSTn)	0	1
<3> Read ASIFn register (confirm that TXBF bit = 0) Write data (n)			
	<4> Generate start bit Start data (n - 1) transmission <<Transmission in progress>>	1	1
	<5> Generate transmission completion interrupt (INTSTn)	0	1
<6> Read ASIFn register (confirm that TXSF bit = 1) There is no write data			
	<7> Generate start bit Start data (n) transmission <<Transmission in progress>>		
	<8> Generate transmission completion interrupt (INTSTn)	0	0
<9> Read ASIFn register (confirm that TXSF bit = 0) Clear (0) the CAE bit or TXE bit of ASIMn register	Initialize internal circuits		

(4) Receive operation

An awaiting reception state is set by setting CAE bit to 1 in the ASIMn register and then setting RXE bit to 1 in the ASIMn register. To start a receive operation, detects a start bit first. The start bit is detected by sampling RXDn pin. When the receive operation begins, serial data is stored sequential in the reception shift register according to the baud rate that was set. A reception completion interrupt (INTSRn) is generated each time the reception of one frame of data is completed. Normally, the receive data is transferred from the reception buffer register (RXBn) to memory by this interrupt servicing (n = 0 to 2).

(a) Reception enabled state

The receive operation is set to reception enabled state by setting the RXE bit in the ASIM0 register to 1.

- RXE bit = 1: Reception enabled state
- RXE bit = 0: Reception disabled state

In reception disabled state, the reception hardware stands by in the initial state. At this time, the contents of the reception buffer register (RXBn) are retained, and no reception completion interrupt or reception error interrupt is generated.

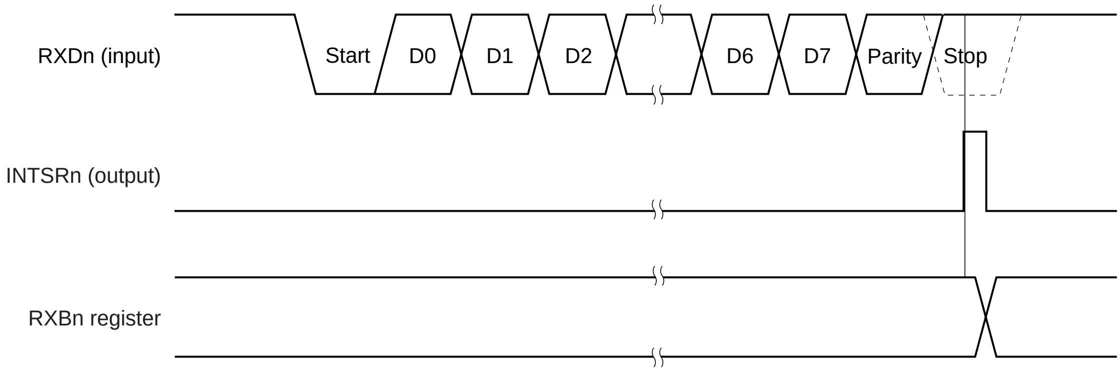
(b) Starting a receive operation

A receive operation is started by the detection of a start bit.
The RXDn pin is sampled according to the serial clock from the dedicated baud rate generator (BRG) of UARTn (n = 0 to 2).

(c) Reception completion interrupt

When RXE bit = 1 in the ASIMn register and the reception of one frame of data is completed (the stop bit is detected), a reception completion interrupt (INTSRn) is generated and the receive data within the reception shift register is transferred to RXBn at the same time.
Also, if an overrun error (OVE) occurs, the receive data at that time is not transferred to the reception buffer register (RXBn), and either a reception completion interrupt (INTSRn) or a reception error interrupt (INTSERn) is generated (the receive data within the reception shift register is transferred to RXBn) according to the ISRM bit setting in the ASIMn register.
Even if a parity error (PE) or framing error (FE) occurs during a reception operation, the receive operation continues until stop bit is received, and after reception is completed, either a reception completion interrupt (INTSRn) or a reception error interrupt (INTSERn) is generated according to the ISRM bit setting in the ASIMn register.
If the RXE bit is reset (0) during a receive operation, the receive operation is immediately stopped. The contents of the reception buffer register (RXBn) and of the asynchronous serial interface status register (ASISn) at this time do not change, and no reception completion interrupt (INTSRn) or reception error interrupt (INTSERn) is generated.
No reception completion interrupt is generated when RXE bit = 0 (reception is disabled).

Figure 12-11: Asynchronous Serial Interface Reception Completion Interrupt Timing



(5) Reception error

The three types of error that can occur during a receive operation are a parity error, framing error, or overrun error. The data reception result is that the various flags of the ASISn register are set (1), and a reception error interrupt (INTSERn) or a reception completion interrupt (INTSRn) is generated at the same time. The ISRM bit of the ASIMn register specifies whether INTSERn or INTSRn is generated.

The type of error that occurred during reception can be detected by reading the contents of the ASISn register during the INTSERn or INTSRn interrupt servicing (n = 0 to 2).

The contents of the ASISn register are reset (0) by reading it.

Table 12-3: Reception Error Causes

Error Flag	Reception Error	Cause
PE	Parity error	The parity specification during transmission did not match the parity of the reception data
FE	Framing error	No stop bit was detected
OVE	Overrun error	The reception of the next data was completed before data was read from the reception buffer register (RXBn)

(a) Separation of reception error interrupt

A reception error interrupt can be separated from the INTSRn interrupt and generated as an INTSERn interrupt by clearing the ISRM bit of the ASIMn register to 0.

Figure 12-12: When Reception Error Interrupt Is Separated from INTSRn Interrupt (ISRM Bit = 0)

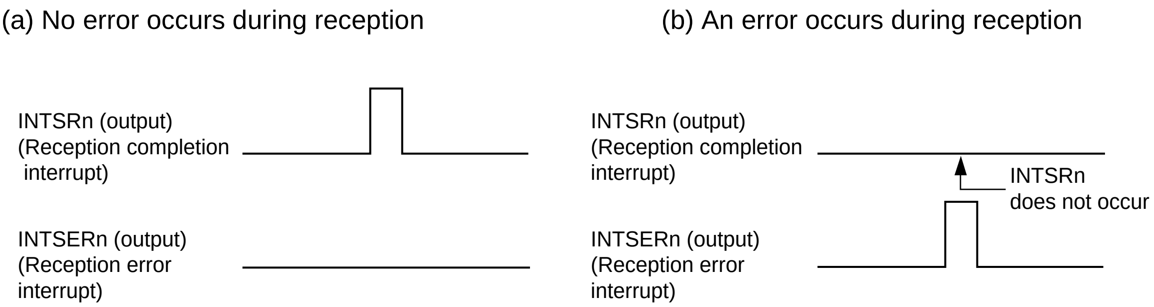
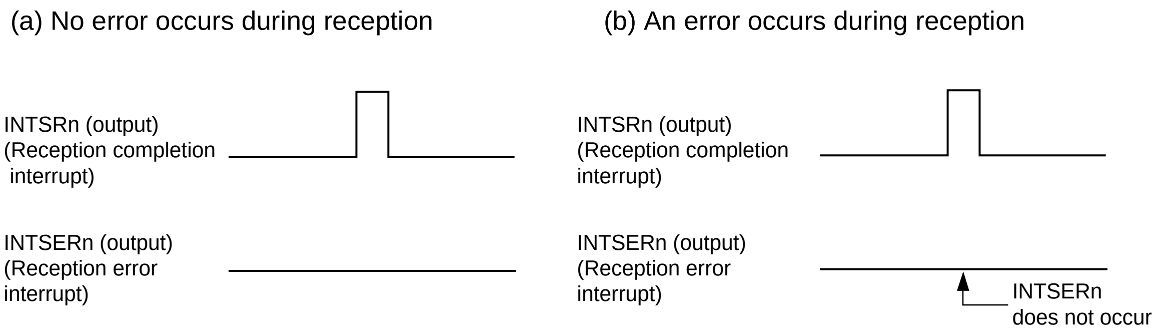


Figure 12-13: When Reception Error Interrupt Is Included in INTSRn Interrupt (ISRM Bit = 1)



(6) Parity types and corresponding operation

A parity bit is used to detect a bit error in communication data. Normally, the same type of parity bit is used at the transmission and reception sides.

(a) Even parity

-During transmission
The parity bit is controlled so that the number of bits with the value “1” within the transmit data including the parity bit is even. The parity bit value is as follows.

• If the number of bits with the value “1” within the transmit data is odd: 1

• If the number of bits with the value “1” within the transmit data is even: 0

-During reception
The number of bits with the value “1” within the receive data including the parity bit is counted, and a parity error is generated if this number is odd.

(b) Odd parity

- During transmission
In contrast to even parity, the parity bit is controlled so that the number of bits with the value “1” within the transmit data including the parity bit is odd. The parity bit value is as follows.

• If the number of bits with the value “1” within the transmit data is odd: 0

• If the number of bits with the value “1” within the transmit data is even: 1

- During reception
The number of bits with the value “1” within the receive data including the parity bit is counted, and a parity error is generated if this number is even.

(c) 0 parity

During transmission the parity bit is set to “0” regardless of the transmit data.
During reception, no parity bit check is performed. Therefore, no parity error is generated regardless of whether the parity bit is “0” or “1”.

(d) No parity

No parity bit is added to the transmit data.
During reception, the receive operation is performed as if there were no parity bit. Since there is no parity bit, no parity error is generated.

(7) Receive data noise filter

The RXDn signal is sampled at the rising edge of the prescaler output basic clock (Clock). If the same sampling value is obtained twice, the match detector output changes, and this output is sampled as input data. Therefore, data not exceeding one clock width is judged to be noise and is not delivered to the internal circuit (see Figure 12-15). Refer to **12.2.6 (1) (a)Basic clock (Clock)** regarding the basic clock.
Also, since the circuit is configured as shown in Figure 12-14, internal processing during a receive operation is delayed by up to 2 clocks according to the external signal status.

Figure 12-14: Noise Filter Circuit

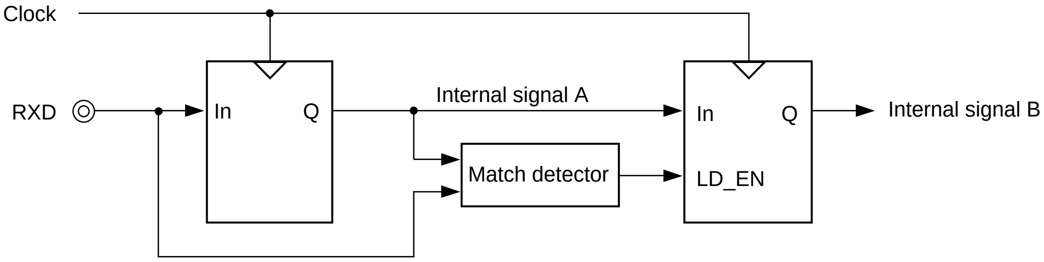
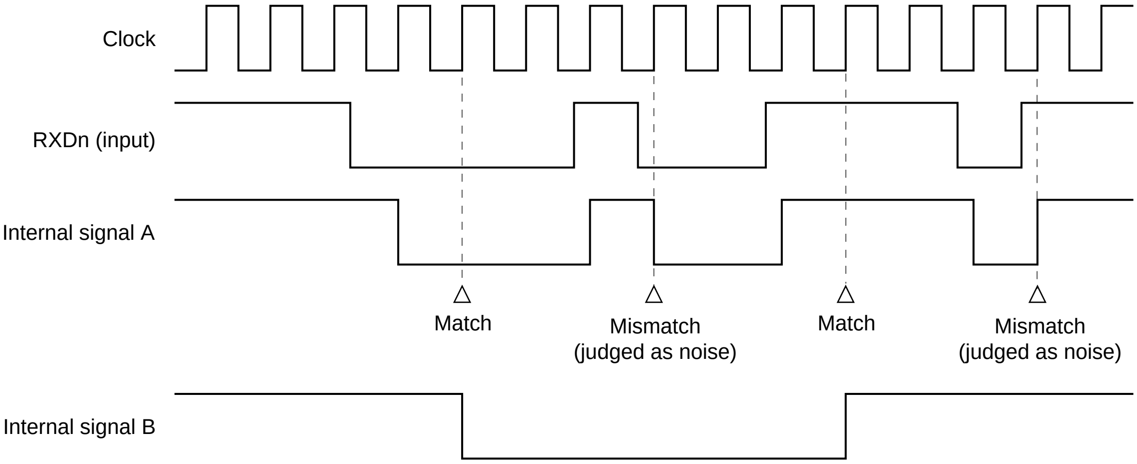


Figure 12-15: Timing of RXDn Signal Judged as Noise



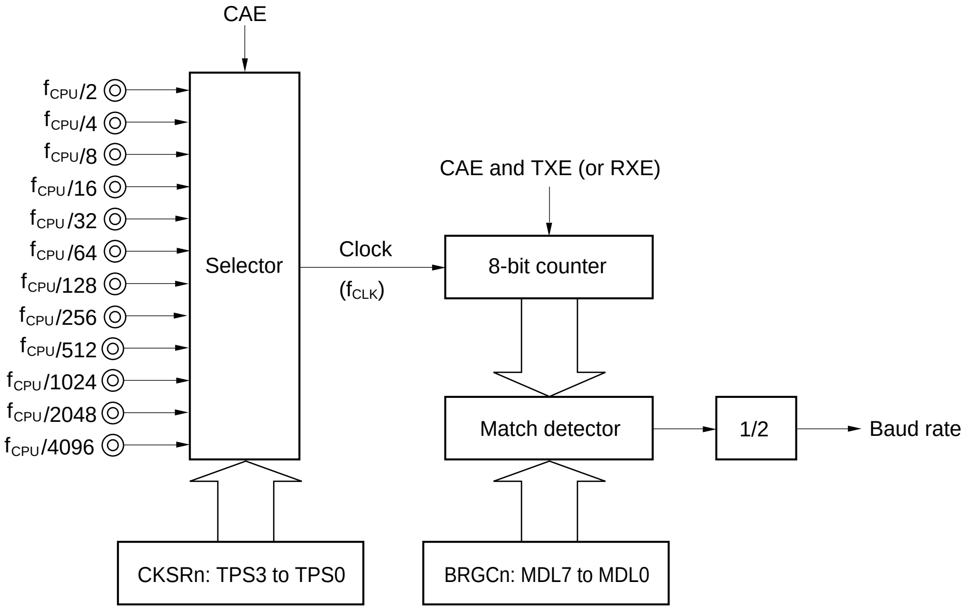
Remark: n = 0 to 2

12.2.6 Dedicated baud rate generators (BRG) of UARTn (n = 0 to 2)

A dedicated baud rate generator, which consists of a source clock selector and an 8-bit programmable counter, generates serial clocks during transmission/reception at UARTn (n = 0 to 2). The dedicated baud rate generator output can be selected as the serial clock for each channel.
Separate 8-bit counters exist for transmission and for reception.

(1) Baud rate generator configuration

Figure 12-16: Baud Rate Generator (BRG) Configuration of UARTn (n = 0 to 2)



Remark: n = 0 to 2

(a) Basic clock (Clock)
When CAE bit = 1 in the ASIMn register, the clock selected according to the TPS3 to TPS0 bits of the CKSRm register is supplied to the transmission/reception unit. This clock is called the basic clock (Clock), and its frequency is referred to as f_{CLK}. When CAE bit = 0, Clock is fixed at low level.

(2) Serial clock generation

A serial clock can be generated according to the settings of the CKSRm and BRGCm registers.
The basic clock to the 8-bit counter is selected according to the TPS3 to TPS0 bits of the CKSRm register.
The 8-bit counter divisor value can be set according to the MDL7 to MDL0 bits of the BRGCm register (m = 0 to 2).

(a) Clock select registers 1 to 3 (CKSR1 to CKSR3)

The CKSRm register is an 8-bit register for selecting the basic block according to the TPS3 to TPS0 bits. The clock selected by the TPS3 to TPS0 bits becomes the basic clock (Clock) of the transmission/ reception module. Its frequency is referred to as f_{CLK}.
This register can be read or written in 8-bit or 1-bit units.

Figure 12-17: Clock Select Registers 1 to 3 (CKSR1 to CKSR3)

	7	6	5	4	3	2	1	0	Address	Initial value
CKSR0	0	0	0	0	TPS3	TPS2	TPS1	TPS0	FFFFFA06H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
CKSR1	0	0	0	0	TPS3	TPS2	TPS1	TPS0	FFFFFA16H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
CKSR2	0	0	0	0	TPS3	TPS2	TPS1	TPS0	FFFFFA26H	00H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>3 to 0</td><td>TPS3 to TPS0</td><td>Specifies the basic clock
<table>
<tr><th>TPS3</th><th>TPS2</th><th>TPS1</th><th>TPS0</th><th>Basic Clock (f_{CLK})</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>$f_{CPU}/2$</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>$f_{CPU}/4$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>$f_{CPU}/8$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>$f_{CPU}/16$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>$f_{CPU}/32$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>$f_{CPU}/64$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>$f_{CPU}/128$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>$f_{CPU}/256$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>$f_{CPU}/512$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>$f_{CPU}/1024$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>$f_{CPU}/2048$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>$f_{CPU}/4096$</td></tr>
<tr><td>1</td><td>1</td><td>Arbitrary</td><td>Arbitrary</td><td>Setting prohibited</td></tr>
</table>
<b>Remark:</b> f_{CPU}: Internal system clock.</td></tr>
</table>

(b) Baud rate generator control registers 0 to 2 (BRGC0 to BRGC2)

The BRGCm register is an 8-bit register that controls the baud rate (serial transfer speed) of UARTn.

This register can be read or written in 8-bit or 1-bit units (m = 0 to 2).

Figure 12-18: Baud Rate Generator Control Registers 0 to 2 (BRGC0 to BRGC2)

	7	6	5	4	3	2	1	0	Address	Initial value
BRGC0	MDL7	MDL6	MDL5	MDL4	MDL3	MDL2	MDL1	MDL0	FFFFFA07H	FFH

	7	6	5	4	3	2	1	0	Address	Initial value
BRGC1	MDL7	MDL6	MDL5	MDL4	MDL3	MDL2	MDL1	MDL0	FFFFFA17H	FFH

	7	6	5	4	3	2	1	0	Address	Initial value
BRGC2	MDL7	MDL6	MDL5	MDL4	MDL3	MDL2	MDL1	MDL0	FFFFFA27H	FFH

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>7 to 0</td><td>MDL7 to MDL0</td><td>
<table>
<tr><th>MDL7</th><th>MDL6</th><th>MDL5</th><th>MDL4</th><th>MDL3</th><th>MDL2</th><th>MDL1</th><th>MDL0</th><th>Divisor Value (k)</th><th>Serial Clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td><td>–</td><td>Setting prohibited</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>8</td><td>$f_{CLK}/8$</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>9</td><td>$f_{CLK}/9$</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>10</td><td>$f_{CLK}/10$</td></tr>
<tr><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>250</td><td>$f_{CLK}/250$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>251</td><td>$f_{CLK}/251$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>252</td><td>$f_{CLK}/252$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>253</td><td>$f_{CLK}/253$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>254</td><td>$f_{CLK}/254$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>255</td><td>$f_{CLK}/255$</td></tr>
</table>
</td></tr>
</table>

Caution: If the MDL7 to MDL0 bits are to be overwritten, TXE bit and RXE bit should be set to 0 in the ASIMn register first.

Remarks: **1.** f_{CLK}:Frequency [Hz] of basic clock selected according to TPS3 to TPS0 bits of CKSRm register

2. k: Value set according to MDL7 to MDL0 bits (k = 8, 9, 10,..., 255)

3. The baud rate is the output clock for the 8-bit counter divided by 2

4. x: don’t care

(c) Baud rate

The baud rate is the value obtained according to the following formula.

$$\text{Baud rate} = \frac{f_{CLK}}{2 \cdot k} \text{ [bps]}$$

f_{CLK} = Frequency [Hz] of basic clock selected according to TPS3 to TPS0 bits of CKSRm register.

k = Value set according to MDL7 to MDL0 bits of BRGCm register (k = 8, 9, 10,..., 255)

(d) Baud rate error

The baud rate error is obtained according to the following formula.

$$\text{Error} = \left(\frac{\text{Actual baud rate (baud rate with error)}}{\text{Desired baud rate (normal baud rate)}} - 1\right) \times 100 \text{ [\%]}$$

Cautions: 1. Make sure that the baud rate error during transmission does not exceed the allowable error of the reception destination.

2. Make sure that the baud rate error during reception is within the allowable baud rate range during reception, which is described in chapter 12.2.6 (3) Allowable baud rate range during reception.

Example: Basic clock frequency = 10 MHz
Settings of MDL7 to MDL0 bits in BRGC0 register = 01000001B (k = 65)
Target baud rate = 76800 bps

Baud rate = $10 \times 10^6 / (2 \times 65)$
= 76923 bps

Error = (76923/76800 - 1) × 100
= 0.160%

(e) Baud rate setting example

Table 12-4: Baud Rate Generator Setting Data

Baud Rate [bps]	f_{CPU} = 20 MHz			f_{CPU} = 16 MHz			f_{CPU} = 5 MHz			f_{CPU} = 4 MHz		
	f_{CLK}	k	ERR	f_{CLK}	k	ERR	f_{CLK}	k	ERR	f_{CLK}	k	ERR
300	$f_{CPU}/256$	130	0.16	$f_{CPU}/256$	104	0.16	$f_{CPU}/64$	130	0.16	$f_{CPU}/64$	104	0.16
600	$f_{CPU}/128$	130	0.16	$f_{CPU}/128$	104	0.16	$f_{CPU}/32$	130	0.16	$f_{CPU}/32$	104	0.16
1200	$f_{CPU}/64$	130	0.16	$f_{CPU}/64$	104	0.16	$f_{CPU}/16$	130	0.16	$f_{CPU}/16$	104	0.16
2400	$f_{CPU}/32$	130	0.16	$f_{CPU}/32$	104	0.16	$f_{CPU}/8$	130	0.16	$f_{CPU}/8$	104	0.16
4800	$f_{CPU}/16$	130	0.16	$f_{CPU}/16$	104	0.16	$f_{CPU}/4$	130	0.16	$f_{CPU}/4$	104	0.16
9600	$f_{CPU}/8$	130	0.16	$f_{CPU}/8$	104	0.16	$f_{CPU}/2$	130	0.16	$f_{CPU}/2$	104	0.16
19200	$f_{CPU}/4$	130	0.16	$f_{CPU}/4$	104	0.16	$f_{CPU}/2$	65	0.16	$f_{CPU}/2$	64	0.16
31250	$f_{CPU}/2$	160	0	$f_{CPU}/2$	128	0	$f_{CPU}/2$	40	0	$f_{CPU}/2$	32	0
38400	$f_{CPU}/2$	130	0.16	$f_{CPU}/2$	104	0.16	$f_{CPU}/2$	33	-1.38	$f_{CPU}/2$	26	0.16
76800	$f_{CPU}/2$	65	0.16	$f_{CPU}/2$	52	0.16	$f_{CPU}/2$	16	1,70	$f_{CPU}/2$	13	0.16
153600	$f_{CPU}/2$	33	-1.38	$f_{CPU}/2$	26	0.16	$f_{CPU}/2$	8	1.70	$f_{CPU}/2$	7	-7.00

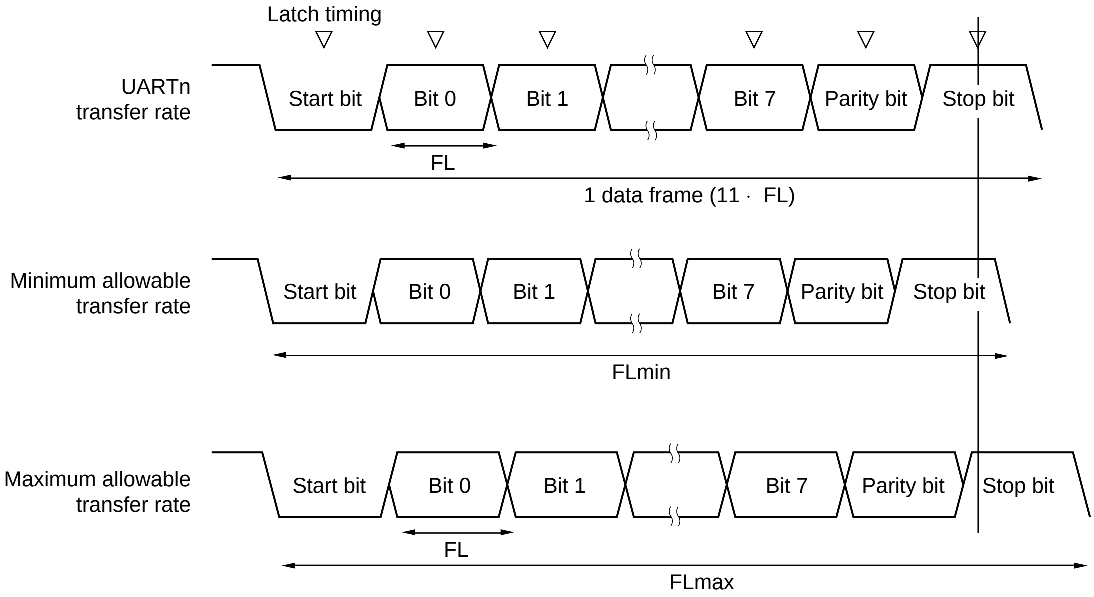
Remark: f_{CPU}: System clock frequency
f_{CLK}: Basic clock frequency
k: Setting values of MDL7 to MDL0 bits in BRGCm register
ERR: Baud rate error [%]

(3) Allowable baud rate range during reception

The degree to which a discrepancy from the transmission destination’s baud rate is allowed during reception is shown below.

Caution: The equations described below should be used to set the baud rate error during reception so that it always is within the allowable error range.

Figure 12-19: Allowable Baud Rate Range During Reception



As shown in Figure 12-19, after the start bit is detected, the receive data latch timing is determined according to the counter that was set by the BRGCm register. If all data up to the final data (stop bit) is in time for this latch timing, the data can be received normally.
Applying this to 11-bit reception is, theoretically, as follows.

$$FL = BR^{-1}$$

BR: UARTn baud rate
k: BRGCm register setting value
FL: 1-bit data length

When the latch timing margin is made 2 basic clocks (Clock), the minimum allowable transfer rate (FLmin) is as follows.

$$FLmin = 11 \times FL - \frac{k-2}{2k} \times FL = \frac{21k+2}{2k} \times FL$$

Therefore, the transfer destination’s maximum baud rate (BRmax) that can be received is as follows.

$$BRmax = \left(\frac{FLmin}{11}\right)^{-1} = \frac{22k}{21k+2} \times BR$$

Similarly, the maximum allowable transfer rate (FLmax) can be obtained as follows.

$$\frac{10}{11} \times \text{FLmax} = 11 \times \text{FL} - \frac{k+2}{2k} \times \text{FL} = \frac{21k-2}{2k} \times \text{FL}$$

$$\text{FLmax} = \frac{21k-2}{20k} \times \text{FL} \times 11$$

Therefore, the transfer destination's minimum baud rate (BRmin) that can be received is as follows.

$$\text{BRmin} = \left(\frac{\text{FLmax}}{11}\right)^{-1} = \frac{20k}{21k+2} \times \text{BR}$$

The allowable baud rate error of UARTn and the transfer destination can be obtained as follows from the expressions described above for computing the minimum and maximum baud rate values.

Table 12-5: Maximum and Minimum Allowable Baud Rate Error

Division Ratio (k)	Maximum Allowable Baud Rate Error	Minimum Allowable Baud Rate Error
8	+3.53%	–3.61%
20	+4.26%	–4.31%
50	+4.56%	–4.58%
100	+4.66%	–4.67%
255	+4.72%	–4.73%

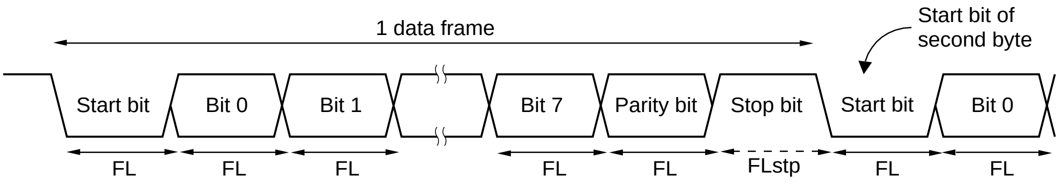
Remarks: **1.** The reception precision depends on the number of bits in one frame, the basic clock frequency, and the division ratio (k). The higher the basic clock frequency and the larger the division ratio (k), the higher the precision.

2. k: BRGCm setting value

(4) Transfer rate during continuous transmission

During continuous transmission, the transfer rate from a stop bit to the next start bit is extended two clocks of basic clock (Clock) longer than normal. However, on the reception side, the transfer result is not affected since the timing is initialized by the detection of the start bit.

Figure 12-20: Transfer Rate During Continuous Transmission



Representing the 1-bit data length by FL, the stop bit length by FLstp, and the basic clock frequency by f_{CLK} fields the following equation.

$$\text{FLstp} = \text{FL} + 2/f_{CLK}$$

Therefore, the transfer rate during continuous transmission is as follows.

$$\text{Transfer rate} = 11 \text{ x FL} = 2/f_{CLK}$$

12.2.7 Precautions

When the supply of clocks to UARTn (n = 0 to 2) is stopped (for example, IDLE or STOP mode), operation stops with each register retaining the value it had immediately before the supply of clocks was stopped. The TXDn pin output also holds and outputs the value it had immediately before the supply of clocks was stopped. However, operation is not guaranteed after the supply of clocks is restarted. Therefore, after the supply of clocks is restarted, the circuits should be initialized by setting CAE bit = 0, RXE bit = 0, and TXE bit = 0 in the ASIMn register.

12.3 Clocked Serial Interfaces 0, 1 (CSI0, CSI1)

12.3.1 Features

- High-speed transfer: Maximum 5 Mbps
- Master mode or slave mode can be selected
- Transmission data length: 8 bits or 16 bits
- Transfer data direction can be switched between MSB first and LSB first
- Eight clock signals can be selected (7 master clocks and 1 slave clock)
- 3-wire type

- SOn	:Serial transmit data output
- SIn	:Serial transmit data input
- $\overline{\text{SCKn}}$	:Serial clock input/output

- Interrupt sources: 1 type
  - Transmission/reception completion interrupt (INTCSIn)
- Transmission/reception mode and reception-only mode can be specified
- Two transmission buffers (SOTBFn/SOTBFLn, SOTBn/SOTBLn) and two reception buffers (SIRBn/SIRBLn, SIRBEn/SIRBELn) are provided on chip
- Single transfer mode and repeat transfer mode can be specified

Remark: n = 0, 1

12.3.2 Configuration

CSIn is controlled via the clocked serial interface mode register (CSIMn) (n = 0, 1).
Transmission/reception of data is performed with reading SIOn register (n = 0, 1).

(1) Clocked serial interface mode registers 0, 1 (CSIM0, CSIM1)

The CSIMn register is an 8-bit register that specifies the operation of CSIn.

(2) Clocked serial interface clock selection registers 0, 1 (CSIC0, CSIC1)

The CSICn register is an 8-bit register that controls the CSIn serial transfer operation.

(3) Serial I/O shift registers 0, 1 (SIO0, SIO1)

The SIOn register is a 16-bit shift register that converts parallel data into serial data.
The SIOn register is used for both transmission and reception.
Data is shifted in (reception) and shifted out (transmission) from the MSB or LSB side.
The actual transmission/reception operations are started up by access of the buffer register.

(4) Serial I/O shift registers L0, L1 (SIOL0, SIOL1)

The SIOLn register is an 8-bit shift register that converts parallel data into serial data.
The SIOLn register is used for both transmission and reception.
Data is shifted in (reception) and shifted out (transmission) from the MSB or LSB side.
The actual transmission/reception operations are started up by access of the buffer register.

(5) Clocked serial interface reception buffer registers 0, 1 (SIRB0, SIRB1)

The SIRBn register is a 16-bit buffer register that stores receive data.

(6) Clocked serial interface reception buffer registers L0, L1 (SIRBL0, SIRBL1)

The SIRBLn register is an 8-bit buffer register that stores receive data.

(7) Clocked serial interface read-only reception buffer registers 0, 1 (SIRBE0, SIRBE1)

The SIRBEn register is a 16-bit buffer register that stores receive data.
The SIRBEn register is the same as the SIRBn register. It is used to read the contents of the SIRBn register.

(8) Clocked serial interface read-only reception buffer registers L0, L1 (SIRBEL0, SIRBEL1)

The SIRBELn register is an 8-bit buffer register that stores receive data.
The SIRBELn register is the same as the lower bytes of the SIRBn register. It is used to read the contents of the SIRBLn register.

(9) Clocked serial interface transmission buffer registers 0, 1 (SOTB0, SOTB1)

The SOTBn register is a 16-bit buffer register that stores transmit data.

(10) Clocked serial interface transmission buffer registers L0, L1 (SOTBL0, SOTBL1)

The SOTBLn register is an 8-bit buffer register that stores transmit data.

(11) Clocked serial interface initial transmission buffer registers 0, 1 (SOTBF0, SOTBF1)

The SOTBFn register is a 16-bit buffer register that stores the initial transmit data in the repeat transfer mode.

(12) Clocked serial interface initial transmission buffer registers L0, L1 (SOTBFL0, SOTBFL1)

The SOTBFLn register is an 8-bit buffer register that stores initial transmit data in the repeat transfer mode.

(13) Selector

The selector selects the serial clock to be used.

(14) Serial clock control circuit

Controls the serial clock supply to the shift register. Also controls the clock output to the $\overline{\text{SCKn}}$ pin when the internal clock is used.

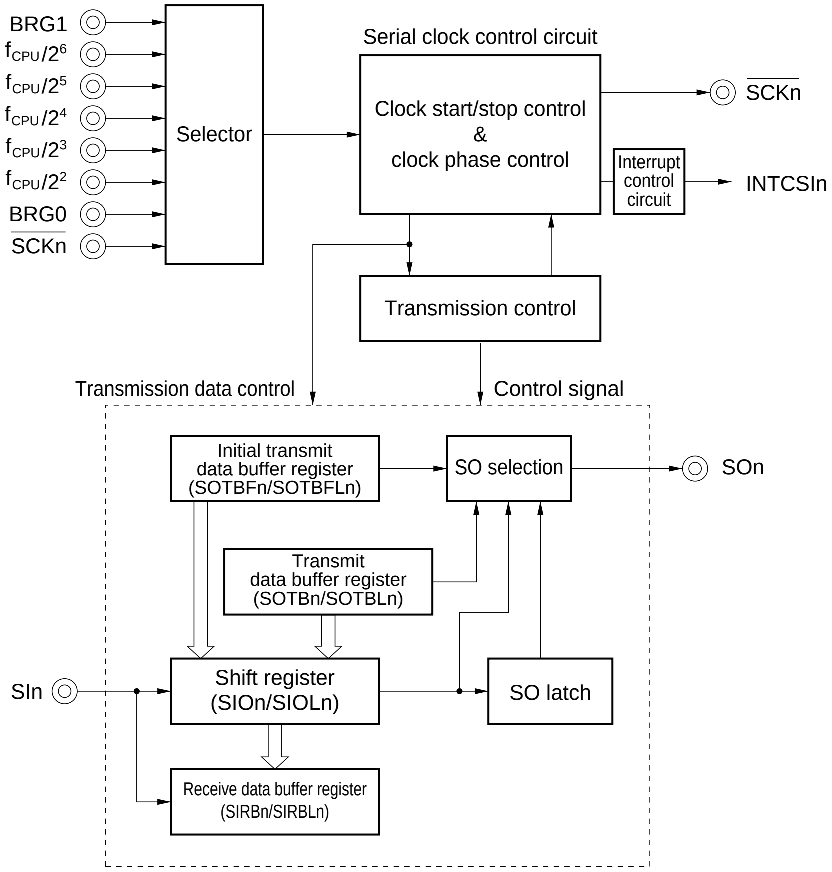
(15) Serial clock counter

Counts the serial clock output or input during transmission/reception operation, and checks whether 8-bit data transmission/reception has been performed.

(16) Interrupt control circuit

Controls the interrupt request timing.

Figure 12-21: Block Diagram of Clocked Serial Interfaces


BRG1
$f_{CPU}/2^6$
$f_{CPU}/2^5$
$f_{CPU}/2^4$
$f_{CPU}/2^3$
$f_{CPU}/2^2$
BRG0
$\overline{\text{SCKn}}$
Selector
Serial clock control circuit
Clock start/stop control & clock phase control
$\overline{\text{SCKn}}$
Interrupt control circuit
INTCSIn
Transmission control
Transmission data control
Control signal
Initial transmit data buffer register (SOTBFn/SOTBFLn)
SO selection
SOn
Transmit data buffer register (SOTBn/SOTBLn)
SIn
Shift register (SIOn/SIOLn)
SO latch
Receive data buffer register (SIRBn/SIRBLn)

Remark: n = 0, 1

12.3.3 Control registers

(1) Clocked serial interface mode registers 0, 1 (CSIM0, CSIM1)

The CSIMn register controls the CSIn operation (n = 0, 1).
These registers can be read/written in 8-bit or 1-bit units (however, bit 0 is read-only).

Figure 12-22: Clocked Serial Interface Mode Registers 0, 1 (CSIM0, CSIM1)

	7	6	5	4	3	2	1	0	Address	Initial value
CSIM0	CSIE	TRMD	CCL	DIR	CSIT	AUTO	0	CSOT	FFFFF900H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
CSIM1	CSIE	TRMD	CCL	DIR	CSIT	AUTO	0	CSOT	FFFFF910H	00H

Bit Position	Bit Name	Function
7	CSIE	Enables/disables CSIn operation. 0: Enable CSIn operation. 1: Disable CSIn operation. The internal CSIn circuit can be reset asynchronously by setting the CSIE bit to 0. For the $\overline{\text{SCKn}}$ and SOn pin output status when the CSIE bit = 0, refer to **12.3.5 Output pins**.
6	TRMD	Specifies transmission/reception mode. 0: Receive-only mode 1: Transmission/reception mode When the TRMD bit = 0, receive-only transfer is performed and the SOn pin output is fixed to low level. Data reception is started by reading the SIRBn register. When the TRMD bit = 1, transmission/reception is started by writing data to the SOTBn register.
5	CCL	Specifies data length. 0: 8 bits 1: 16 bits
4	DIR	Specifies transfer direction mode (MSB/LSB). 0: First bit of transfer data is MSB 1: First bit of transfer data is LSB
3	CSIT	Controls delay of interrupt request signal. 0: No delay 1: Delay mode (interrupt request signal is delayed 1/2 cycle). **Caution: The delay mode (CSIT bit = 1) is effective only in the master mode (CKS2 to CSK0 bits of the CSICn register are not 111B). In the slave mode (CKS2 to CKS0 bits are 111B), do not set the delay mode.**
2	AUTO	Specifies single transfer mode or repeat transfer mode. 0: Single transfer mode 1: Repeat transfer mode
0	CSOT	Flag indicating transfer status. 0: Idle status 1: Transfer execution status **Caution: The CSOT bit is cleared (0) by writing 0 to the CSIE bit.**

Remark: n = 0, 1

Caution: Overwriting the TRMD, CCL, DIR, CSIT, and AUTO bits of the CSIMn register can be done only when the CSOT bit = 0. If these bits are overwritten at any other time, the operation cannot be guaranteed.

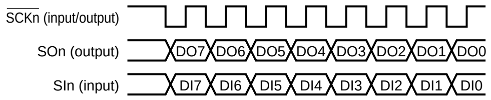
(2) Clocked serial interface clock selection registers 0, 1 (CSIC0, CSIC1)

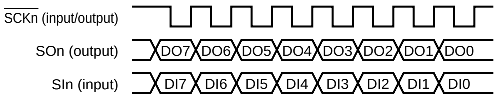
The CSICn register is an 8-bit register that controls the CSIn transfer operation (n = 0, 1).
This register can be read/written in 8-bit or 1-bit units.

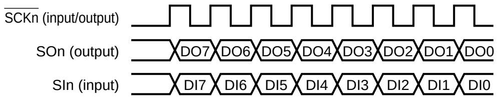
Figure 12-23: Clocked Serial Interface Clock Selection Registers 0, 1 (CSIC0, CSIC1)

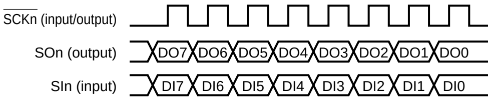
	7	6	5	4	3	2	1	0	Address	Initial value
CSIC0	0	0	0	CKP	DAP	CKS2	CKS1	CKS0	FFFFF901H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
CSIC1	0	0	0	CKP	DAP	CKS2	CKS1	CKS0	FFFFF911H	00H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>4, 3</td><td>CKP, DAP</td><td>Specifies operation mode
<table>
<tr><th>CKP</th><th>DAP</th><th>Operation Mode</th></tr>
<tr><td>0</td><td>0</td><td>
SCKn (input/output)
SOn (output)
SIn (input)
DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0
DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0

</td></tr>
<tr><td>0</td><td>1</td><td>
SCKn (input/output)
SOn (output)
SIn (input)
DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0
DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0

</td></tr>
<tr><td>1</td><td>0</td><td>
SCKn (input/output)
SOn (output)
SIn (input)
DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0
DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0

</td></tr>
<tr><td>1</td><td>1</td><td>
SCKn (input/output)
SOn (output)
SIn (input)
DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0
DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0

</td></tr>
</table>
Remark: n = 0, 1</td></tr>
<tr><td>2 to 0</td><td>CKS2 to CKS0</td><td>Specifies input clock
<table>
<tr><th>CKS2</th><th>CKS1</th><th>CKS0</th><th>Input Clock</th><th>Mode</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>f_{CPU} /4</td><td>Master mode</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Internal BRG Channel 0</td><td>Master mode</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Internal BRG Channel 1</td><td>Master mode</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>f_{CPU} /8</td><td>Master mode</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>f_{CPU} /16</td><td>Master mode</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>f_{CPU} /32</td><td>Master mode</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>f_{CPU} /64</td><td>Master mode</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>External clock (SCKn)</td><td>Slave mode</td></tr>
</table>
Remarks: **1.** f_{CPU}: Internal system clock frequency.
2. n = 0, 1</td></tr>
</table>

Caution: The CSICn register can be overwritten only when the CSIE bit of the CSIMn register = 0.

(3) Clocked serial interface reception buffer registers 0, 1 (SIRB0, SIRB1)

The SIRBn register is a 16-bit buffer register that stores receive data (n = 0, 1).
When the receive-only mode is set (TRMD bit of CSIMn register = 0), the reception operation is started by reading data from the SIRBn register.
These registers are read-only, in 16-bit units.
In addition to reset input, these registers can also be initialized by clearing (0) the CSIE bit of the CSIMn register.

Figure 12-24: Clocked Serial Interface Reception Buffer Registers 0, 1 (SIRB0, SIRB1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SIRB0	SIRB15	SIRB14	SIRB13	SIRB12	SIRB11	SIRB10	SIRB9	SIRB8	SIRB7	SIRB6	SIRB5	SIRB4	SIRB3	SIRB2	SIRB1	SIRB0	FFFFF902H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SIRB1	SIRB15	SIRB14	SIRB13	SIRB12	SIRB11	SIRB10	SIRB9	SIRB8	SIRB7	SIRB6	SIRB5	SIRB4	SIRB3	SIRB2	SIRB1	SIRB0	FFFFF912H	0000H

Bit Position	Bit Name	Function
15 to 0	SIRB15 to SIRB0	Store receive data.

Cautions: 1. Read the SIRBn register only when the 16-bit data length has been set (CCL bit of CSIMn register = 1).

2. When the single transfer mode has been set (AUTO bit of CSIMn register = 0), perform read operation only in the idle state (CSOT bit of CSIMn register = 0). If the SIRBn register is read during data transfer, the data cannot be guaranteed.

(4) Clocked serial interface reception buffer registers L0, L1 (SIRBL0, SIRBL1)

The SIRBLn register is an 8-bit buffer register that stores receive data (n = 0, 1).
When the receive-only mode is set (TRMD bit of CSIMn register = 0), the reception operation is started by reading data from the SIRBLn register.
These registers are read-only, in 8-bit units.
In addition to reset input, these registers can also be initialized by clearing (0) the CSIE bit of the CSIMn register.
The SIRBLn register is the same as the lower bytes of the SIRBn register.

Figure 12-25: Clocked Serial Interface Reception Buffer Registers L0, L1 (SIRBL0, SIRBL1)

	7	6	5	4	3	2	1	0	Address	Initial value
SIRBL0	SIRB7	SIRB6	SIRB5	SIRB4	SIRB3	SIRB2	SIRB1	SIRB0	FFFFF902H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
SIRBL1	SIRB7	SIRB6	SIRB5	SIRB4	SIRB3	SIRB2	SIRB1	SIRB0	FFFFF912H	00H

Bit Position	Bit Name	Function
7 to 0	SIRB7 to SIRB0	Stores receive data.

Cautions: 1. Read the SIRBLn register only when the 8-bit data length has been set (CCL bit of CSIMn register = 0).

2. When the single transfer mode is set (AUTO bit of CSIMn register = 0), perform read operation only in the idle state (CSOT bit of CSIMn register = 0). If the SIRBLn register is read during data transfer, the data cannot be guaranteed.

(5) Clocked serial interface read-only reception buffer registers 0, 1 (SIRBE0, SIRBE1)

The SIRBEn register is a 16-bit buffer register that stores receive data (n = 0, 1).
These registers are read-only, in 16-bit units.
In addition to reset input, this register can also be initialized by clearing (0) the CSIE bit of the CSIMn register.
The SIRBEn register is the same as the SIRBn register. It is used to read the contents of the SIRBn register.

Figure 12-26: Clocked Serial Interface Read-Only Reception Buffer Registers 0, 1 (SIRBE0, SIRBE1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SIRBE0	SIRBE15	SIRBE14	SIRBE13	SIRBE12	SIRBE11	SIRBE10	SIRBE9	SIRBE8	SIRBE7	SIRBE6	SIRBE5	SIRBE4	SIRBE3	SIRBE2	SIRBE1	SIRBE0	FFFFF906H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SIRBE1	SIRBE15	SIRBE14	SIRBE13	SIRBE12	SIRBE11	SIRBE10	SIRBE9	SIRBE8	SIRBE7	SIRBE6	SIRBE5	SIRBE4	SIRBE3	SIRBE2	SIRBE1	SIRBE0	FFFFF916H	0000H

Bit Position	Bit Name	Function
15 to 0	SIRBE15 to SIRBE0	Store receive data.

Cautions: 1. The receive operation is not started even if data is read from the SIRBEn register

2. The SIRBEn register can be read only if the 16-bit data length is set (CCL bit of CSIMn register = 1).

(6) Clocked serial interface read-only reception buffer registers L0, L1 (SIRBEL0, SIRBEL1)

The SIRBELn register is an 8-bit buffer register that stores receive data (n = 0, 1).
These registers are read-only, in 8-bit units.
In addition to reset input, this register can also be initialized by clearing (0) the CSIE bit of the CSIMn register.
The SIRBELn register is the same as the lower byte of the SIRBn register. It is used to read the contents of the SIRBLn register.

Figure 12-27: Clocked Serial Interface Read-Only Reception Buffer Registers L0, L1(SIRBEL0, SIRBEL1)

	7	6	5	4	3	2	1	0	Address	Initial value
SIRBEL0	SIRBE7	SIRBE6	SIRBE5	SIRBE4	SIRBE3	SIRBE2	SIRBE1	SIRBE0	FFFFF906H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
SIRBEL1	SIRBE7	SIRBE6	SIRBE5	SIRBE4	SIRBE3	SIRBE2	SIRBE1	SIRBE0	FFFFF916H	00H

Bit Position	Bit Name	Function
7 to 0	SIRBE7 to SIRBE0	Store receive data.

Cautions: 1. The receive operation is not started even if data is read from the SIRBELn register.
2. The SIRBELn register can be read only if the 8-bit data length has been set (CCL bit of CSIMn register = 0).

(7) Clocked serial interface transmission buffer registers 0, 1 (SOTB0, SOTB1)

The SOTBn register is a 16-bit buffer register that stores transmit data (n = 0, 1).
When the transmission/reception mode is set (TRMD bit of CSIMn register = 1), the transmission operation is started by writing data to the SOTBn register.
This register can be read/written in 16-bit units.

Figure 12-28: Clocked Serial Interface Transmission Buffer Registers 0, 1 (SOTB0, SOTB1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SOTB0	SOTB15	SOTB14	SOTB13	SOTB12	SOTB11	SOTB10	SOTB9	SOTB8	SOTB7	SOTB6	SOTB5	SOTB4	SOTB3	SOTB2	SOTB1	SOTB0	FFFFF904H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SOTB1	SOTB15	SOTB14	SOTB13	SOTB12	SOTB11	SOTB10	SOTB9	SOTB8	SOTB7	SOTB6	SOTB5	SOTB4	SOTB3	SOTB2	SOTB1	SOTB0	FFFFF914H	0000H

Bit Position	Bit Name	Function
15 to 0	SOTB15 to SOTB0	Store transmit data.

Cautions: 1. Access the SOTBn register only when the 16-bit data length is set (CCL bit of CSIMn register = 1).

2. When the single transfer mode is set (AUTO bit of CSIMn register = 0), perform access only in the idle state (CSOT bit of CSIMn register = 0). If the SOTBn register is accessed during data transfer, the data cannot be guaranteed.

(8) Clocked serial interface transmission buffer registers L0, L1 (SOTBL0, SOTBL1)

The SOTBLn register is an 8-bit buffer register that stores transmit data (n = 0, 1).
When the transmission/reception mode is set (TRMD bit of CSIMn register = 1), the transmission operation is started by writing data to the SOTBLn register.
These registers can be read/written in 8-bit units.
The SOTBLn register is the same as the lower bytes of the SOTBn register.

Figure 12-29: Clocked Serial Interface Transmission Buffer Registers L0, L1 (SOTBL0, SOTBL1)

	7	6	5	4	3	2	1	0	Address	Initial value
SOTBL0	SOTB7	SOTB6	SOTB5	SOTB4	SOTB3	SOTB2	SOTB1	SOTB0	FFFFF904H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
SOTBL1	SOTB7	SOTB6	SOTB5	SOTB4	SOTB3	SOTB2	SOTB1	SOTB0	FFFFF914H	00H

Bit Position	Bit Name	Function
7 to 0	SOTB7 to SOTB0	Store transmit data.

Cautions: 1. Access the SOTBLn register only when the 8-bit data length has been set (CCL bit of CSIMn register = 0).

2. When the single transfer mode is set (AUTO bit of CSIMn register = 0), perform access only in the idle state (CSOT bit of CSIMn register = 0). If the SOTBLn register is accessed during data transfer, the data cannot be guaranteed.

(9) Clocked serial interface initial transmission buffer registers 0, 1 (SOTBF0, SOTBF1)

The SOTBFn register is a 16-bit buffer register that stores initial transmission data in the repeat transfer mode (n = 0, 1).
The transmission operation is not started even if data is written to the SOTBFn register.
These registers can be read/written in 16-bit units.

Figure 12-30: Clocked Serial Interface Initial Transmission Buffer Registers 0, 1 (SOTBF0, SOTBF1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SOTBF0	SOTBF15	SOTBF14	SOTBF13	SOTBF12	SOTBF11	SOTBF10	SOTBF9	SOTBF8	SOTBF7	SOTBF6	SOTBF5	SOTBF4	SOTBF3	SOTBF2	SOTBF1	SOTBF0	FFFFF908H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SOTBF1	SOTBF15	SOTBF14	SOTBF13	SOTBF12	SOTBF11	SOTBF10	SOTBF9	SOTBF8	SOTBF7	SOTBF6	SOTBF5	SOTBF4	SOTBF3	SOTBF2	SOTBF1	SOTBF0	FFFFF918H	0000H

Bit Position	Bit Name	Function
15 to 0	SOTBF15 to SOTBF0	Stores initial transmission data in repeat transfer mode.

Caution: Access the SOTBFn register only when the 16-bit data length has been set (CCL bit of CSIMn register = 1), and only in the idle state (CSOT bit of CSIMn register = 0). If the SOTBFn register is accessed during data transfer, the data cannot be guaranteed.

(10) Clocked serial interface initial transmission buffer registers L0, L1 (SOTBFL0, SOTBFL1)

The SOTBFLn register is an 8-bit buffer register that stores initial transmission data in the repeat transfer mode (n = 0, 1).

The transmission operation is not started even if data is written to the SOTBFLn register.

These registers can be read/written in 8-bit units.

The SOTBFLn register is the same as the lower bytes of the SOTBFn register.

Figure 12-31: Clocked Serial Interface Initial Transmission Buffer Registers L0, L1 (SOTBFL0, SOTBFL1)

	7	6	5	4	3	2	1	0	Address	Initial value
SOTBFL0	SOTBF7	SOTBF6	SOTBF5	SOTBF4	SOTBF3	SOTBF2	SOTBF1	SOTBF0	FFFFF908H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
SOTBFL1	SOTBF7	SOTBF6	SOTBF5	SOTBF4	SOTBF3	SOTBF2	SOTBF1	SOTBF0	FFFFF918H	00H

Bit Position	Bit Name	Function
7 to 0	SOTBF7 to SOTBF0	Store initial transmission data in repeat transfer mode.

Caution: Access the SOTBFLn register only when the 8-bit data length has been set (CCL bit of CSIM0 register = 0), and only in the idle state (CSOT bit of CSIMn register = 0). If the SOTBFLn register is accessed during data transfer, the data cannot be guaranteed.

(11) Serial I/O shift registers 0, 1 (SIO0, SIO1)

The SIOn register is a 16-bit shift register that converts parallel data into serial data (n = 0, 1).

The transfer operation is not started even if the SIOn register is read.

These registers are read-only, in 16-bit units.

In addition to reset input, this register can also be initialized by clearing (0) the CSIE bit of the CSIMn register.

Figure 12-32: Serial I/O Shift Registers 0, 1 (SIO0, SIO1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SIO0	SIO15	SIO14	SIO13	SIO12	SIO11	SIO10	SIO9	SIO8	SIO7	SIO6	SIO5	SIO4	SIO3	SIO2	SIO1	SIO0	FFFFF90AH	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SIO1	SIO15	SIO14	SIO13	SIO12	SIO11	SIO10	SIO9	SIO8	SIO7	SIO6	SIO5	SIO4	SIO3	SIO2	SIO1	SIO0	FFFFF91AH	0000H

Bit Position	Bit Name	Function
15-0	SIO15 to SIO0	Data is shifted in (reception) or shifted out (transmission) from the MSB or LSB side.

Caution: Access the SIOn register only when the 16-bit data length has been set (CCL bit of CSIMn register = 1), and only in the idle state (CSOT bit of CSIMn register = 0). If the SIOn register is accessed during data transfer, the data cannot be guaranteed.

(12) Serial I/O shift registers L0, L1 (SIOL0, SIOL1)

The SIOLn register is an 8-bit shift register that converts parallel data into serial data (n = 0, 1).
The transfer operation is not started even if the SIOLn register is read.
These registers are read-only, in 8-bit units.
In addition to reset input, this register can also be initialized by clearing (0) the CSIE bit of the CSIMn register.
The SIOLn register is the same as the lower bytes of the SIOn register.

Figure 12-33: Serial I/O Shift Registers L0, L1 (SIOL0, SIOL1)

	7	6	5	4	3	2	1	0	Address	Initial value
SIOL0	SIO7	SIO6	SIO5	SIO4	SIO3	SIO2	SIO1	SIO0	FFFFF90AH	00H

	7	6	5	4	3	2	1	0	Address	Initial value
SIOL1	SIO7	SIO6	SIO5	SIO4	SIO3	SIO2	SIO1	SIO0	FFFFF91AH	00H

Bit Position	Bit Name	Function
7 to 0	SIO7 to SIO0	Data is shifted in (reception) or shifted out (transmission) from the MSB or LSB side.

Caution: Access the SIOLn register only when the 8-bit data length has been set (CCL bit of CSIMn register = 0), and only in the idle state (CSOT bit of CSIMn register = 0). If the SIOLn register is accessed during data transfer, the data cannot be guaranteed.

12.3.4 Operation

(1) Single transfer mode

(a) Usage

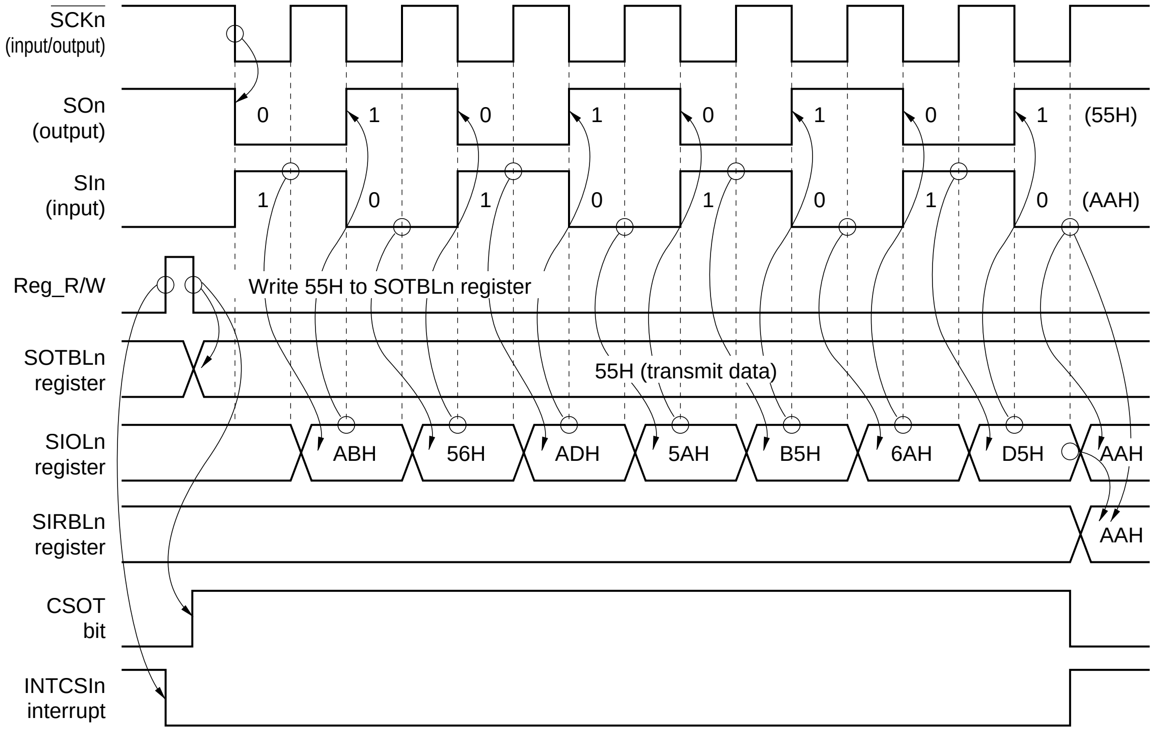
In the receive-only mode (TRMD bit of CSIMn register = 0), transfer is started by reading[Note 1] the receive data buffer register (SIRBn/SIRBLn) (n = 0, 1).
In the transmission/reception mode (TRMD bit of CSIMn register = 1), transfer is started by writing[Note 2] to the transmit data buffer register (SOTBn/SOTBLn).
In the slave mode, the operation must be enabled beforehand (CSIE bit of CSIMn register = 1).
When transfer is started, the value of the CSOT bit of the CSIMn register becomes 1 (transmission execution status).
Upon transfer completion, the transmission/reception completion interrupt (INTCSIn) is set (1), and the CSOT bit is cleared (0). The next data transfer request is then waited for.

Notes: **1.** When the 16-bit data length (CCL bit of CSIMn register = 1) has been set, read the SIRBn register. When the 8-bit data length (CCL bit of CSIMn register = 0) has been set, read the SIRBLn register.
2. When the 16-bit data length (CCL bit of CSIMn register = 1) has been set, write to the SOTBn register. When the 8-bit data length (CCL bit of CSIMn register = 0) has been set, write to the SOTBLn register.

Caution: When the CSOT bit of the CSIMn register = 1, do not manipulate the CSIn register.

Figure 12-34: Timing Chart in Single Transfer Mode (1/2)

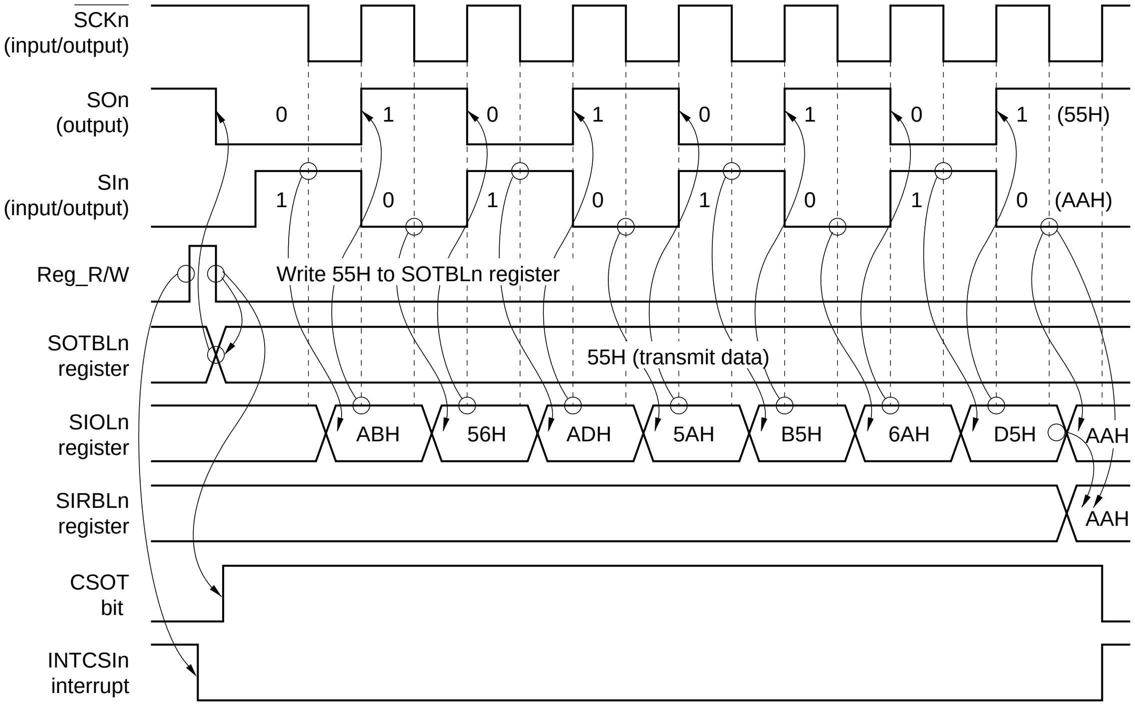
(a) In transmission/reception mode, data length: 8 bits, transfer direction: MSB first, no interrupt delay, single transfer mode, operation mode: CKP bit = 0, DAP bit = 0



Remarks: **1.** n = 0, 1
2. Reg_R/W:Internal signal. This signal indicates that receive data buffer register (SIRBn/ SIRBLn) read or transmit data buffer register (SOTBn/SOTBLn) write was performed.

Figure 12-34: Timing Chart in Single Transfer Mode (2/2)

(b) In transmission/reception mode, data length: 8 bits, transfer direction: MSB first, no interrupt delay, single transfer mode, operation mode: CKP bit = 0, DAP bit = 1



Remarks: **1.** n = 0, 1

2. Reg_R/W:Internal signal. This signal indicates that receive data buffer register (SIRBn/SIRBLn) read or transmit data buffer register (SOTBn/SOTBLn) write was performed.

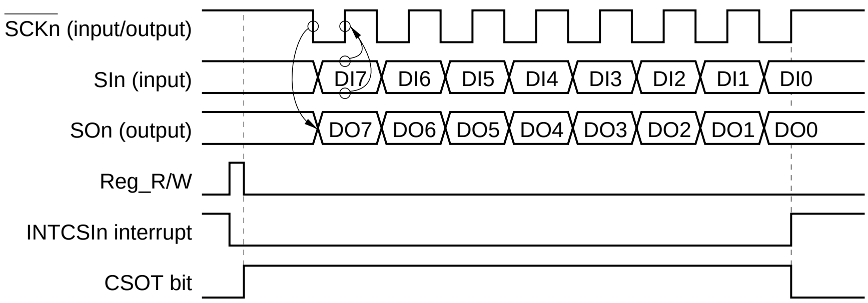
(b) Clock phase selection

The following shows the timing when changing the conditions for clock phase selection (CKP bit of CSICn register) and data phase selection (DAP bit of CSICn register) under the following conditions.

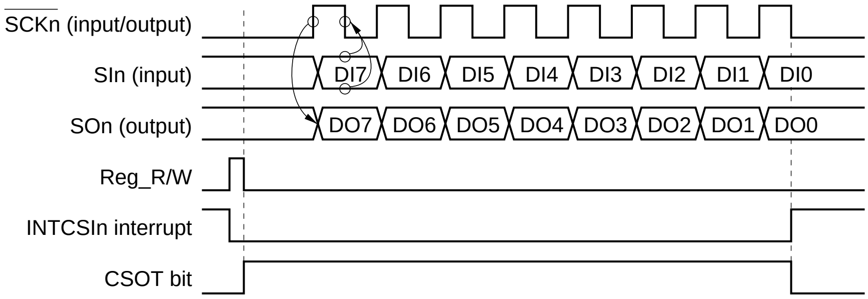
- Data length = 8 bits (CCL bit of CSIMn register = 0)
- First bit of transfer data = MSB (DIR bit of CSIMn register = 0)
- No interrupt request signal delay control (CSIT bit of CSIMn register = 0)

Figure 12-35: Timing Chart According to Clock Phase Selection (1/2)

(a) When CKP bit = 0, DAP bit = 0



(b) When CKP bit = 1, DAP bit = 0

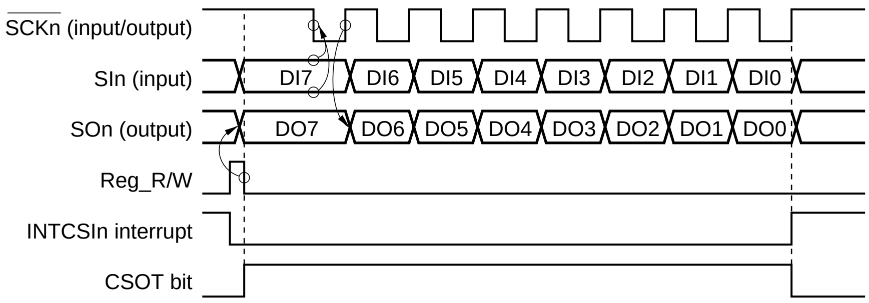


Remarks: **1.** n = 0, 1

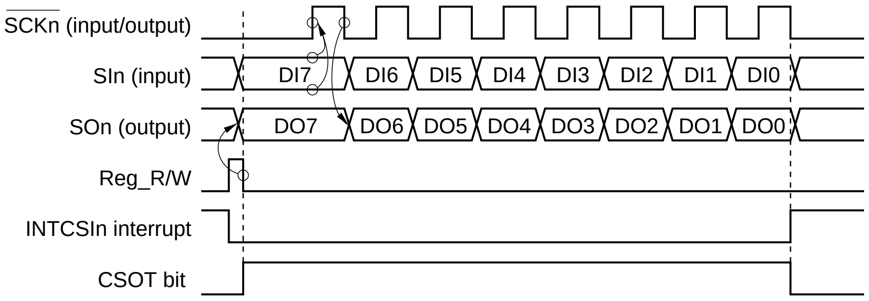
2. Reg_R/W:Internal signal. This signal indicates that receive data buffer register (SIRBn/SIRBLn) read or transmit data buffer register (SOTBn/SOTBLn) write was performed.

Figure 12-35: Timing Chart According to Clock Phase Selection (2/2)

(c) When CKP bit = 0, DAP bit = 1



(d) When CKP bit = 1, DAP bit = 1



Remarks: **1.** n = 0, 1

2. Reg_R/W:Internal signal. This signal indicates that receive data buffer register (SIRBn/SIRBLn) read or transmit data buffer register (SOTBn/SOTBLn) write was performed.

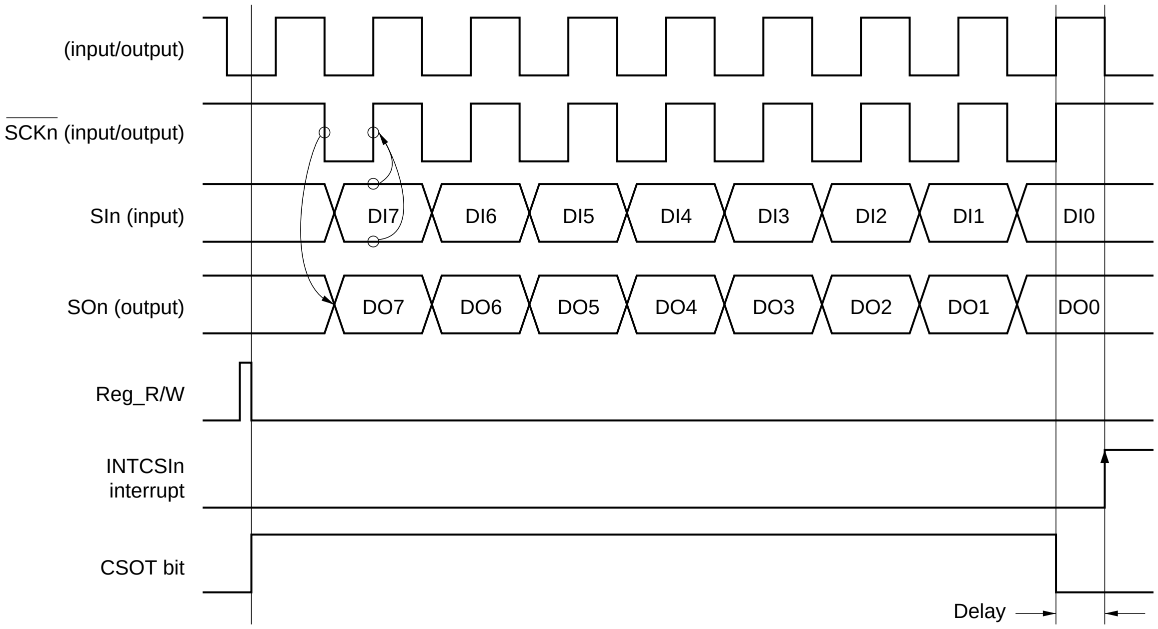
(c) Transmission/reception completion interrupt request signals (INTCSI0, INTCSI1)

INTCSI0n is set (1) upon completion of data transmission/reception.

Caution: The delay mode (CSIT bit = 1) is valid only in the master mode (bits CKS2 to CKS0 of the CSICn register are not 111B). The delay mode cannot be set when the slave mode is set (bits CKS2 to CKS0 = 111B).

Figure 12-36: Timing Chart of Interrupt Request Signal Output in Delay Mode (1/2)

(a) When CKP bit = 0, DAP bit = 0

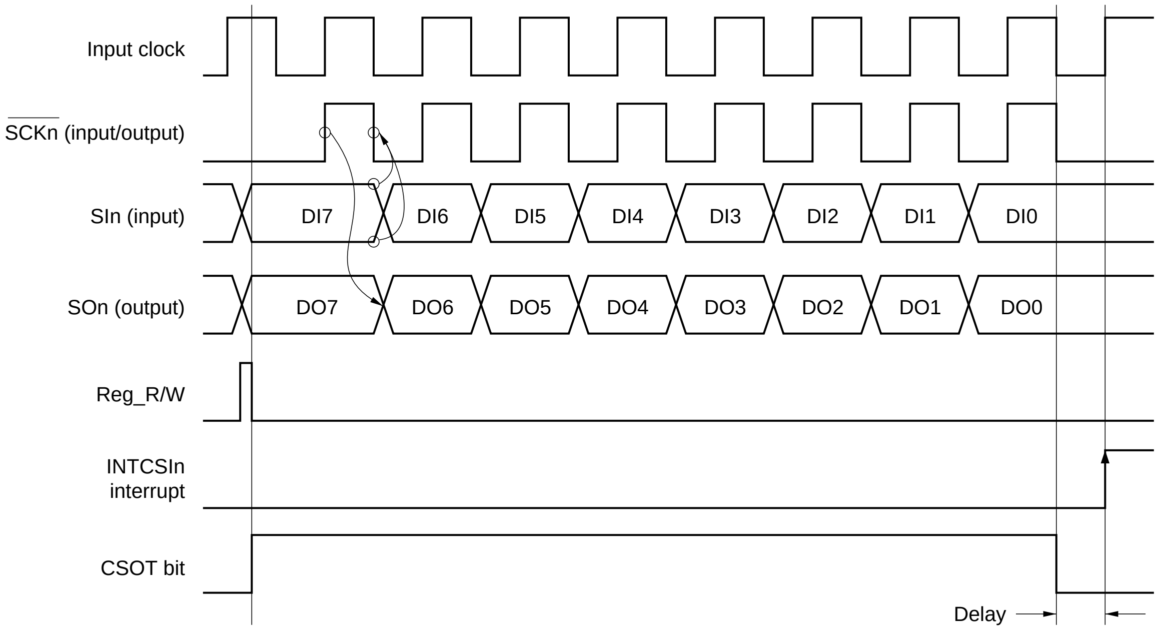


Remarks: 1. n = 0, 1

2. Reg_R/W:Internal signal. This signal indicates that receive data buffer register (SIRBn/ SIRBLn) read or transmit data buffer register (SOTBn/SOTBLn) write was performed.

Figure 12-36: Timing Chart of Interrupt Request Signal Output in Delay Mode (2/2)

(b) When CKP bit = 1, DAP bit = 1



Remarks: **1.** n = 0, 1

2. Reg_R/W:Internal signal. This signal indicates that receive data buffer register (SIRBn/SIRBLn) read or transmit data buffer register (SOTBn/SOTBLn) write was performed.

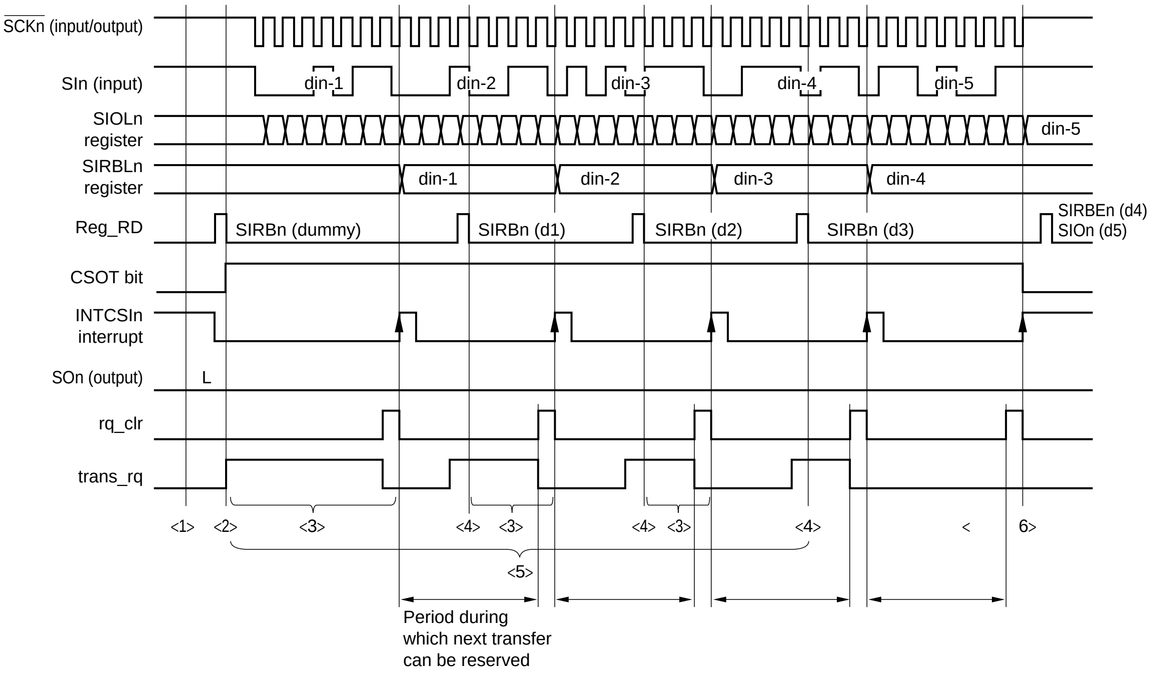
(2) Repeat transfer mode

(a) Usage (receive-only)

<1> Set the repeat transfer mode (AUTO bit of CSIMn register = 1) and the receive-only mode (TRMD bit of CSIMn register = 0).
<2> Read SIRBn register (start transfer with dummy read).
<3> Wait for transmission/reception completion interrupt request (INTCSIn).
<4> When the transmission/reception completion interrupt request (INTCSIn) has been set to (1), read the SIRBn register**Note** (reserve next transfer).
<5> Repeat steps <3> and <4> (n - 2) times (n: number of transfer data).
<6> Following output of the last transmission/reception completion interrupt request (INTCSIn), read the SIRBn register and the SIOn register**Note**.

Note: When transferring n number of data, receive data is loaded by reading the SIRBn register from the first data to the (n - 2)-th data. The (n-1)-th data is loaded by reading the SIRBEn register, and the n-th (last) data is loaded by reading the SIOn register.

Figure 12-37: Repeat Transfer (Receive-Only) Timing Chart



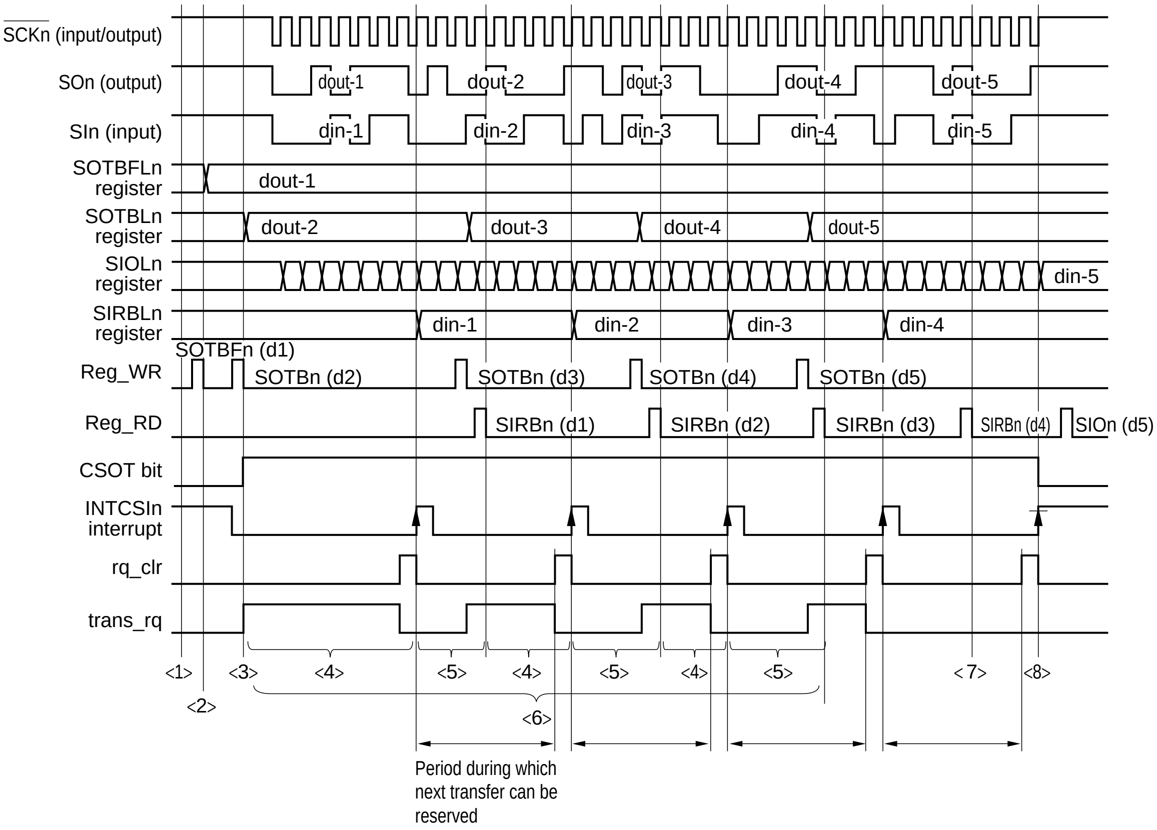
Remarks: **1.** n = 0, 1
2. Reg_RD:Internal signal. This signal indicates that the receive data buffer register (SIRBn/SIRBLn) has been read.
rq_clr: Internal signal. Transfer request clear signal.
trans_rq: Internal signal. Transfer request signal.

In the case of the repeat transfer mode, two transfer requests are set at the start of the first transfer. Following the transmission/reception completion interrupt request (INTCSIn), transfer is continued if the SIRBn register can be read within the next transfer reservation period. If the SIRBn register cannot be read, transfer ends and the SIRBn register does not receive the new value of the SIOn register.
The last data can be obtained by reading the SIOn register following completion of the transfer.

(b) Usage (transmission/reception)

<1> Set the repeat transfer mode (AUTO bit of CSIMn register = 1) and the transmission/reception mode (TRMD bit of CSIMn register = 1).
<2> Write the first data to the SOTBFn register.
<3> Write the 2nd data to the SOTBn register (start transfer).
<4> Wait for transmission/reception completion interrupt request (INTCSIn).
<5> When the transmission/reception completion interrupt request (INTCSIn) has been set to (1), write the next data to the SOTBn register (reserve next transfer), and read the SIRBn register to load the receive data.
<6> Repeat steps <4> and <5> as long as data to be sent remains.
<7> Wait for the INTCSIn interrupt. When the interrupt request signal is set to (1), read the SIRBn register to load the (n - 1)-th receive data.
<8> Following the last transmission/reception completion interrupt request (INTCSIn), read the SIOn register to load the n-th (last) receive data.

Figure 12-38: Repeat Transfer (Transmission/Reception) Timing Chart



Remarks: 1. n = 0, 1

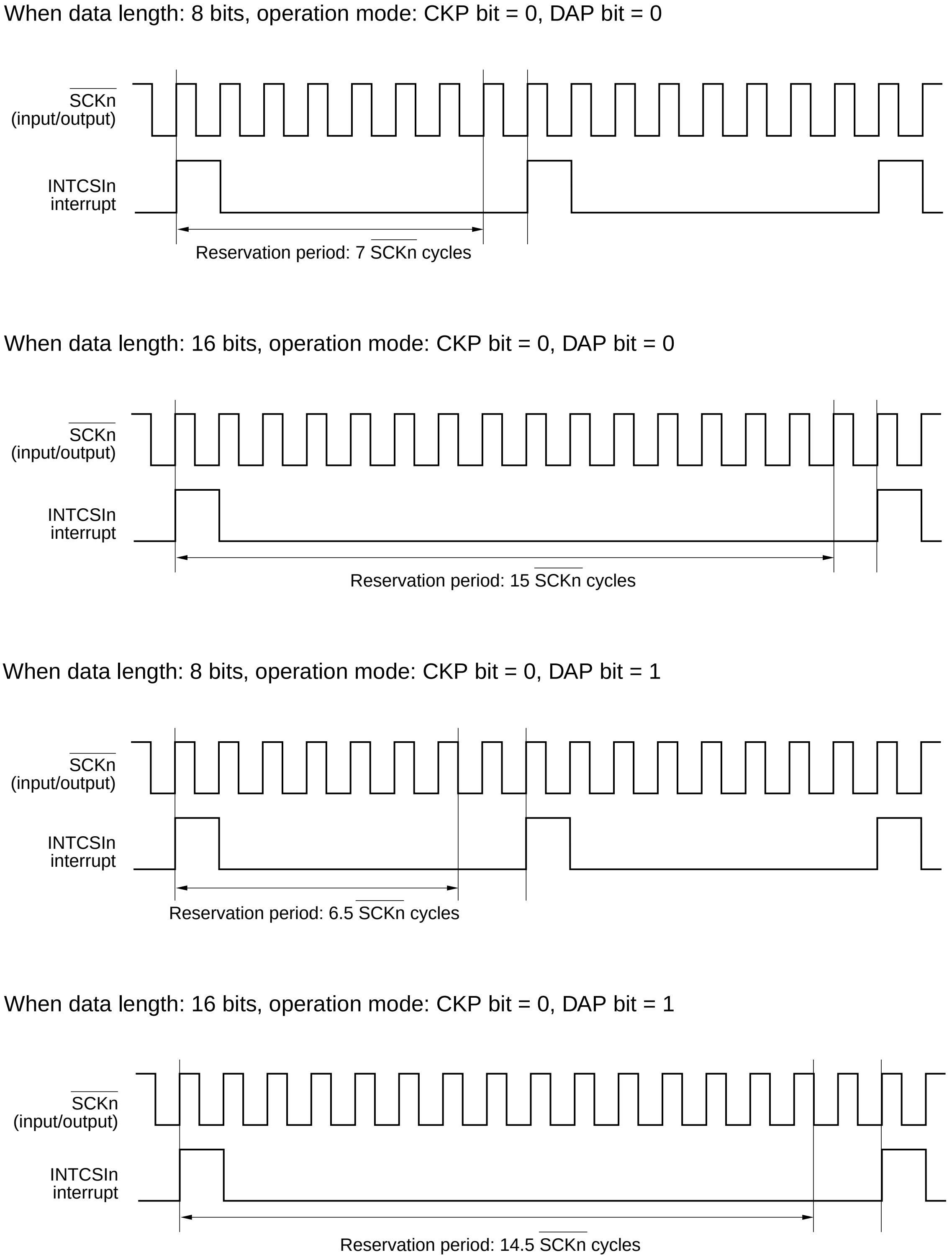
2. Reg_WR:Internal signal. This signal indicates that the transmit data buffer register (SOTBn/SOTBLn) has been written.
Reg_RD:Internal signal. This signal indicates that the receive data buffer register (SIRBn/SIRBLn) has been read.
rq_clr: Internal signal. Transfer request clear signal.
trans_rq: Internal signal. Transfer request signal.

In the case of the repeat transfer mode, two transfer requests are set at the start of the first transfer. Following the transmission/reception completion interrupt request (INTCSIn), transfer is continued if the SOTBn register can be written within the next transfer reservation period. If the SOTBn register cannot be written, transfer ends and the SIRBn register does not receive the new value of the SIOn register. The last receive data can be obtained by reading the SIOn register following completion of the transfer.

(c) Next transfer reservation period

In the repeat transfer mode, the next transfer must be prepared with the period shown in Figure 12-39.

Figure 12-39: Timing Chart of Next Transfer Reservation Period



(a) When data length: 8 bits, operation mode: CKP bit = 0, DAP bit = 0

(b) When data length: 16 bits, operation mode: CKP bit = 0, DAP bit = 0

(c) When data length: 8 bits, operation mode: CKP bit = 0, DAP bit = 1

(d) When data length: 16 bits, operation mode: CKP bit = 0, DAP bit = 1

$\overline{\text{SCKn}}$
(input/output)
INTCSIn
interrupt
Reservation period: 7 $\overline{\text{SCKn}}$ cycles
$\overline{\text{SCKn}}$
(input/output)
INTCSIn
interrupt
Reservation period: 15 $\overline{\text{SCKn}}$ cycles
$\overline{\text{SCKn}}$
(input/output)
INTCSIn
interrupt
Reservation period: 6.5 $\overline{\text{SCKn}}$ cycles
$\overline{\text{SCKn}}$
(input/output)
INTCSIn
interrupt
Reservation period: 14.5 $\overline{\text{SCKn}}$ cycles

Remark: n = 0, 1

(d) Cautions

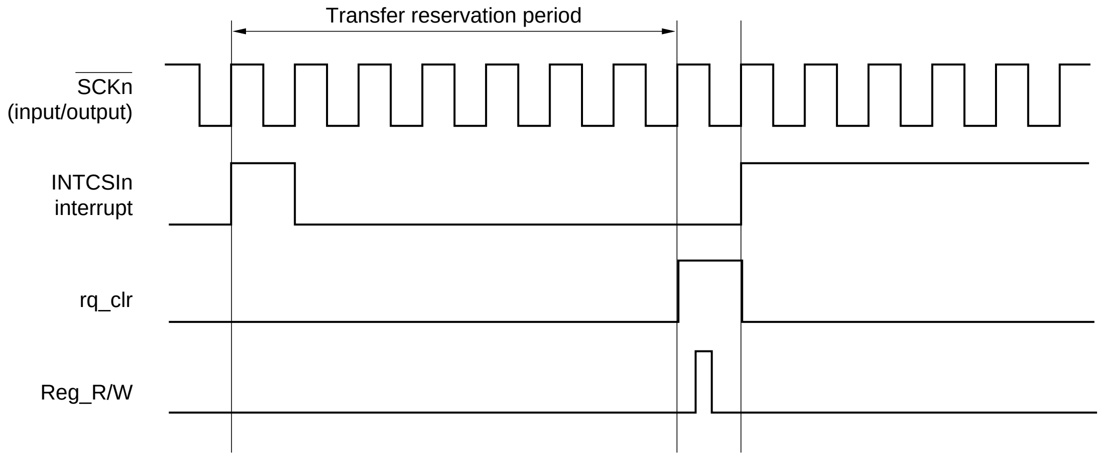
To continue repeat transfers, it is necessary to either read the SIRBn register or write to the SOTBn register during the transfer reservation period.

If access is performed to the SIRBn register or the SOTBn register when the transfer reservation period is over, the following occurs.

- In case of contention between transfer request clear and register access

Since request cancellation has higher priority, the next transfer request is ignored. Therefore, transfer is interrupted, and normal data transfer cannot be performed.

Figure 12-40: Transfer Request Clear and Register Access Contention

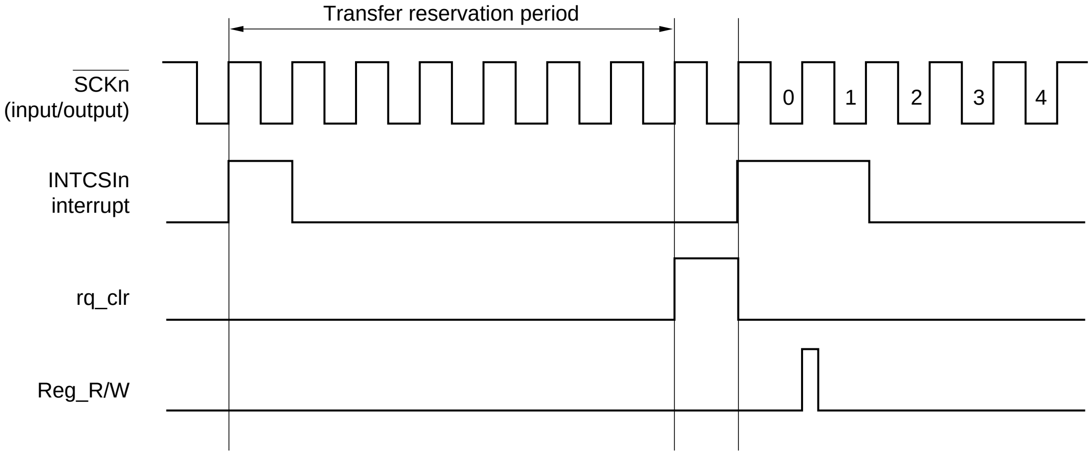


Remarks: **1.** n = 0, 1

2. rq_clr: Internal signal. Transfer request clear signal.
Reg_WR:Internal signal. This signal indicates that the transmit data buffer register (SOTBn/SOTBLn) has been written.

- In case of contention between interrupt request and register access
Since continuous transfer has stopped once, executed as a new repeat transfer.
In the slave mode, a bit phase error transfer error results (refer to Figure 12-41).
In the transmission/reception mode, the value of the SOTBFn register is retransmitted, and illegal data is sent.

Figure 12-41: Interrupt Request and Register Access Contention



Remarks: 1. n = 0, 1
2. rq_clr: Internal signal. Transfer request clear signal.
Reg_WR:Internal signal. This signal indicates that the transmit data buffer register (SOTBn/SOTBLn) has been written.

12.3.5 Output pins

(1) $\overline{\text{SCKn}}$ pin

When the CSIn operation is disabled (CSIE bit of CSIMn register = 0), the $\overline{\text{SCKn}}$ pin output status is as follows (n = 0, 1).

CKP	CKS2	CKS1	CKS0	$\overline{\text{SCKn}}$ Pin Output
0	Don’t care	Don’t care	Don’t care	Fixed to high level
1	1	1	1	Fixed to high level
	Other than above			Fixed to low level

Remarks: **1.** n = 0, 1

2. When any of bits CKP and CKS2 to CKS0 of the CSICn register is overwritten, the $\overline{\text{SCKn}}$ pin output changes.

(2) SOn pin

When the CSIn operation is disabled (CSIE bit of CSIMn register = 0), the SOn pin output status is as follows (n = 0, 1).

TRMD	DAP	AUTO	CCL	DIR	SOn Pin Output
0	Don’t care	Don’t care	Don’t care	Don’t care	Fixed at low level
1	0	Don’t care	Don’t care	Don’t care	SO latch value (low level)
	1	0	0	0	SOTB7 value
				1	SOTB0 value
			1	0	SOTB15 value
				1	SOTB0 value
		1	0	0	SOTBF7 value
				1	SOTBF0 value
			1	0	SOTBF15 value
				1	SOTBF0 value

Remarks: **1.** When any of bits TRMD, CCL, DIR, AUTO, and CSICn of the CSIMn register or DAP bit of the CSICn register is overwritten, the SOn pin output changes.

2. SOTBm: Bit m of SOTBn register (m = 0, 7, 15)

3. SOTBFm: Bit m of SOTBFn register (m = 0, 7, 15)

4. n = 0, 1

12.3.6 Dedicated baud rate generators 0, 1 (BRG0, BRG1)

(1) Selecting the baud rate generator

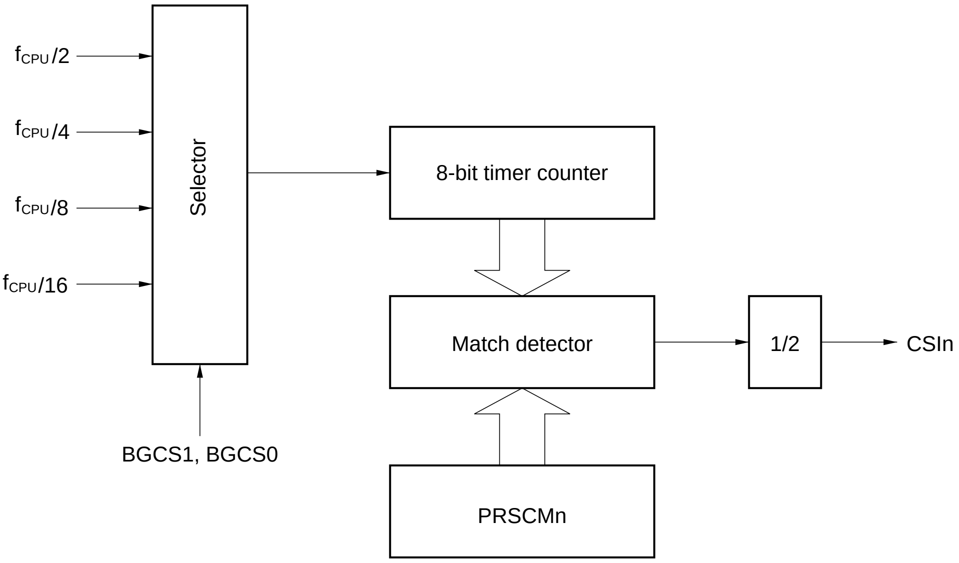
The CSI0 and CSI1 serial clocks can be selected between dedicated baud rate generator output or internal system clock (f_{CPU}).

The serial clock source is specified by bits CKS2 to CKS0 of registers CSIC0 and CSIC1 (refer to **12.3.3 (2)Clocked serial interface clock selection registers 0, 1 (CSIC0, CSIC1)**).

If the dedicated baud rate generator output is specified, BRG0 or BRG1 respectively is selected as the clock source.

Since the same serial clock can be shared for transmission and reception, baud rate is the same for the transmission/reception.

Figure 12-42: Baud Rate Generators 0, 1 (BRG0, BRG1) Block Diagram



Remarks: **1.** f_{CPU}: Internal system clock

2. n = 0, 1

(2) Configuration

BRGn is configured of an 8-bit timer counter that generates the baud rate signal, a prescaler mode register n (PRSMn) that controls baud rate signal generation, a prescaler compare register n (PRSCMn) that sets the value of the 8-bit timer counter, and a prescaler (n = 0, 1).

(a) Input clock

The internal system clock (f_{CPU}) is input to BRGn.

(b) Prescaler mode registers 0, 1 (PRSM0, PRSM1)

The PRSMn register controls the generation of the CSI0 and CSI1 baud rate signals respectively. This register can be read/written in 8-bit or 1-bit units (n = 0, 1).

Figure 12-43: Prescaler Mode Registers 0, 1 (PRSM0, PRSM1)

	7	6	5	4	3	2	1	0	Address	Initial value
PRSM0	0	0	0	CE	0	0	BGCS1	BGCS0	FFFFF920H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
PRSM1	0	0	0	CE	0	0	BGCS1	BGCS0	FFFFF930H	00H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>4</td><td>CE</td><td>Enables baud rate counter operation.
0: Stop baud rate counter operation and fix baud rate output signal to 0.
1: Enable baud rate counter operation and start baud rate output operation.</td></tr>
<tr><td>1, 0</td><td>BGCS1,
BGCS0</td><td>Selects count clock for baud rate counter.
<table>
<tr><th>BGCS1</th><th>BGCS0</th><th>Count Clock Selection</th></tr>
<tr><td>0</td><td>0</td><td>$f_{CPU}/2$</td></tr>
<tr><td>0</td><td>1</td><td>$f_{CPU}/4$</td></tr>
<tr><td>1</td><td>0</td><td>$f_{CPU}/8$</td></tr>
<tr><td>1</td><td>1</td><td>$f_{CPU}/16$</td></tr>
</table>
<b>Remark:</b> f_{CPU}: Internal system clock.</td></tr>
</table>

Cautions: 1. Do not change the value of the BGCS1, BGCS0 bits during transmission/reception operation.
2. Set the PRSMn register prior to setting the CE bit to 1.

(c) Prescaler compare registers 0, 1 (PRSCM0, PRSCM1)

PRSCMn is an 8-bit compare register that sets the value of the 8-bit timer counter.
This register can be read/written in 8-bit or 1-bit units (n = 0, 1).

Figure 12-44: Prescaler Compare Registers 0, 1 (PRSCM0, PRSCM1)

	7	6	5	4	3	2	1	0	Address	Initial value
PRSCM0	PRSCM7	PRSCM6	PRSCM5	PRSCM4	PRSCM3	PRSCM2	PRSCM1	PRSCM0	FFFFF922H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
PRSCM1	PRSCM7	PRSCM6	PRSCM5	PRSCM4	PRSCM3	PRSCM2	PRSCM1	PRSCM0	FFFF932H	00H

Bit Position	Bit Name	Function
7 to 0	PRSCM7 to PRSCM0	Compare value of the 8-bit timer counter.

Cautions: 1. The internal timer counter is cleared by writing to the PRSMn register. Therefore, do not write to the PRSCMn register during transmission.

2. Set the PRSCMn register prior to setting the CE bit of the PRSMn register to 1. If the contents of the PRSCMn register are overwritten when the value of the CE bit is 1, the cycle of the baud rate signal is not guaranteed.

(d) Baud rate signal cycle

The baud rate signal cycle is calculated as follows.

- **When setting value of PRSCMn register is 00H**
(Cycle of signal selected with bits BGCS1, BGCS0 of PRSMn register) / 256 × 2

- **In cases other than above**
(Cycle of signal selected with bits BGCS1, BGCS2 of PRSMn register) / (setting value of PRSCMn register) × 2

(e) Baud rate setting example

Table 12-6: Baud Rate Generator Setting Data

<1> When f_{CPU} = 16 MHz

BGCS1	BGCS0	PRSCM Register Value	Clock (Hz)
0	0	1	4000000
0	0	2	2000000
0	0	4	1000000
0	0	8	500000
0	0	16	250000
0	0	40	100000
0	0	80	50000
0	0	160	25000
0	1	200	10000
1	0	200	5000

<2> When f_{CPU} = 20 MHz

BGCS1	BGCS0	PRSCM Register Value	Clock (Hz)
0	0	2	2500000
0	0	5	1000000
0	0	10	500000
0	0	20	250000
0	0	50	100000
0	0	100	50000
0	0	200	25000
0	1	250	10000
1	0	250	5000

Caution: Set the transfer clock so that it does not fall below the minimum value of 200 ns of the $\overline{\text{SCKn}}$ cycle (t_{CYSK1}) prescribed in the electrical specifications.

Chapter 13 FCAN Interface Function

13.1 Features

- Active support of extended format (ISO 11898, former CAN specification version 2.0B active), supporting transmission and reception of standard and extended frame format messages

- 3 CAN modules

- CAN bus speed up to 1 Mbit per second

- Direct message storage for minimum CPU burden

- Configurable number of message buffers per CAN module

- 64 message buffers in total

- Mask option for receive messages (BasicCAN channels)

- 4 masks per CAN module (each mask can be assigned to each message)

- Buffered reception (FIFO)

- Message buffers can be redefined in normal operation mode

- FCAN interface and CPU share common RAM area

- Interrupt on receive, transmit and error condition

- Time stamp and global time system function

- Two power-save modes
  - SLEEP mode: wake-up at CAN bus activity
  - STOP mode: no wake-up at CAN bus activity

- Diagnostic features
  - Readable error counters
  - CAN bus status information register
  - Receive-only mode (e.g. used for automatic bit rate detection)
  - Bus error cause information

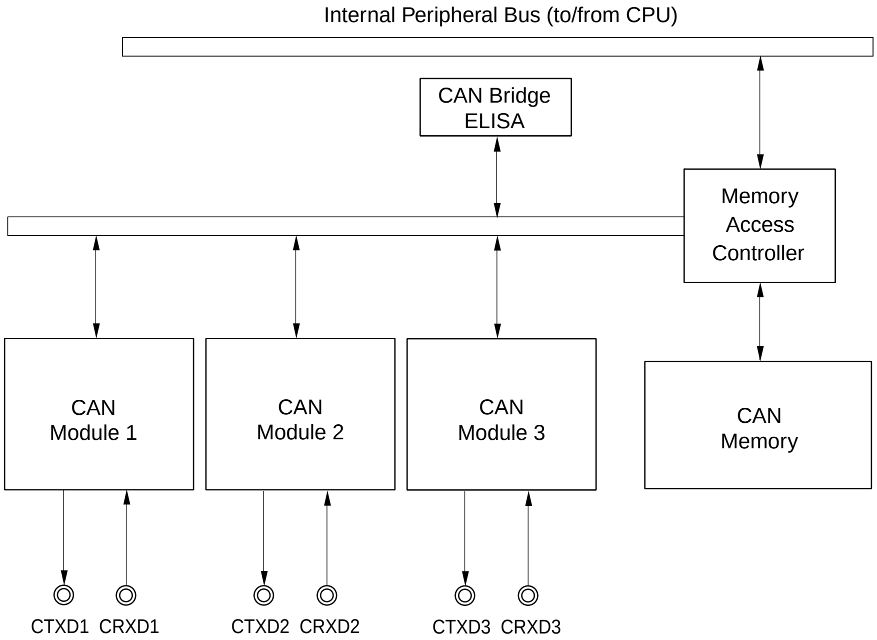
- Event processing by CAN bridge ELISA

13.2 Outline of the FCAN System

13.2.1 General

The FCAN (Full-CAN) system of the V850E/CA1 (Atomic) supports 3 independent CAN modules (CAN module 1, CAN module 2, CAN module 3), which provide each an interface to a Controller Area Network (CAN).
The CAN modules are conform to ISO 11898, former CAN specification version 2.0B active.
An external bus transceiver has to be used to connect a CAN module to a CAN bus. That external bus transceiver converts the transmit data line and receive data line signals to the necessary electrical signal characteristic on the CAN bus itself.
All protocol activities in a CAN module are handled by hardware (transfer layer).
The CAN modules themselves provide no memory for the necessary data buffers, rather all CAN modules have access to the common CAN memory area via a memory access controller (MAC).
The MAC allows integration of machines other than CAN modules (e.g. CAN bridge). The CAN bridge accomplishes data processing among the CAN modules without involving the CPU.
The CPU also accesses to the common CAN memory via the MAC. The MAC offers data scan capability beside controlling the arbitration of CAN modules, CAN bridge or CPU accesses to the CAN memory.
By means of that scan capability inner priority inversions at message transmissions are automatically avoided and received messages are sorted into the corresponding receive message buffers according to an inner storage priority rule.

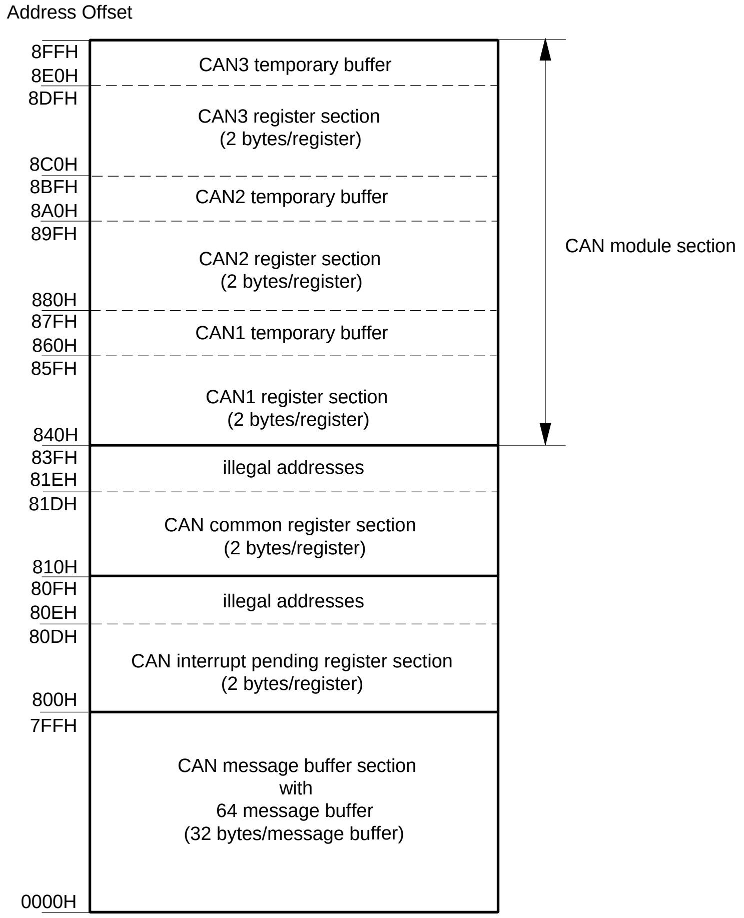
Figure 13-1: Functional Blocks of the FCAN Interface



13.2.2 CAN memory and register layout

All buffers and registers of the FCAN system are arranged within a memory layout of 3 KB.

Figure 13-2: Memory Area of the FCAN System



Remarks: 1. Effective address = PP_BASE + address offset

2. The memory area is located in the 16 KB programmable peripheral I/O area of the V850E/CA1 (Atomic). The base address (PP_BASE) of the programmable peripheral I/O area is set by the BPC register (refer to **3.4.9 Programmable peripheral I/O registers**).
3. The memory area of the FCAN system is divided into certain functional sections. The start and end addresses of those sections are given as an address offset value.

Caution: Before accessing any register or buffer of the FCAN system the base address PP_BASE must be fixed by the BPC register.

The sections within the FCAN memory layout contain areas, which are defined as illegal addresses or CANx temporary buffer (x = 1 to 3).

Remarks: 1. Areas defined as illegal addresses contain neither FCAN registers nor FCAN buffers. Those area must not be read nor written by user program.

2. CANx temporary buffers can be accessed by CPU (write and read accesses) when the GOM bit of the CGST register is cleared (0) (means FCAN system inactive).
Whenever the FCAN system is in global operating mode (GOM = 1) the temporary buffer must not be written by the CPU. The global interrupt GINT2 signals accidental write accesses by CPU while the FCAN system is active.

(1) CAN message buffer section

The message buffer section consists of 64 message buffers. Each message buffer allocates 32 bytes.

The message buffers are not statically distributed and linked to the CAN modules, rather the user must determine the link of a message buffer to a CAN module by software. As a consequence the message buffers can be allocated to a CAN module according to the need of the particular CAN network.

Table 13-1: Configuration of the CAN Message Buffer Section

Address Offset [Note]	Name
000H to 01FH	Message buffer 0
020H to 03FH	Message buffer 1
040H to 05FH	Message buffer 2
. . .	
7C0H to 7DFH	Message buffer 62
7E0H to 7FFH	Message buffer 63

Note: The address of a message buffer entry is calculated according to the following formula: effective address = PP_BASE + address offset

Each message buffer has the same register layout (refer to Table 13-2).

Table 13-2: CAN Message Buffer Registers Layout

Address Offset **Note 1, 2**	Symbol[Note 1]	Name	Ref. Page	Access Type			
				R/W	1 bit	8 bit	16 bits
(m x 20H) + 000H	M_EVTm0	Message event register 0	424	R/W		×	
(m x 20H) + 001H	M_EVTm1	Message event register 1		R/W		×	
(m x 20H) + 002H	M_EVTm2	Message event register 2 (Unused)	–	–			
(m x 20H) + 003H	M_EVTm3	Message event register 3	424	R/W		×	
(m x 20H) + 004H	M_DLCm	Message data length code register	420	R/W		×	
(m x 20H) + 005H	M_CTRLm	Message control register	421	R/W		×	
(m x 20H) + 006H	M_TIMEm	Message time stamp register	423	R/W			×
(m x 20H) + 008H	M_DATAm0	Message data byte 0	418	R/W		×	
(m x 20H) + 009H	M_DATAm1	Message data byte 1		R/W		×	
(m x 20H) + 00AH	M_DATAm2	Message data byte 2		R/W		×	
(m x 20H) + 00BH	M_DATAm3	Message data byte 3		R/W		×	
(m x 20H) + 00CH	M_DATAm4	Message data byte 4		R/W		×	
(m x 20H) + 00DH	M_DATAm5	Message data byte 5		R/W		×	
(m x 20H) + 00EH	M_DATAm6	Message data byte 6		R/W		×	
(m x 20H) + 00FH	M_DATAm7	Message data byte 7		R/W		×	
(m x 20H) + 010H	M_IDLm	Message identifier register (lower half-word)	413	R/W			×
(m x 20H) + 012H	M_IDHm	Message identifier register (upper half-word)		R/W			×
(m x 20H) + 014H	M_CONFm	Message configuration register	414	R/W		×	
(m x 20H) + 015H	M_STATm	Message status register	415	R		×	
(m x 20H) + 016H	SC_STATm	Message set/clear status register	417	W			×
(m x 20H) + 018H to (m x 20H) + 01FH	–	Reserved	–	–			

Notes: **1.** m = number of CAN message buffer (m = 00 to 63)

2. The address of a message buffer entry is calculated according to the following formula: effective address = PP_BASE + address offset

(2) CAN Interrupt Pending Registers Section

The layout of the interrupt pending register section is shown in Table 13-3.

Table 13-3: Relative Addresses of CAN Interrupt Pending Registers

Address Offset[Note]	Symbol	Name	Ref. Page	Access Type				Comment
				R/W	1 bit	8 bits	16 bits	
800H	CCINTP	CAN interrupt pending register	408	R		×	×	
802H	CGINTP	CAN global interrupt pending register	409	R		×	×	
				W			×	bit-set function only
804H	C1INTP	CAN1 interrupt pending register	411	R		×	×	
				W		×	×	bit-clear function only
806H	C2INTP	CAN2 interrupt pending register	411	R		×	×	
				W		×	×	bit-clear function only
808H	C3INTP	CAN3 interrupt pending register	411	R		×	×	
				W		×	×	bit-clear function only

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = PP_BASE + address offset

(3) CAN Common Registers Section

The layout of the common register section is shown in Table 13-4.

Table 13-4: Relative Addresses of CAN Common Registers

Address Offset[Note]	Symbol	Name	Ref. Page	Access Type				Comment
				R/W	1 bit	8 bits	16 bits	
80CH	CSTOP	CAN stop register	396	R/W		×	×	
810H	CGST	CAN global status register	399	R	×	×	×	
				W			×	bit set/clear function
812H	CGIE	CAN global interrupt enable register	401	R	×	×	×	
				W			×	bit set/clear function
814H	CGCS	CAN main clock select register	397	R	×	×	×	
				W	×	×	×	only if GOM bit = 0
816H	CGTEN	CAN timer event enable register	403	R/W	×	×	×	
818H	CGTSC	CAN global time system counter	404	R	×	×	×	
				W			×	complete clear only
81AH	CGMSS	CAN message search start register	405	W			×	write only
	CGMSR	CAN message search result register	406	R	×	×	×	read only
81CH	CTBR	CAN test bus register	407	R/W		×	×	

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = PP_BASE + address offset

(4) CAN Module Registers Section

The appropriate register section of each CAN module is shown in Table 13-5 for CAN module 1, in Table 13-6 for CAN module 2 and in Table 13-7 for CAN module 3.

Table 13-5: Relative Addresses of CAN Module 1 Registers

Address Offset[Note]	Symbol	Name	Ref. Page	Access Type				Comment
				R/W	1 bit	8 bits	16 bits	
840H	C1MASKL0	CAN1 mask 0 register L	425	R/W		×	×	lower half-word
842H	C1MASKH0	CAN1 mask 0 register H		R/W		×	×	upper half-word
844H	C1MASKL1	CAN1 mask 1 register L		R/W		×	×	lower half-word
846H	C1MASKH1	CAN1 mask 1 register H		R/W		×	×	upper half-word
848H	C1MASKL2	CAN1 mask 2 register L		R/W		×	×	lower half-word
84AH	C1MASKH2	CAN1 mask 2 register H		R/W		×	×	upper half-word
84CH	C1MASKL3	CAN1 mask 3 register L		R/W		×	×	lower half-word
84EH	C1MASKH3	CAN1 mask 3 register H		R/W		×	×	upper half-word
850H	C1CTRL	CAN1 control register	427	R		×	×	
				W			×	bit set/clear function
852H	C1DEF	CAN1 definition register	431	R		×	×	
				W			×	bit set/clear function
854H	C1LAST	CAN1 information register	434	R		×	×	read only
856H	C1ERC	CAN1 error counter register	435	R		×	×	read only
858H	C1IE	CAN1 interrupt enable register	436	R		×	×	
				W			×	bit set/clear function
85AH	C1BA	CAN1 bus activity register	438	R		×	×	
				W			×	bit set/clear function
85CH	C1BRP	CAN1 bit rate prescaler register	440	R		×	×	
				W		×	×	in initialisation state only (ISTAT = 1)
	C1DINF	CAN1 bus diagnostic information register	445	R		×	×	in diagnostic mode only
85EH	C1SYNC	CAN1 synchronization control register	442	R		×	×	
				W		×	×	

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = PP_BASE + address offset

Table 13-6: Relative Addresses of CAN Module 2 Registers

Address Offset[Note]	Symbol	Name	Ref. Page	Access Type				Comment
				R/W	1 bit	8 bits	16 bits	
880H	C2MASKL0	CAN2 mask 0 register L	425	R/W		×	×	lower half-word
882H	C2MASKH0	CAN2 mask 0 register H		R/W		×	×	upper half-word
884H	C2MASKL1	CAN2 mask 1 register L		R/W		×	×	lower half-word
886H	C2MASKH1	CAN2 mask 1 register H		R/W		×	×	upper half-word
888H	C2MASKL2	CAN2 mask 2 register L		R/W		×	×	lower half-word
88AH	C2MASKH2	CAN2 mask 2 register H		R/W		×	×	upper half-word
88CH	C2MASKL3	CAN2 mask 3 register L		R/W		×	×	lower half-word
88EH	C2MASKH3	CAN2 mask 3 register H		R/W		×	×	upper half-word
890H	C2CTRL	CAN2 control register	427	R		×	×	
				W			×	bit set/clear function
892H	C2DEF	CAN2 definition register	431	R		×	×	
				W			×	bit set/clear function
894H	C2LAST	CAN2 information register	434	R		×	×	read only
896H	C2ERC	CAN2 error counter register	435	R		×	×	read only
898H	C2IE	CAN2 interrupt enable register	436	R		×	×	
				W			×	bit set/clear function
89AH	C2BA	CAN2 bus activity register	438	R		×	×	
				W			×	bit-set/clear function
89CH	C2BRP	CAN2 bit rate prescaler register	440	R		×	×	
				W		×	×	in initialisation state only (ISTAT bit = 1)
	C2DINF	CAN2 bus diagnostic information register	445	R		×	×	in diagnostic mode only
89EH	C2SYNC	CAN2 synchronization control register	442	R		×	×	
				W		×	×	

Note: The address of a CAN module 2 register is calculated according to the following formula: effective address = PP_BASE + address offset

Table 13-7: Relative Addresses of CAN Module 3 Registers

Address Offset[Note]	Symbol	Name	Ref. Page	Access Type				Comment
				R/W	1 bit	8 bits	16 bits	
8C0H	C3MASKL0	CAN3 mask 0 register L	425	R/W		×	×	lower half-word
8C2H	C3MASKH0	CAN3 mask 0 register H		R/W		×	×	upper half-word
8C4H	C3MASKL1	CAN3 mask 1 register L		R/W		×	×	lower half-word
8C6H	C3MASKH1	CAN3 mask 1 register H		R/W		×	×	upper half-word
8C8H	C3MASKL2	CAN3 mask 2 register L		R/W		×	×	lower half-word
8CAH	C3MASKH2	CAN3 mask 2 register H		R/W		×	×	upper half-word
8CCH	C3MASKL3	CAN3 mask 3 register L		R/W		×	×	lower half-word
8CEH	C3MASKH3	CAN3 mask 3 register H		R/W		×	×	upper half-word
8D0H	C3CTRL	CAN3 control register	427	R		×	×	
				W			×	bit set/clear function
8D2H	C3DEF	CAN3 definition register	431	R		×	×	
				W			×	bit set/clear function
8D4H	C3LAST	CAN3 information register	434	R		×	×	read only
8D6H	C3ERC	CAN3 error counter register	435	R		×	×	read only
8D8H	C3IE	CAN3 interrupt enable register	436	R		×	×	
				W			×	bit set/clear function
8DAH	C3BA	CAN3 bus activity register	438	R		×	×	
				W			×	bit set/clear function
8DCH	C3BRP	CAN3 bit rate prescaler register	440	R		×	×	
				W		×	×	in initialisation state only (ISTAT bit = 1)
	C3DINF	CAN3 bus diagnostic information register	445	R		×	×	in diagnostic mode only
8DEH	C3SYNC	CAN3 synchronization control register	442	R		×	×	
				W		×	×	

Note: The address of a CAN module 3 register is calculated according to the following formula:
effective address = PP_BASE + address offset

(5) CAN Bridge ELISA Register Section

The layout of the ELISA register section is shown in Table 13-8.

Table 13-8: Relative Addresses of CAN Bridge ELISA Registers

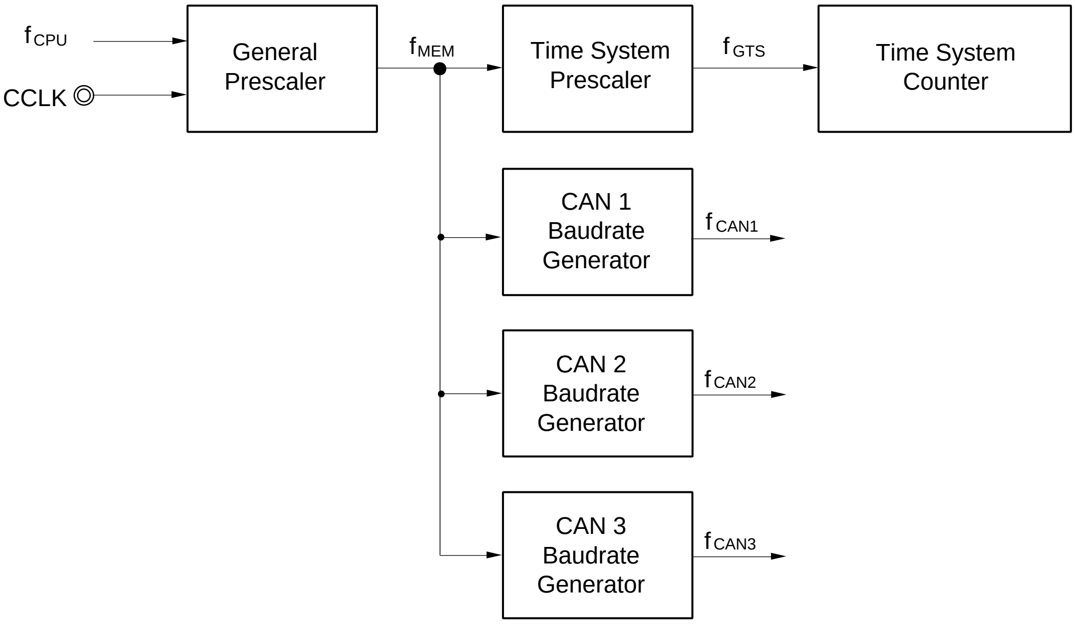
Address Offset[Note]	Symbol	Name	Ref. Page	Access Type				Comment
				R/W	1 bit	8 bits	16 bits	
A10H	TEP0	Timer event pointer register 0	480	R/W		×		
A11H	TEP1	Timer event pointer register 1		R/W		×		
A12H	–	Reserved						
A13H	TEP3	Timer event pointer register 3	480	R/W		×		
A14H	SEPCC	Script event pointer and command counter register	481	R/W		×	×	
A16H	EEPS	ELISA event processing status register	482	R/W		×	×	
A18H	ELSR	ELISA status register	483	R		×	×	
				W			×	bit set/clear function
A1AH	ELC	ELISA last processed command register	485	R		×	×	read only
A1CH	ETBL	ELISA temporary buffer (lower half-word)	486	R/W		×	×	
A1EH	ETBH	ELISA temporary buffer (upper half-word)		R/W		×	×	

Note: The address of CAN bridge ELISA register is calculated according to the following formula: effective address = PP_BASE + address offset

13.2.3 Clock structure

All functional blocks within the FCAN system are supplied by a unique clock (f_{MEM}) derived from the internal system clock (f_{CPU}) or an external clock (f_{EXT}).

Figure 13-3: Clock Structure of the FCAN System



The general prescaler, controlled by the CGCS register, selects and scales either the internal system clock (f_{CPU}) or an external clock (f_{EXT}), which is supplied via the CCLK input pin.

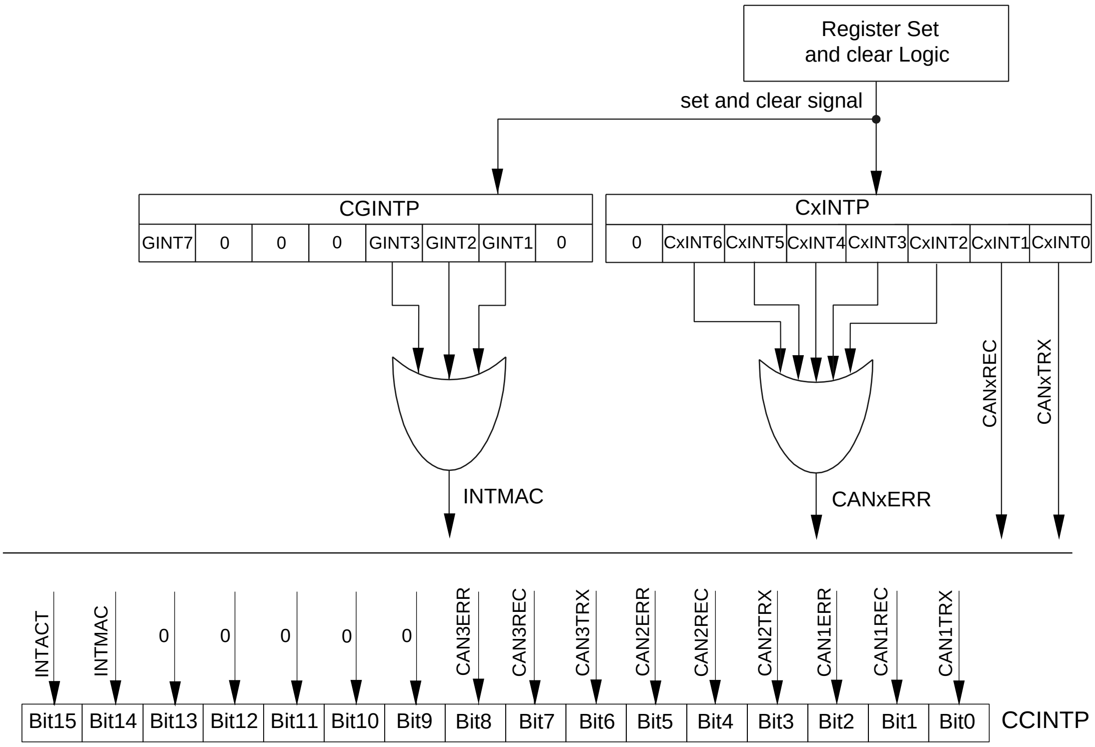
A functional block for a global time system is integrated in the FCAN system. That functional block is supplied by the global time system clock (f_{GTS}), which is derived from f_{MEM}. The time system prescaler scales f_{GTS} and is also controlled by the CGCS register.
The time base of the global time system is realised by the 16-bit free-running counter, the CAN global time system counter (CGTSC). Time stamp information is captured from the CGTSC counter. (For details refer to chapter **13.2.5 Time stamp**).

The global time system clock f_{GTS} is also the base for event generation used of the CAN Bridge ELISA. For details of CAN Bridge ELISA related event generation refer to chapter 13.5.

13.2.4 Interrupt handling

The very high number of interrupt events generated by the FCAN system does not allow to assign an independent interrupt vector of the V850E/CA1 (Atomic) to each event. Therefore, the interrupt request signals are bundled into groups and the grouped interrupt request signal is then assigned to an independent interrupt vector.
The concept of interrupt request signal bundling leads to the fact that all interrupt request signals of the FCAN system are designed as interrupt pending signals. Interrupt pending signals are not automatically treated by an interrupt service routine like interrupt request signals with an unambiguous interrupt vector. Rather, on occurrence of the interrupt event the interrupt signal is generated and latched.
In the interrupt service routine the software must analyse, which particular interrupt event caused the interrupt request by scanning the interrupt pending flags of a bundled interrupt signal group. After the particular interrupt has been identified, the corresponding interrupt pending flag must be reset by software at least before leaving the interrupt service routine.

Figure 13-4: FCAN Interrupt Bundling of V850E/CA1 (Atomic)



Remark: x = 1 to 3

The interrupt pending registers of the FCAN system are:

- CGINTP: Global interrupt pending register
- C1INTP: CAN module 1 interrupt pending register
- C2INTP: CAN module 2 interrupt pending register
- C3INTP: CAN module 3 interrupt pending register

Additionally the entire interrupt pending flags are summarized in one register, the CAN interrupt pending register (CCINTP). However, the CCINTP register is a read-only register, and cannot be used for clearing the interrupt pending flags.

For details on the interrupt pending registers refer to chapter 13.3.3.

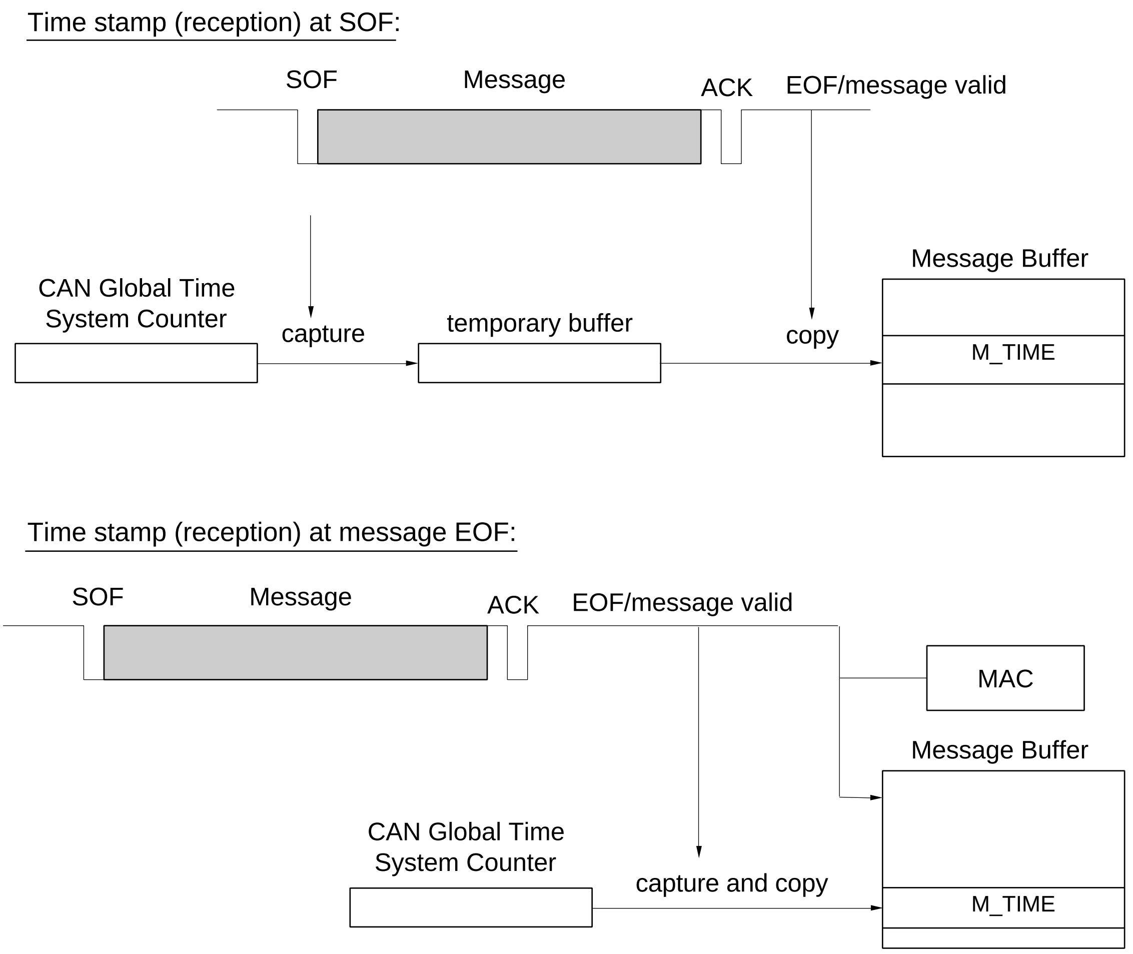
13.2.5 Time stamp

The FCAN system offers a time stamp capture capability at message reception and transmission. The time stamp capture function is used to realize a synchronized, global clock in a CAN network, also called global time system. However, the development and functionality of such a global clock system has to be implemented by the user.

For time stamp capturing at message reception two trigger events are selectable (see Figure 13-5). The counter value of the CAN global time system counter (CGTSC) is either captured upon the start-of-frame signal (SOF) of the receive message or it is captured at the time the message is detected as valid, i.e. if no error was detected until the last but one bit of the end-of-frame (EOF) was received. The selection of the two trigger options is controlled by the TMR bit in the CxCTRL register (x = 1 to 3). The capture value itself is stored in the M_TIMEm register (m = 00 to 63) of the message buffer, for which the received message has been accepted.

Remark: The value of M_TIMEm register is undefined when an error occurs while receiving the message.

Figure 13-5: Time Stamp Capturing at Message Reception



For the time stamp capturing at message transmission the SOF signal of the transmit message is used as the event trigger (see Figure 13-6).
The captured value from the CGTSC counter is written into particular data bytes of the transmit message's data field. Table 13-9 shows the scheme about which data bytes of the data field are replaced with the time stamp capture value according to the setting of the M_DLCm register (m = 00 to 63).

Figure 13-6: Time Stamp Capturing at Message Transmission

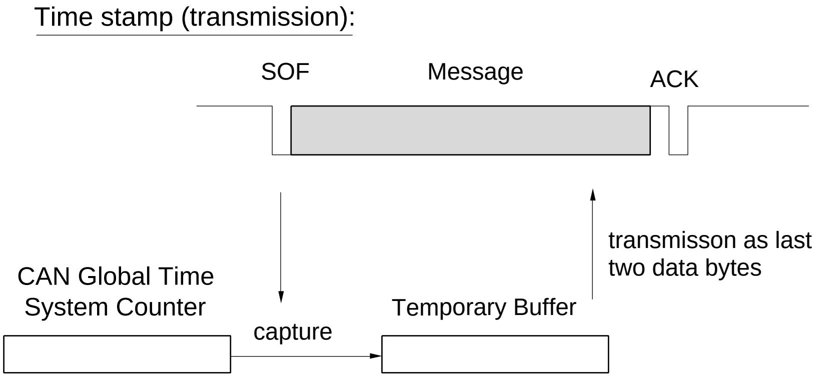

Time stamp (transmission):
SOF
Message
ACK
transmisson as last two data bytes
CAN Global Time System Counter
capture
Temporary Buffer

Table 13-9: Transmitted Data On the CAN Bus (ATS = 1)

M_DLCm	Bus Data 1	Bus Data 2	Bus Data 3	Bus Data 4	Bus Data 5	Bus Data 6	Bus Data 7	Bus Data 8
1	M_DATAm 0	–	–	–	–	–	–	–
2	lower 8-bit of CGTSC[Note]	upper 8-bit of CGTSC-[Note]	–	–	–	–	–	–
3	M_DATAm 0	lower 8-bit of CGTSC[Note]	upper 8-bit of CGTSC-[Note]	–	–	–	–	–
4	M_DATAm 0	M_DATAm 1	lower 8-bit of CGTSC[Note]	upper 8-bit of CGTSC-[Note]	–	–	–	–
5	M_DATAm 0	M_DATAm 1	M_DATAm 2	lower 8-bit of CGTSC[Note]	upper 8-bit of CGTSC-[Note]	–	–	–
6	M_DATAm 0	M_DATAm 1	M_DATAm 2	M_DATAm 3	lower 8-bit of CGTSC[Note]	upper 8-bit of CGTSC-[Note]	–	–
7	M_DATAm 0	M_DATAm 1	M_DATAm 2	M_DATAm 3	M_DATAm 4	lower 8-bit of CGTSC[Note]	upper 8-bit of CGTSC-[Note]	–
8	M_DATAm 0	M_DATAm 1	M_DATAm 2	M_DATAm 3	M_DATAm 4	M_DATAm 5	lower 8-bit of CGTSC[Note]	upper 8-bit of CGTSC-[Note]

Note: CGTSC value captured at SOF.

Remark: m = 00 to 63

13.2.6 Message handling

In the FCAN system the assignment of message buffers to the CAN modules is not defined by hardware. Each message buffer in the message buffer section can be assigned to any CAN module by software. The message buffers have individual configuration registers to assign the CAN module and to specify the message buffer type.
Basically, a message buffer can be selected as a transmit message buffer or as a receive message buffer. For receive message buffers there are further differentiations according to the mask links.

(1) Message transmission

According to the CAN protocol the highest prior message must always gain the CAN bus access against lower prior messages sent by other nodes at the same time (due to arbitration mechanism of CAN protocol) and against messages waiting to be transmitted in the same node (i.e. inner priority inversion).
The FCAN system scans the message buffer section at the beginning of each message transmit to analyse that no other message with a higher priority is waiting to be transmitted on the same CAN bus. The FCAN system avoids inner priority inversion automatically.

Example:

5 transmit messages are waiting to be sent at the same time in the example shown in Table 13-10. Although the priority of the transmit messages are not sorted according any scheme, the sequence of transmits on the CAN bus is:

<1> message buffer number 15 (ID = 023H)
<2> message buffer number 1 (ID = 120H)
<3> message buffer number 22 (ID = 123H)
<4> message buffer number 14 (ID = 223H)
<5> message buffer number 2 (ID = 229H)

Table 13-10: Example for Automatic Transmission Priority Detection

Message Buffer Address Offset[Note1]	Message Buffer Number	Message Buffer Link	Message Buffer Type[Note2]	Waiting for Transmission	Identifier
7E0H	63				
. . .	. . .				
300H	24				
2E0H	23				
2C0H	22	CAN 1	TRX	✓	123H
2A0H	21				
280H	20				
260H	19				
240H	18				
220H	17				
200H	16				
1E0H	15	CAN 1	TRX	✓	023H
1C0H	14	CAN 1	TRX	✓	223H
1A0H	13				
180H	12				
160H	11				
140H	10				
120H	9				
100H	8				
0E0H	7				
0C0H	6				
0A0H	5				
080H	4				
060H	3				
040H	2	CAN 1	TRX	✓	229H
020H	1	CAN 1	TRX	✓	120H
000H	0				

Notes: **1.** The address of a message buffer entry is calculated according to the following formula:
effective address = PP_BASE + address offset
2. TRX = transmit message

Caution: **In case more than 5 transmit messages are linked to a CAN module, the user must allocate the 5 higher prior transmit messages to message buffers with a lower address. There is no sorting needed among the 5 higher prior message buffer.**

Table 13-11: Example for Transmit Buffer Allocation When More Than 5 Buffers Linked to a CAN Module

Message Buffer Address Offset[Note1]	Message Buffer Number	Message Buffer Link	Message Buffer Type[Note2]	Identifier
7E0H	63			
. . .	. . .			
300H	24			
2E0H	23			
2C0H	22	CAN1	TRX	005H
2A0H	21			
280H	20			
260H	19	CAN1	TRX	006H
240H	18			
220H	17			
200H	16			
1E0H	15	CAN1	TRX	007H
1C0H	14	CAN1	TRX	001H [Note 3]
1A0H	13			
180H	12			
160H	11			
140H	10	CAN1	TRX	003H [Note 3]
120H	9			
100H	8			
0E0H	7			
0C0H	6	CAN1	TRX	000H [Note 3]
0A0H	5			
080H	4			
060H	3			
040H	2	CAN1	TRX	004H [Note 3]
020H	1	CAN1	TRX	002H [Note 3]
000H	0			

Notes: **1.** The address of a message buffer entry is calculated according to the following formula:
effective address = PP_BASE + address offset

2. TRX = transmit message

3. 5 higher prior transmit messages assigned to messages buffer with lower address values.

(2) Message reception

Due to the vast initialisation possibilities for each message buffer in the FCAN system, it is possible that a received message fits in several message buffers assigned to a CAN module.
A fixed rule according to the priority classes has been implemented to avoid arbitrary message storage and uncontrolled behaviour.
The storage priority for data frames and for remote frames is different (refer to Table 13-12 and Table 13-13).

Table 13-12: Storage Priority for Reception of Data Frames

Priority Class	Condition
1 (high)	received data frame fits in non-masked receive buffer
2	received data frame fits in receive buffer linked to mask 0
3	received data frame fits in receive buffer linked to mask 1
4	received data frame fits in receive buffer linked to mask 2
5 (low)	received data frame fits in receive buffer linked to mask 3

Table 13-13: Storage priority for Reception of Remote Frames

Priority Class	Condition
1 (high)	received remote frame fits in transmit buffer
2	received remote frame fits in non-masked receive buffer
3	received remote frame fits in receive buffer linked to mask 0
4	received remote frame fits in receive buffer linked to mask 1
5	received remote frame fits in receive buffer linked to mask 2
6 (low)	received remote frame fits in receive buffer linked to mask 3

Caution: A priority class with lower priority don’t provide a backup for classes with higher priority. That means that a message (i.e. data frame / remote frame) is explicitly stored in the priority class with higher priority and never stored in the lower prior class.

Example:

Two receive message buffers are linked to CAN module 1:

- Buffer 1: non-masked receive buffer with identifier ID_K
- Buffer 2: receive buffer with ID_K linked to mask 2.

Under that configuration a message with ID_K is never stored in the receive buffer linked to mask 2, but always into the non-masked receive buffer.

Furthermore, there is a fixed inner storage rule in case several buffers of the same priority class are linked to a CAN module. For the inner priority class storage rule the data new flag (DN) in the M_STATm register is the first storage criteria (m = 00 to 63).
Whenever the DN flag cannot provide an unambiguous criteria for storing the message (i.e. there are several message buffers of the same priority class with DN flag set or not set) the physical message buffer number is chosen as the second criteria.

Table 13-14: Inner Storage Priority Within a Priority Class

Priority	First Criteria	Priority	Second Criteria
1 (high)	DN flag not set	1 (high)	lowest physical message buffer number
		2 (low)	next physical message buffer number
2 (low)	DN flag set	1 (high)	lowest physical message buffer number
		2 (low)	next physical message buffer number

Example:

When the very first message is received, which fits into several message buffer of the same priority class, the DN flag in all buffers is not set, hence that message is stored in the buffer with the lowest physical buffer number. Subsequent messages are stored to the message buffers in ascending message buffer number order as long the DN flags remains as set into the buffer of the previous message storage.
As soon the CPU reads one of the message buffer with DN flag set and then clears the DN flag, the storing in ascending message buffer number order is interrupted.

Due to the storage priority for receive messages it is possible to design multiple buffer arrays for a CAN message – while not all message buffers assigned to the same identifier contain new data (DN flag set) the FCAN system will store the data in the next free message buffer (DN flag cleared).

13.2.7 Mask handling

The FCAN system supports two concepts of message reception, the BasicCAN concept and the Full-CAN concept.
In the Full-CAN concept a particular message buffer accepts only one single message, hence there is no further sorting and filtering required by software. As a consequence only one unambiguous identifier is assigned to a message buffer.

In the BasicCAN concept a receive message buffer operates as a channel, which can accept several messages. After reception software must sort, respectively filter, which particular message has been received.
By the usage of hardware masks the range of receive messages can be limited to reduce the CPU load caused by message sorting.

In the FCAN system each CAN module provides 4 different masks.
For a receive message buffer assigned to a CAN module one of the 4 masks can be selected when the BasicCAN concept is used.

When using a mask, a certain identifier value must be written into the identifier register M_IDm (equals 32 bit value build by M_IDHm and M_IDLm) of the receive message buffer at initialisation.
Then the linked mask CxMASKn composed from CxMASKHn and CxMASKLn determines which identifier bits of a received message must match exactly to accept the received message for the message buffer.
The mask facilitates that certain identifier bits of the received message will not be compared with the corresponding identifier bits of the message buffer, thus several messages might be accepted for the receive message buffer.

Remarks: **1.** n = 0 to 3
2. m = 00 to 63
3. x = 1 to 3

13.2.8 Remote frame handling

The FCAN macro offers enhanced features for generating remote frames and for the reaction of a CAN module upon remote frames.

(1) Generation of a remote frame

According to the CAN specification a remote frame has the same format as a data frame except the RTR bit of the control field, which has recessive level, and the data field, which is omitted completely.
By means of a remote frame, receiving nodes can request the transmitting node of a particular message for sending an update of that message to the CAN bus. Usually remote frames are generated from CAN nodes which do not provide the requested message by themselves.

In the FCAN system a remote frame is automatically sent, when setting the transmit request bit (TRQ) of the M_STATm register for a message buffer defined as receive message buffer (m = 00 to 63). Same as for generating a data frame from a transmit message buffer, the ready bit (RDY) of M_STATm register must be set (1).

Remote frames can also be generated by means of a transmit message buffer by setting the RTR bit of the M_CTRLm register, and using the same transmission procedure as for data frames. However, from application point of view that method is not recommended, because it consumes message buffer resources unnecessarily. A data frame in a CAN network can be provided, i.e. transmitted, by only one node. All other nodes in the network may receive that data frame. Using a transmit message buffer for a remote frame generation means that two message buffers for handling of one message within one node are required - one receive message buffer for the reception of a data frame, and the transmit message buffer explicitly for the remote frame generation.

(2) Reception of a remote frame

The FCAN allows the reception of remote frames in message buffers defined for reception or for transmission.

(a) Reception in a receive message buffer

If a remote frame is received in a message buffer m (m = 00 to 63) configured for reception, the following message buffer information will be updated:

M_DLCm	message data length code register
M_CTRLm	message control register
M_TIMEm	message time stamp register (16-bit)
M_DATAm0	message data byte 0
M_DATAm1	message data byte 1
M_DATAm2	message data byte 2
M_DATAm3	message data byte 3
M_DATAm4	message data byte 4
M_DATAm5	message data byte 5
M_DATAm6	message data byte 6
M_DATAm7	message data byte 7
M_IDLm	message identifier register (lower half word)
M_IDHm	message identifier register (upper half word)
M_STATm	message status register

Remarks: **1.** Receiving a remote frame in a receive message buffer does not activate any automatic remote frame handling activities from the FCAN system. The application software must handle the remote frame in the expected way.

2. RMDE0, RMDE1 bits as well as ATS bit of M_CTRLm register are set to 0.

(b) Reception in a transmit message buffer

When the FCAN system searches for the corresponding message buffer after reception of a remote frame and finds a message buffer with a matching identifier, which is defined for transmission, the content of the remote frame is not stored but programmable reactions are launched.
Accepting a remote frame for a transmit message buffer does not change the content of the transmit message buffer except the DN flag of the M_STATm register depending on the setting of the RMDE0, RMDR1 and RTR bits of the M_CTRLm register (refer to Table 13-15).
The remote frame reception in a transmit message buffer causes a reaction according to the setting of the RMDE0, RMDR1 bits and the RTR bit of the M_CTRLm register. The following reactions are programmable:

- Generation of an auto-answer (i.e. TRQ bit of the transmit message buffer is automatically set without any CPU interaction).
- Signalling the remote frame reception by updating the DN flag in the transmit message buffer.
- No reaction at all.

Table 13-15 shows the detailed handling (reaction) upon the reception of a remote frame for a transmit message buffer depending on the settings of RMDE0, RMDE1 and RTR flags.

Table 13-15: Remote Frame Handling upon Reception into a Transmit Message Buffer

M_CTRLm setting			Resulting Automatic Remote Frame Handling	
RMDE0	RMDE1	RTR	DN flag	other actions
0	0	x	no change	– („ignore remote frame“)
1	0	0	Clear when transmit message buffer sent successfully	send transmit message buffer (data frame) as an automatic answer.
		1	no change	_ **Note**
0	1	x	DN is set upon reception	–
1	1	0	Clear when transmit message buffer sent successfully	send transmit message buffer (data frame) as an automatic answer.
		1	DN is set upon reception	_ **Note**

Note: Auto-answer upon remote frame is suppressed, because the transmit message buffer is configured to send a remote frame (RTR = 1).

Remarks: **1.** In case a remote frame is automatically answered upon receiving a remote frame for a transmit message buffer, the reception of the remote frame is not notified by a receive interrupt. However, the successful transmission of the data frame (i.e. the automatic answer) is notified by the corresponding transmit interrupt.
2. m = 00 to 63

13.2.9 FCAN System Event Handling

The modular designed FCAN system allows the connection not only of CAN modules to the message buffers, but also of other machines assigned to do (for example) data or time management. The requirements of a modular concept allowing several CAN modules and other machines to be operated on the same "system" at the same time is similar to the requirements of a multi-tasking operating system. For such an operating system a mechanism must be foreseen which allows information exchange between the tasks without the need that each tasks knows every detail of the other tasks running on the system. A possible concept that meets such requirements is that a task stores its information in an area accessible for all tasks (global area) and to set some flags which indicate that new information is available.
This principle is also used for the FCAN implementation. The exchange of information is done using an area that is accessible by all modules – the message buffer area - and by defining some flags for the information exchange. These flags are located within each message buffer, allowing to set individually whether the information of a message buffer should be known only be the assigned CAN module or but other machines as well. Setting the flags is event driven, which means the occurrence of an event is necessary. Such events can be either caused by CAN modules (e.g. reception or transmission of messages from and to message buffers) or by any other machine (e.g. time management unit).

(a) CAN Module Events

Each message buffer m (m = 00 to 63) has an event request flag (ERQ flag in the appropriate M_STATm register) that indicates whether event processing for this message is pending. In addition to this flag two 8-bit event pointers (M_EVTm0 and M_EVTm1) are available.
M_EVTm0 is assigned for an event caused either by receiving a new data frame into the message buffer or by the successful transmission of a data frame from the message buffer.
M_EVTm1 is assigned for an event caused by receiving a remote frame into the message buffer, if the message is defined as transmit message buffer.

Caution: To identify a remote frame on a transmit message, the RMDE0 has to cleared (0) ("auto-answering off") and the RMDE1 has to set (1) (set DN flag when receiving remote frame). When setting the TRQ flag of the message, the DN flag must be cleared at the same time.

If one of the two events occurs the CAN module checks the value of the related event pointer. If the pointer is set to zero, no event processing is executed. If the value is different from zero the CAN module will set the ERQ flag for the message. This flag is checked by an event-processing machine and will cause the start of the event processing sequence.

(b) Machine Events

Machines can also generate events. In principle the event handling for machine events is similar to that of CAN module events, although there are some differences in the use of the message buffer assigned to the machine. The structure of the message buffer will be adapted to the needs of the machine, as it is not used to store CAN bus information, but the ERQ flag is still available. By testing this flag the event-processing machine will see whether an event processing sequence should be started or not.

(c) Timer Events

In addition to CAN module and machine events, there are also four timer events available. The timer events are derived from the 16-bit free-running counter CGTSC also used for time stamping. (refer to chapter **13.3.2 (5)CAN timer event enable register (CGTEN)**)

For each of the four timer events, a unique event pointer can be defined.

13.3 Control and Data Registers

13.3.1 Bit set/clear function

Direct writing of data (bit operations, read-modify write, direct writing of a target value) is not allowed to few specific registers, where bit setting and bit clearing might be performed by CPU and by the FCAN system. The following registers of the FCAN system are concerned.

- CAN global status register (CGST)
- CAN global interrupt enable register (CGIE)
- CAN global interrupt pending register (CGINTP)
- CAN x interrupt pending registers (CxINTP)
- CAN x control registers (CxCTRL)
- CAN x definition registers (CxDEF)
- CAN x interrupt enable registers (CxIE)
- CAN x bus activity registers (CxBA)
- ELISA status register (ELSR)

Remark: x = 1 to 3

Registers like above, where bit access and direct write operations are prohibited, are organized in such a way that all bits allowed for manipulation are located in the lower byte (bits 7 to 0), while in the upper byte (bits 15 to 8) either no or read-only information is located.

The registers can be read in the usual way to get all 16 data bits in their actual setting (ref. to appropriated register description).

For setting or clearing any of the lower 8 bits the following mechanism is implemented:
When writing 16-bit data to the register address, each of the lower 8 data bits indicates whether the corresponding register bit should be cleared (data bit set) or remain unchanged (data bit not set). Each of the upper 8 data bits indicates whether the corresponding register bit should be set (data bit set) or remain unchanged (data bit cleared).
The organization of 16-bit data write for such registers is shown in Figure 13-7.

Figure 13-7: 16-Bit Data Write Operation for Specific Registers

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ST_7	ST_6	ST_5	ST_4	ST_3	ST_2	ST_1	ST_0	CL_7	CL_6	CL_5	CL_4	CL_3	CL_2	CL_1	CL_0

<table>
<tr><th>Bit Name</th><th>Function</th></tr>
<tr><td>ST_n</td><td>Sets the register bit n.
0: No change of register bit n
1: Register bit n is set (1)</td></tr>
<tr><td>CL_n</td><td>Clears the register bit n.
0: No change of register bit n
1: Register bit n is cleared (0)</td></tr>
<tr><td>ST_n, CL_n</td><td>Sets/clears the Register bit n.
<table>
<tr><th>ST_n</th><th>CL_n</th><th>Status of Register Bit n</th></tr>
<tr><td>0</td><td>1</td><td>Register bit n is cleared (0)</td></tr>
<tr><td>1</td><td>0</td><td>Register bit n is set (1)</td></tr>
<tr><td colspan="2">Others</td><td>No change in register bit n value.</td></tr>
</table>
</td></tr>
</table>

Remarks:

1. If only bits are to be cleared, the 16-bit write access can be replaced by an 8-bit write access to the register address. If only bits are to be set, the 16-bit write access can be replaced by an 8-bit write access to the register address+1. Nevertheless, for better visibility of the program code it is recommended to perform only 16-bit write accesses.

2. n = 0 to 7

13.3.2 Common registers

(1) CAN stop register (CSTOP)

The CSTOP register controls the clock supply of the FCAN system.
This register can be read/written in 8-bit and16-bit units.

Figure 13-8: CAN Stop Register (CSTOP)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CSTOP	CSTP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	80CH	0000H

Bit Position	Bit Name	Function
15	CSTP	Controls the clock supply for the complete FCAN system. The CSTP flag can be used to reduce the power consumption when the FCAN system is set to SLEEP mode and STOP mode to a minimum. 0: FCAN system is supplied with clock f_{MEM}. 1: Clock supply of the FCAN system is stopped. **Remark:** When switching off the clock supply of the FCAN system during SLEEP mode, wake-up by CAN bus activity is possible. But, instead of CxINT4 interrupt (i.e. wake-up from SLEEP mode interrupt), the GINT3 interrupt must be used. **Cautions: 1. In case CSTP is set (1), access to the register and buffer of the FCAN system is impossible, except access to the CSTOP register.** **2. Do not set CSTP = 1 while the FCAN system is under normal operation, especially while a CAN module handles messages on the CAN bus. A sudden stop of the FCAN system might cause malfunctions of the entire CAN network.**

Note: The address of an interrupt pending register is calculated according to the following formula:
effective address = PP_BASE + address offset

(2) CAN main clock select register (CGCS)

The CGCS register controls the internal memory access clock (f_{MEM}), which is used as main clock for each CAN module, as well as the global time system clock (f_{GTS}), used for the time stamp function and event generation. (For details refer to chapter **13.2.3 Clock structure**)
These register can be read/written in 1-bit, 8-bit and16-bit units.

Figure 13-9: CAN Main Clock Select Register (CGSC) (1/2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CGCS	CGTS7	CGTS6	CGTS5	CGTS4	CGTS3	CGTS2	CGTS1	CGTS0	GTSC0	GTSC0	0	MCS	MCP3	MCP2	MCP1	MCP0	814H	7F05H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15 to 8</td><td>CGTS7 to CGTS0</td><td>Specifies the 8-bit prescaler compare value for the global time system clock (f_{GTS}) (ref. to Fig. 13-11).
<table>
<tr><th>CGTS7 to CGTS0 (k)</th><th>Prescaler (k + 1)</th><th>Global Time System Clock
$f_{GTS} = f_{GTS1} / (k + 1)$</th></tr>
<tr><td>0</td><td>1</td><td>$f_{GTS} = f_{GTS1}$</td></tr>
<tr><td>1</td><td>2</td><td>$f_{GTS} = f_{GTS1} / 2$</td></tr>
<tr><td>2</td><td>3</td><td>$f_{GTS} = f_{GTS1} / 3$</td></tr>
<tr><td>⋮</td><td>⋮</td><td>⋮</td></tr>
<tr><td>255</td><td>256</td><td>$f_{GTS} = f_{GTS1} / 256$</td></tr>
</table>
<b>Remark:</b> The global time system clock is the source clock for the 16-bit timer used for the time stamp functionality. This clock is common for all CAN modules.</td></tr>
<tr><td>7, 6</td><td>GTCS1, GTCS0</td><td>Selects the global time system basic clock (f_{GTS1}) from the memory clock (f_{MEM}) (ref. to Fig. 13-11).
<table>
<tr><th>GTCS1</th><th>GTCS0</th><th>Global Time System Basic Clock (f_{GTS1})</th></tr>
<tr><td>0</td><td>0</td><td>$f_{GTS1} = f_{MEM} / 2$</td></tr>
<tr><td>0</td><td>1</td><td>$f_{GTS1} = f_{MEM} / 4$</td></tr>
<tr><td>1</td><td>0</td><td>$f_{GTS1} = f_{MEM} / 8$</td></tr>
<tr><td>1</td><td>1</td><td>$f_{GTS1} = f_{MEM} / 16$</td></tr>
</table></td></tr>
<tr><td>4</td><td>MCS</td><td>Selects input clock for the memory access clock prescaler (f_{MEM1}) (ref. to Fig. 13-10).
0: f_{MEM1} = internal system clock (f_{CPU})
1: f_{MEM1} = external clock input (f_{EXT})[Note]

<b>Note:</b> If the external clock input is selected and $f_{MEM} = f_{MEM1}$ is selected (MCP3 to MCP0 = 0000B), the duty cycle of the external clock f_{EXT} must be 50%.</td></tr>
</table>

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = PP_BASE + address offset

Figure 13-9: CAN Main Clock Select Register (CGSC) (2/2)

Bit Position	Bit Name	Function
3 to 0	MCP3 to MCP0	Specifies the prescaler for the memory access clock (f_{MEM}) (ref. to Fig. 13-10).

MCP3	MCP2	MCP1	MCP0	Prescaler (m+1)	Memory Clock $f_{MEM} = f_{MEM1}$ / (m+1)
0	0	0	0	1	$f_{MEM} = f_{MEM1}$
0	0	0	1	2	$f_{MEM} = f_{MEM1}$ / 2
0	0	1	0	3	$f_{MEM} = f_{MEM1}$ / 3
⋮				⋮	
1	1	1	1	16	$f_{MEM} = f_{MEM1}$ / 16

Figure 13-10: Configuration of FCAN System Main Clock

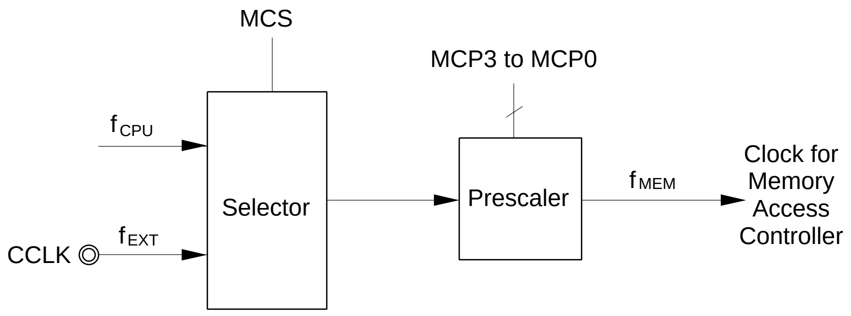
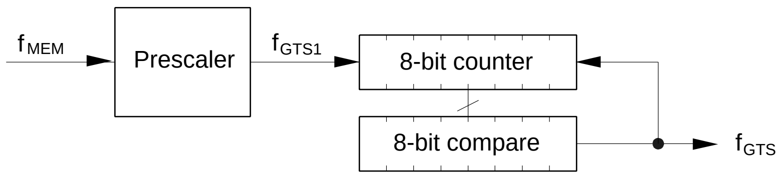


Figure 13-11: Configuration of FCAN Global Time System Clock



Remark: The global time system clock (f_{GTS}) is also the clock base for the timer event module in the CAN bridge ELISA, meaning all time events are derived from this clock. See timer event module in the CAN bridge ELISA description for details.

(3) CAN global status register (CGST)

The CGST register indicates and controls the operation modes of the FCAN system. Additionally the version number of the FCAN system can be obtained.

This register can be read in 1-bit, 8-bit and 16-bit units. It can be written in 16-bit units only. For setting and clearing certain bits a special set/clear method applies. (Refer to chapter 13.3.1)

Figure 13-12: CAN Global Status Register (CGST) (1/2)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CGST	0	0	0	0	0	0	0	1	MERR	0	0	0	EFSD	TSM	EVM	GOM	810H	0100H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CGST	0	0	0	0	ST_EFSD	ST_TSM	ST_EVM	ST_GOM	CL_MERR	0	0	0	CL_EFSD	CL_TSM	CL_EVM	CL_GOM	810H

Read (1/2)

Bit Position	Bit Name	Function
7	MERR	Indicates the error status of the memory access controller (MAC). 0: No error occurrence 1: At least one error occurred since the flag was cleared last A MAC error occurs under the following conditions: - An attempt to clear the GOM flag was performed although not all CAN modules are set to initialization state. - Access to an illegal address, or access is prohibited by MAC (see GOM flag description below)
3	EFSD	Enable forced shut down. 0: Forced shut down is disabled. 1: Forced shut down is enabled. **Remark:** In case of an emergency it might be necessary to reset all CAN modules immediately. In this case the EFSD flag has to be set before clearing the GOM flag.
2	TSM	Indicates the operating mode of the CAN global time system counter (CGTSC). 0: CAN global time system counter is stopped. 1: CAN global time system counter is operating.
1	EVM	Indicates the event operating mode. 0: CAN bridge ELISA is disabled. 1: CAN bridge ELISA is enabled. **Remark:** The EVM bit enables CAN bridge ELISA. Clearing the bit might need some time depending on the actual status of the CAN bridge ELISA. To ensure that the event operations are really disabled (e.g. when stopping then CAN interface to save power) the software has to wait until the EVM bit is finally cleared (0).

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 13-12: CAN Global Status Register (CGST) (2/2)

Read (2/2)

Bit Position	Bit Name	Function
0	GOM	Indicates the global operating mode. 0: Access to CAN module registers is prohibited, except mask registers and temporary buffers.[Note 1] 1: Operation of all CAN modules are enabled. Temporary buffers can be read only.[Note 1] **Caution: To ensure that resetting the CAN modules do not cause any unexpected behaviour on the CAN bus, the GOM flag can only be cleared, if all CAN modules are set into initialisation state (exception: forced-shut-down, see EFSD flag). If the software clears the flag while at least one CAN module is still not in initialisation state (ISTAT flag of CxCTRL register (x = 1 to 3) is set (1)), the GOM flag remains set.**

Write

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>11, 3</td><td>ST_EFSD,
CL_EFSD</td><td>Sets/clears the EFSD bit.
<table>
<tr><th>ST_EFSD</th><th>CL_EFSD</th><th>Status of EFSD Bit</th></tr>
<tr><td>0</td><td>1</td><td>EFSD bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>EFSD bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in EFSD bit value.</td></tr>
</table></td></tr>
<tr><td>10, 2</td><td>ST_TSM,
CL_TSM</td><td>Sets/clears the TSM bit.
<table>
<tr><th>ST_TSM</th><th>CL_TSM</th><th>Status of TSM Bit</th></tr>
<tr><td>0</td><td>1</td><td>TSM bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>TSM bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in TSM bit value.</td></tr>
</table></td></tr>
<tr><td>9, 1</td><td>ST_EVM,
CL_EVM</td><td>Sets/clears the EVM bit.
<table>
<tr><th>ST_EVM</th><th>CL_EVM</th><th>Status of EVM Bit</th></tr>
<tr><td>0</td><td>1</td><td>EVM bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>EVM bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in EVM bit value.</td></tr>
</table></td></tr>
<tr><td>8, 0</td><td>ST_GOM,
CL_GOM</td><td>Sets/clears the GOM bit.
<table>
<tr><th>ST_GOM</th><th>CL_GOM</th><th>Status of GOM Bit</th></tr>
<tr><td>0</td><td>1</td><td>GOM bit is cleared (0).[Note 2]</td></tr>
<tr><td>1</td><td>0</td><td>GOM bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in GOM bit value.</td></tr>
</table></td></tr>
<tr><td>7</td><td>CL_MERR</td><td>Clears the MERR bit.
0: No change of MERR bit.
1: MERR bit is cleared (0).</td></tr>
</table>

Notes: **1.** Access to the message buffer area is not affected.
2. Refer to description of GOM flag above.

(4) CAN global interrupt enable register (CGIE)

The CGIE register enables the global interrupts of the FCAN system.
This register can be read in 1-bit, 8-bit and16-bit units. It can be written in 16-bit units only. For setting and clearing certain bits a special set/clear method applies. (Refer to chapter 13.3.1)

Figure 13-13: CAN Global Interrupt Enable Register (CGIE) (1/2)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CGIE	0	0	0	0	1	0	0	01	G_IE7	0	0	0	0	G_IE2	G_IE1	0	812H	0000H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CGIE	ST_ G_IE7	0	0	0	0	ST_ G_IE2	ST_ G_IE1	0	CL_ G_IE7	0	0	0	0	CL_ G_IE2	CL_ G_IE1	0	812H

Read

Bit Position	Bit Name	Function
7	G_IE7	Enables interrupt by CAN bridge ELISA. 0: Interrupt disabled 1: Interrupt enabled
2	G_IE2	Enables illegal address interrupt. 0: Interrupt disabled 1: Interrupt enabled **Remarks:** **1.** Interrupt signals an illegal address access (refer to Figure 13-2). **2.** Interrupt signals a write access to temporary buffer while GOM bit of the CGST register is set (1).
1	G_IE1	Enables invalid write access interrupt. 0: Interrupt disabled 1: Interrupt enabled **Remarks:** **1.** Interrupt signals a write access to a CAN module register while GOM bit of the CGST register is cleared (0). **2.** Interrupt signals an illegal FCAN system shut down, i.e. GOM bit is going to be cleared while at least one of the CAN modules is not in initialisation state.

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 13-13: CAN Global Interrupt Enable Register (CGIE) (2/2)

Write

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15, 7</td><td>ST_G_IE7,
CL_G_IE7</td><td>Sets/clears the G_IE7 bit.
<table>
<tr><th>ST_G_IE7</th><th>CL_G_IE7</th><th>Status of G_IE7 Bit</th></tr>
<tr><td>0</td><td>1</td><td>G_IE7 bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>G_IE7 bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in G_IE7 bit value.</td></tr>
</table></td></tr>
<tr><td>10, 2</td><td>ST_G_IE2,
CL_G_IE2</td><td>Sets/clears the G_IE2 bit.
<table>
<tr><th>ST_G_IE2</th><th>CL_G_IE2</th><th>Status of G_IE2 Bit</th></tr>
<tr><td>0</td><td>1</td><td>G_IE2 bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>G_IE2 bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in G_IE2 bit value.</td></tr>
</table></td></tr>
<tr><td>9, 1</td><td>ST_G_IE1,
CL_G_IE1</td><td>Sets/clears the G_IE1 bit.
<table>
<tr><th>ST_G_IE1</th><th>CL_G_IE1</th><th>Status of G_IE1 Bit</th></tr>
<tr><td>0</td><td>1</td><td>G_IE1 bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>G_IE1 bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in G_IE1 bit value.</td></tr>
</table></td></tr>
</table>

(5) CAN timer event enable register (CGTEN)

The CGTEN register enables/disables the 4 timer events.
This register can read and written in 1-bit, 8-bit and 16-bit units.

Figure 13-14: CAN Timer Event Enable Register (CGTEN)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CGTEN	0	0	0	0	0	0	0	0	0	0	0	0	CTEN3	CTEN2	CTEN1	CTEN0	818H	0000H

Read

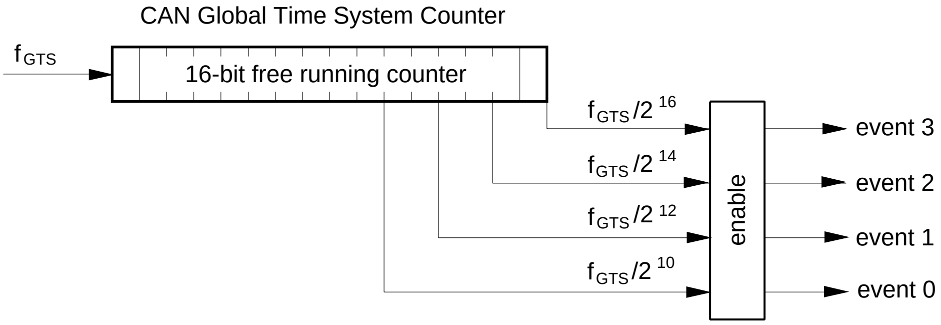
Bit Position	Bit Name	Function
3	CTEN3	Enables CAN timer event 3. 0: Timer event disabled 1: Timer event enabled
2	CTEN2	Enables CAN timer event 2. 0: Timer event disabled 1: Timer event enabled
1	CTEN1	Enables CAN timer event 1. 0: Timer event disabled 1: Timer event enabled
0	CTEN0	Enables CAN timer event 0. 0: Timer event disabled 1: Timer event enabled

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

The timer events are as follows:

- timer event 0: $f_{GTS} / 2^{10}$
- timer event 1: $f_{GTS} / 2^{12}$
- timer event 2: $f_{GTS} / 2^{14}$
- timer event 3: $f_{GTS} / 2^{16}$

Figure 13-15: CAN Global Time System Counter and event generation



Caution: If the event processing is disabled (EVM bit of CGST register = 0), the CTENx flags have no function (x = 0 to 3).

(6) CAN global time system counter (CGTSC)

The CGTSC register holds the value of the free-running 16-bit CAN global time system counter. (For details refer to chapters **13.2.3 Clock structure** and **13.2.5 Time stamp**)

This register can be read and written[Note 1] in 16-bit units only.

Figure 13-16: CAN Global Time System Counter (CGTSC)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note2]	Initial value
CGTSC	TSC15	TSC14	TSC13	TSC12	TSC11	TSC10	TSC9	TSC8	TSC7	TSC6	TSC5	TSC4	TSC3	TSC2	TSC1	TSC0	818H	0000H

Notes: **1.** When writing is performed to CGTSC register, the counter is cleared to 0.
2. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Remark: The CGTSC register can be read at any time.

(7) CAN message search start register (CGMSS)

The CGMSS register controls the start of a message search. It can be used for a fast message retrieval within the message buffers matching a search criteria (e.g. messages with DN flag set). This register is write-only and must be written in 16-bit units.

Figure 13-17: CAN Message Search Start Register (CGMSS)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CGMSS	CIDE	CERQ	CTRQ	CMSK	CDN	0	SMN1	SMN0	0	0	STRT5	STRT4	STRT3	STRT2	STRT1	STRT0	81AH	–

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15</td><td>CIDE</td><td>Search criteria for message identifier type (IDE).
0: Do not check status of the message identifier type.
1: Message identifier type must be standard identifier (IDE = 0).</td></tr>
<tr><td>14</td><td>CERQ</td><td>Search criteria for event processing request flag (ERQ) of the M_STATm registers.
0: Do not check status of the ERQ flag.
1: ERQ flag must be set.</td></tr>
<tr><td>13</td><td>CTRQ</td><td>Search criteria for transmit request flag (TRQ) and message ready flag (RDY) of the M_STATm registers.
0: Do not check status of TRQ flag and RDY flag.
1: TRQ flag and RDY flag must be set.</td></tr>
<tr><td>12</td><td>CMSK</td><td>Search criteria for the mask link bits MT2 to MT0 of the M_CONFm registers.
0: Do not check mask link bits.
1: Check only message buffers not linked with a mask.</td></tr>
<tr><td>11</td><td>CDN</td><td>Search criteria for data new flag (DN) of the M_STATm registers.
0: Do not check status of the DN flag.
1: DN flag must be set.</td></tr>
<tr><td>9, 8</td><td>SMN1,
SMN0</td><td>Specifies the CAN module number to search for.
<table>
<tr><th>SMN1</th><th>SMN0</th><th>CAN Module Number</th></tr>
<tr><td>0</td><td>0</td><td>Search for message buffers not linked to any CAN module.</td></tr>
<tr><td>0</td><td>1</td><td>Search for message buffers linked to CAN module 1</td></tr>
<tr><td>1</td><td>0</td><td>Search for message buffers linked to CAN module 2</td></tr>
<tr><td>1</td><td>1</td><td>Search for message buffers linked to CAN module 3</td></tr>
</table>
<b>Remark:</b> The SMNO2 to SMNO0 bits define which messages are checked by the message search. Only messages assigned to the CAN module defined by SMNO2 to SMNO0 are checked, all other messages are ignored.</td></tr>
<tr><td>5 to 0</td><td>STRT5 to STRT0</td><td>Specifies the number of message buffer the search starts for. (0 to 63)
<b>Remarks:</b> <b>1.</b> Any search will start from the message number defined by STRT5 to STRT0 and end at the highest available message buffer. If a search results in multiple matches, the lowest buffer number is returned.
<b>2.</b> To get the next match without modifying the search criteria the STRT5 to STRT0 bits must be set to the succeeding number of the found one in (MFND5 to MFND0) of the CGMSR register.</td></tr>
</table>

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = PP_BASE + address offset

Remark: m = 00 to 63

(8) CAN message search result register (CGMSR)

The CGMSR register returns the result of a message search, started by writing the CGMSS register.

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

Figure 13-18: CAN Message Search Start Register (CGMSS)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note1]	Initial value
CGMSR	0	0	0	0	0	0	MM	AM	0[Note3]	0[Note3]	MFND5	MFND4	MFND3	MFND2	MFND1	MFND0	81AH	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>9, 8</td><td>MM,
AM</td><td>Indicates the match result of the preceding message search.
<table><tr><th>MM</th><th>AM</th><th>Number of Hits</th></tr><tr><td>×</td><td>0</td><td>No match</td></tr><tr><td>0</td><td>1</td><td>1 message meets the search criteria.</td></tr><tr><td>1</td><td>1</td><td>Several message meet the search criteria.[Note 2]</td></tr></table></td></tr>
<tr><td>5 to 0</td><td>MFND5 to MFND0</td><td>Indicates the number of the message buffer, which was found by the message search. (0 to 63)[Note 2]
<b>Remarks:</b> <b>1.</b> Any search will start from the message number defined by STRT5 to STRT0 and end at the highest available message buffer. If a search results in multiple matches, the lowest buffer number is returned.
<b>2.</b> To get the next match without modifying the search criteria the STRT5 to STRT0 bits must be set to the succeeding number of the found one in (MFND5 to MFND0) of the CGMSR register.</td></tr>
</table>

Notes: **1.** The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

2. If a message search finds several message buffers meeting the search option, the MM flag is set. In that case the MFND5 to MFND0 bits return number of the message buffer with the lowest number.

3. Value of Bits 6 and 7 is undefined after search function.

(9) CAN test bus register (CTBR)

For test purposes an internal test bus is available. The CTBR register controls this test bus capability.

This register can be read and written in 8-bit and 16-bit units.

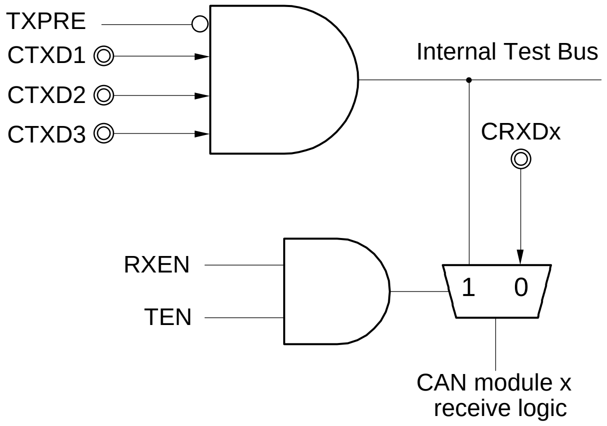
Figure 13-19: CAN Test Bus Register (CTBR)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CTBR	0	0	0	0	0	0	0	0	0	0	0	0	0	RXEN	TXPRE	TEN	81CH	0000H

Bit Position	Bit Name	Function
2	RXEN	Enables the receive line. 0: CAN module receive lines are input from the corresponding CANxRX pins. 1: CAN module receive lines are input from to the internal test bus.
1	TXPRE	Presets the transmit lines. 0: No preset on the transmit lines. 1: Error injection into the internal test bus by forcing all transmit pins to an dominant level.
	TEN	Enables internal test bus. 0: Internal test bus is disabled. 1: Internal test bus is enabled.

The figure below shows the structure of the internal CAN test bus.

Figure 13-20: Internal CAN Test Bus Structure



Remarks: **1.** Both, TEN bit and RXEN bit must be set (1) to use the internal CAN bus.
2. Using the internal CAN bus connects all CAN modules (CAN module 1 to CAN module 3) to one internal CAN bus. The internal CAN bus is used to operate the FCAN system without any external hardware (e.g. CAN transceiver, bus harness, etc.).
3. x = 1 to 3

Caution: The internal test bus must only be used when none of the CAN modules are connected to a CAN bus.

13.3.3 CAN interrupt pending registers

(1) CAN interrupt pending register (CCINTP)

The CCINTP register summarizes all grouped interrupt pending signals. Each of them is assigned to an unambiguous interrupt vector of the V850E/CA1 (Atomic).
This register is read-only and can be read in 8-bit and16-bit units.

Figure 13-21: CAN Interrupt Pending Register (CCINTP)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset Note 1	Initial value
CCINTP	INTACT	INTMAC	0	0	0	0	0	CAN3ERR	CAN3REC	CAN3TRX	CAN2ERR	CAN2REC	CAN2TRX	CAN1ERR	CAN1REC	CAN1TRX	800H	0000H

Bit Position	Bit Name Note 2	Function
15	INTACT	Indicates an interrupt of the CAN bridge ELISA (GINT7 bit of CGINTP register). 0: No Interrupt pending 1: Interrupt pending
14	INTMAC	Indicates a MAC interrupt (OR function of GINT3 to GINT1 bits of CGINTP register). 0: No Interrupt pending 1: Interrupt pending
8, 5, 2	CANxERR	Indicates an error interrupt of CAN module x (OR function of CxINT6 to CxINT2 bits of CGINTP register). 0: No Interrupt pending 1: Interrupt pending
7, 4, 1	CANxREC	Indicates a receive completion interrupt of CAN module x (CxINT1 bit of CGINTP register). 0: No Interrupt pending 1: Interrupt pending
6, 3, 0	CANxTRX	Indicates a transmit completion interrupt of CAN module x (CxINT0 bit of CGINTP register). 0: No Interrupt pending 1: Interrupt pending

Notes: **1.** The register address is calculated according to the following formula:
effective address = PP_BASE + address offset
2. x = 1 to 3

Remark: The CCINTP register is a read-only register, which summarizes the CAN interrupt pending signals. Therefore it cannot be used to clear the interrupt pending signals after servicing. The interrupt pending signals must be cleared in the dedicated interrupt pending registers CGINTP, C1INTP, C2INTP and C3INTP.

(2) CAN global interrupt pending register (CGINTP)

The CGINTP register indicates the global interrupt pending signals. The interrupt pending flags can be cleared by writing to the register according to the special bit-clear method. (Refer to chapter 13.3.1)

This register can be read in 8-bit and 16-bit units. It can be written in 16-bit units only.

Figure 13-22: CAN Global Interrupt Pending Register (CGINTP) (1/2)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CGINTP	0	0	0	0	0	0	0	0	GINT7	0	0	0	GINT3	GINT2	GINT1	0	802H	0000H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CGINTP	0	0	0	0	0	0	0	0	CL_ GINT7	0	0	0	CL_ GINT3	CL_ GINT2	CL_ GINT1	0	802H

Read

Bit Position	Bit Name	Function
7	GINT7	Indicates an interrupt of the CAN bridge ELISA (GINT7 bit of CGINTP register). 0: No Interrupt pending 1: Interrupt pending
3	GINT3	Indicates a wake-up interrupt from CAN sleep mode while clock supply to the FCAN system was stopped (ref. to CSTOP register). 0: No Interrupt pending 1: Interrupt pending
2	GINT2	Indicates an illegal address access interrupt. 0: No Interrupt pending 1: Interrupt pending **Remarks:** **1.** Interrupt signals an illegal address access (refer to Figure 13-2). **2.** Interrupt signals a write access to temporary buffer while GOM bit of the CGST register is set (1).
1	GINT1	Indicates an invalid write access interrupt. 0: No Interrupt pending 1: Interrupt pending **Remarks:** **1.** Interrupt signals a write access to a CAN module register while GOM bit of the CGST register is cleared (0). **2.** Interrupt signals an illegal FCAN system shut down, i.e. GOM bit is going to be cleared while at least one of the CAN modules is not in initialisation state.

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 13-22: CAN Global Interrupt Pending Register (CGINTP) (2/2)

Write

Bit Position	Bit Name	Function
7	CL_GINT7	Clears the interrupt pending bit GINT7. 0: No change of GINT7 bit. 1: GINT7 bit is cleared (0).
3	CL_GINT3	Clears the interrupt pending bit GINT3. 0: No change of GINT3 bit. 1: GINT3 bit is cleared (0).
2	CL_GINT2	Clears the interrupt pending bit GINT2. 0: No change of GINT2 bit. 1: GINT2 bit is cleared (0).
1	CL_GINT1	Clears the interrupt pending bit GINT1. 0: No change of GINT1 bit. 1: GINT1 bit is cleared (0).

Remarks: **1.** The interrupts GINT1, GINT2 and GINT7 are only generated when the corresponding interrupt enable bit in the CGIE register is set.

2. In the CGIE register is no interrupt enable bit implemented for GINT3. Thus this interrupt cannot be disabled.

3. The interrupt pending bits must be cleared by software in the interrupt service routine.

Caution: In case the interrupt pending bit is not cleared by software in the interrupt service routine, no subsequent interrupt is generated anymore.

(3) CAN 1 to 3 interrupt pending registers (C1INTP to C3INTP)

The C1INTP to C3INTP registers indicate the corresponding CAN module interrupt pending signals. The interrupt pending flags can be cleared by writing to the registers according to the special bit-clear method. (Refer to chapter 13.3.1)
This register can be read and written in 8-bit and16-bit units.

Figure 13-23: CAN 1 to 3 Interrupt Pending Registers (C1INTP to C3INTP) (1/2)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset Note 1	Initial value
C1INTP	0	0	0	0	0	0	0	0	0	C1INT6	C1INT5	C1INT4	C1INT3	C1INT2	C1INT1	C1INT0	804H	0000H
C2INTP	0	0	0	0	0	0	0	0	0	C2INT6	C2INT5	C2INT4	C2INT3	C2INT2	C2INT1	C2INT0	806H	0000H
C3INTP	0	0	0	0	0	0	0	0	0	C3INT6	C3INT5	C3INT4	C3INT3	C3INT2	C3INT1	C3INT0	808H	0000H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
C1INTP	0	0	0	0	0	0	0	0	0	CL_ C1INT6	CL_ C1INT5	CL_ C1INT4	CL_ C1INT3	CL_ C1INT2	CL_ C1INT1	CL_ C1INT0	804H
C2INTP	0	0	0	0	0	0	0	0	0	CL_ C2INT6	CL_ C2INT5	CL_ C2INT4	CL_ C2INT3	CL_ C2INT2	CL_ C2INT1	CL_ C2INT0	806H
C3INTP	0	0	0	0	0	0	0	0	0	CL_ C3INT6	CL_ C3INT5	CL_ C3INT4	CL_ C3INT3	CL_ C3INT2	CL_ C3INT1	CL_ C3INT0	808H

Read (1/2)

Bit Position	Bit Name Note 2	Function
6	CxINT6	Indicates a CAN module x error. 0: No Interrupt pending 1: Interrupt pending
5	CxINT5	Indicates a CAN bus error of CAN module x. 0: No Interrupt pending 1: Interrupt pending
4	CxINT4	Indicates a wake-up from sleep mode of CAN module x. 0: No Interrupt pending 1: Interrupt pending
3	CxINT3	Indicates a error passive status on reception of CAN module x. 0: No Interrupt pending 1: Interrupt pending
2	CxINT2	Indicates a error passive or bus off status on transmission of CAN module x. 0: No Interrupt pending 1: Interrupt pending

Notes: **1.** The register address is calculated according to the following formula:
effective address = PP_BASE + address offset
2. x = 1 to 3

Figure 13-23: CAN 1 to 3 Interrupt Pending Registers (C1INTP to C3INTP) (2/2)

Read (2/2)

Bit Position	Bit Name **Note**	Function
1	CxINT1	Indicates a reception completion interrupt of CAN module x. 0: No Interrupt pending 1: Interrupt pending
0	CxINT0	Indicates a transmission completion interrupt of CAN module x. 0: No Interrupt pending 1: Interrupt pending

Write

Bit Position	Bit Name **Note**	Function
6	CL_CxINT6	Clears the interrupt pending bit CxINT6. 0: No change of CxINT6 bit. 1: CxINT6 bit is cleared (0).
5	CL_CxINT5	Clears the interrupt pending bit CxINT5. 0: No change of CxINT5 bit. 1: CxINT5 bit is cleared (0).
4	CL_CxINT4	Clears the interrupt pending bit CxINT4. 0: No change of CxINT4 bit. 1: CxINT4 bit is cleared (0).
3	CL_CxINT3	Clears the interrupt pending bit CxINT3. 0: No change of CxINT3 bit. 1: CxINT3 bit is cleared (0).
2	CL_CxINT2	Clears the interrupt pending bit CxINT2. 0: No change of CxINT2 bit. 1: CxINT2 bit is cleared (0).
1	CL_CxINT1	Clears the interrupt pending bit CxINT1. 0: No change of CxINT1 bit. 1: CxINT1 bit is cleared (0).
0	CL_CxINT0	Clears the interrupt pending bit CxINT0. 0: No change of CxINT0 bit. 1: CxINT0 bit is cleared (0).

Note: x = 1 to 3

Remarks: **1.** The interrupts CxINT1 to CxINT6 are only generated when the corresponding interrupt enable bit in the CGIE register is set.

2. The interrupt pending bits must be cleared by software in the interrupt service routine.

Caution: In case the interrupt pending bit is not cleared by software in the interrupt service routine, no subsequent interrupt is generated anymore.

13.3.4 CAN message buffer registers

(1) Message identifier registers L00 to L63 and H00 to H63 (M_IDL00 to M_IDL63, M_IDH00 to M_IDH63)

The M_IDLm, M_IDHm registers specify the identifier and format of the corresponding message m (m = 00 to 63).
These registers can be read/written 16-bit units.

Figure 13-24: Message Identifier Registers L00 to L63 and H00 to H63 (M_IDL00 to M_IDL63, M_IDH00 to M_IDH63)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note 1]	Initial value
M_IDHm	IDE	0	0	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	012H +(m x 20H)	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
M_IDLm	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	010H +(m x 20H)	undef.

Bit Position	Bit Name	Function
15 (M_IDHm)	IDE	Specifies the format of message identifier. 0: Standard format mode (11-bit) 1: Extended format mode (29-bit)
12 to 0 (M_IDHm)	ID28 to ID16	When IDE = 0 (standard format): ID28 to ID18 specify the 11-bit identifier, where ID28 is the most significant bit. ID17, ID16 contain received data bits.[Note 2, 3] When IDE = 1 (extended format): ID28 to ID16 specify the 13 most significant bits of the 29-bit identifier, where ID28 is the most significant bit.

15 to 0
(M_IDLm)
	ID15 to ID0	When IDE = 0 (standard format):
		ID15 to ID0 contain received data bits.[Note 2, 3]
		When IDE = 1 (extended format):
		ID15 to ID0 specify the 16 least significant bits of the 29-bit identifier.

Notes: **1.** The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

2. In standard format mode (IDE = 0) these bits (ID17 to ID0) are only used for receive message buffers linked to a mask.
- Bits ID17 to ID10 storing the first data byte (D0) is stored, where ID17 is the MSB.
- Bits ID9 to ID2 storing the second data byte (D1), where ID9 is the MSB
- Bits ID1, ID0 contain the two most significant bits 7 and 6 of the third byte (D2)

3. When received message in standard format mode (IDE = 0) has less than 18 data bits, the values of the not received bits are undefined.

Remark: m = 00 to 63

(2) Message configuration registers 00 to 63 (M_CONF00 to M_CONF63)

The M_CONFm registers specify the message type, mask link and CAN module assignment of the corresponding message m (m = 00 to 63).
These registers can be read/written 8-bit units.

Figure 13-25: Message Configuration Registers 00 to 63 (M_CONF00 to M_CONF63)

	7	6	5	4	3	2	1	0	Address Offset[Note 1]	Initial value
M_CONFm	0	0	MT2	MT1	MT0	0	MA1	MA0	014H + (m × 20H)	undef.

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>5 to 3</td><td>MT2 to MT0</td><td>Specifies the message type and mask link.
<table>
<tr><th>MT2</th><th>MT1</th><th>MT0</th><th>Message type and mask link</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Transmit message</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Receive message, no mask linked</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Receive message, mask 0 linked[Note 2]</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Receive message, mask 1 linked[Note 2]</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Receive message, mask 2 linked[Note 2]</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Receive message, mask 3 linked[Note 1]2</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Reserved[Note 3]</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Receive message in diagnostic mode (type 7)[Note 4]</td></tr>
</table></td></tr>
<tr><td>1, 0</td><td>MA1, MA0</td><td>Assigns the message buffer to a CAN module.
<table>
<tr><th>MA1</th><th>MA0</th><th>CAN module assignment</th></tr>
<tr><td>0</td><td>0</td><td>Message buffer is not assigned to a CAN module.[Note 5]</td></tr>
<tr><td>0</td><td>1</td><td>Message buffer is assigned to CAN module 1.</td></tr>
<tr><td>1</td><td>0</td><td>Message buffer is assigned to CAN module 2.</td></tr>
<tr><td>1</td><td>1</td><td>Message buffer is assigned to CAN module 3.</td></tr>
</table></td></tr>
</table>

Notes:

1. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

2. Mask number of the linked CAN module specified by MA1, MA0 bits.

3. CAN module does not handle a message buffer of this type.

4. A message buffer of this type is only handled if the linked CAN module is set to diagnostic mode. In this case all messages received on the CAN bus will be stored in this message buffer, regardless whether they could have been stored in other message buffers as well. Even the type of the identifier (standard or extended) and the type of the frame (remote or data frame) are not respected. In normal operation mode the message buffer is not handled.

5. If the message buffer is not assigned to a CAN module, it can be used as temporary buffer of the application or by the CAN bridge ELISA.

Remark: m = 00 to 63

(3) Message status registers 00 to 63 (M_STAT00 to M_STAT63)

The M_STATm registers indicate transmit and receive status of the corresponding message m (m = 00 to 63). Bits can be set/cleared only by means of the SC_STATm register.
These registers can be read-only in 8-bit units.

Figure 13-26: Message Status Registers 00 to 63 (M_STAT00 to M_STAT63)

	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
M_STATm	0	0	0	0	ERQ	DN	TRQ	RDY	015H + (m × 20H)	undef.

Bit Position	Bit Name	Function
3	ERQ	Indicates a request for event processing by CAN bridge ELISA for this message. 0: No pending event processing. 1: Event processing is pending.
2	DN	Indicates new data received for this message. 0: No new message was received. 1: At least one new message was received. **Remarks:** **1.** If the DN flag is set for a transmit message buffer, it indicates a remote frame reception. In case auto answering (RMDE0 bit of the M_CTRLm register) is active, the DN flag is cleared automatically after the answering data frame is sent. **2.** If the OVM bit of CxCTRL register is cleared (0), a message buffer assigned to the CAN module might be overwritten by new messages, although the DN flag is already set (x = 1 to 3). Checking the MOVR bit of the M_CTRLm register additionally, indicates whether the message buffer has been overwritten. **3.** After copying a received message from the message buffer to the application memory, the DN flag has to be cleared (0) by software.
1	TRQ	Indicates a transmit request of this message. 0: No pending transmit request. 1: Transmit request is pending. **Remark:** If the TRQ flag is set for a receive message, a remote frame is sent. (refer to Table 13-16)
0	RDY	Enables and indicates application processing of this message. 0: Message is processed by the application, and not ready to be handled by the assigned CAN module. 1: Message is ready to be handled by the assigned CAN module. **Remark:** Transmit as well as receive messages are only handled by the assigned CAN module if the RDY flag is set. (refer to Table 13-16)

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Remark: m = 00 to 63

Processing of a transmit or receive message by TRQ and RDY flags is summarized in Table 13-16.

Table 13-16: CAN Message Processing by TRQ and RDY Bits

Message Type	TRQ	RDY	Message Processing
Any	×	0	Message buffer is disabled for any processing by the assigned CAN module.
Receive message	0	1	Message buffer is ready for reception.
	1	1	Request for sending a remote frame.
Transmit message	0	1	No processing of the transmit message.
	1	1	Request for message transmission.

(4) Message set/clear status registers 00 to 63 (SC_STAT0 to SC_STAT63)

The SC_STATm registers set/clear the flags of the corresponding M_STATm registers (m = 00 to 63). By means of this register transmission can be requested and reception can be confirmed. These registers can be written-only in 16-bit units.

Figure 13-27: Message Set/Clear Status Registers 00 to 63 (SC_STAT00 to SC_STAT63)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
SC_STATm	0	0	0	0	ST_ERQ	ST_DN	ST_TRQ	ST_RDY	0	0	0	0	CL_ERQ	CL_DN	CL_TRQ	CL_RDY	016H +(m x 20H)	–

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>11, 3</td><td>ST_ERQ,
CL_ERQ</td><td>Sets/clears the ERQ bit of the M_STATm register.
<table>
<tr><th>ST_ERQ</th><th>CL_ERQ</th><th>Status of ERQ bit</th></tr>
<tr><td>0</td><td>1</td><td>ERQ bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>ERQ bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in ERQ bit value.</td></tr>
</table></td></tr>
<tr><td>10, 2</td><td>ST_DN,
CL_DN</td><td>Sets/clears the DN bit of the M_STATm register.
<table>
<tr><th>ST_DN</th><th>CL_DN</th><th>Status of DN bit</th></tr>
<tr><td>0</td><td>1</td><td>DN bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>DN bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in DN bit value.</td></tr>
</table></td></tr>
<tr><td>9, 1</td><td>ST_TRQ,
CL_TRQ</td><td>Sets/clears the TRQ bit of the M_STATm register.
<table>
<tr><th>ST_TRQ</th><th>CL_TRQ</th><th>Status of TRQ bit</th></tr>
<tr><td>0</td><td>1</td><td>TRQ bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>TRQ bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in TRQ bit value.</td></tr>
</table></td></tr>
<tr><td>8, 0</td><td>ST_RDY,
CL_RDY</td><td>Sets/clears the RDY bit of the M_STATm register.
<table>
<tr><th>ST_RDY</th><th>CL_RDY</th><th>Status of RDY bit</th></tr>
<tr><td>0</td><td>1</td><td>RDY bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>RDY bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in RDY bit value.</td></tr>
</table></td></tr>
</table>

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Remark: m = 00 to 63

(5) Message data registers m0 to m7 (M_DATAm0 to M_DATAm7) (m = 00 to 63)

The M_DATAm0 to M_DATAm7 registers are used to hold the receive or transmit data of the corresponding message m (m = 00 to 63).
These registers can be read/written in 8-bit units.

Figure 13-28: Message Data Registers m0 to m7 (M_DATAm0 to M_DATAm7) (m = 00 to 63)

	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
M_DATAm0	D0_7	D0_6	D0_5	D0_4	D0_3	D0_2	D0_1	D0_0	008H + (m × 20H)	undef.
M_DATAm1	D1_7	D1_6	D1_5	D1_4	D1_3	D1_2	D1_1	D1_0	009H + (m × 20H)	undef.
M_DATAm2	D2_7	D2_6	D2_5	D2_4	D2_3	D2_2	D2_1	D2_0	00AH + (m × 20H)	undef.
M_DATAm3	D3_7	D3_6	D3_5	D3_4	D3_3	D3_2	D3_1	D3_0	00BH + (m × 20H)	undef.
M_DATAm4	D4_7	D4_6	D4_5	D4_4	D4_3	D4_2	D4_1	D4_0	00CH + (m × 20H)	undef.
M_DATAm5	D5_7	D5_6	D5_5	D5_4	D5_3	D5_2	D5_1	D5_0	00DH + (m × 20H)	undef.
M_DATAm6	D6_7	D6_6	D6_5	D6_4	D6_3	D6_2	D6_1	D6_0	00EH + (m × 20H)	undef.
M_DATAm7	D7_7	D7_6	D7_5	D7_4	D7_3	D7_2	D7_1	D7_0	00FH + (m × 20H)	undef.

Bit Position	Bit Name	Function
7 to 0 (M_DATAm0)	D0_7 to D0_0	Contents of the message data byte 0. (first message data byte)
7 to 0 (M_DATAm1)	D1_7 to D1_0	Contents of the message data byte 1.
7 to 0 (M_DATAm2)	D2_7 to D2_0	Contents of the message data byte 2.
7 to 0 (M_DATAm3)	D3_7 to D3_0	Contents of the message data byte 3.
7 to 0 (M_DATAm4)	D4_7 to D4_0	Contents of the message data byte 4.
7 to 0 (M_DATAm5)	D5_7 to D5_0	Contents of the message data byte 5.
7 to 0 (M_DATAm6)	D6_7 to D6_0	Contents of the message data byte 6.
7 to 0 (M_DATAm7)	D7_7 to D7_0	Contents of the message data byte 7.

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Remark: m = 00 to 63

Cautions:

1. **When transmitting data, only the number of bytes defined by the data length code (DLC) in the M_DLCm register are transmitted on the CAN bus. The transmission always starts with M_DATAm0.**
2. **If the ATS flag of the corresponding CxCTRL register is set (1) and the data length code (DLC) in the M_DLCm register is greater or equal 2, the last two bytes which are normally taken from the data part of the message buffer are ignored, and instead of these bytes a time stamp value is sent. (x = 1 to 3) (refer to chapter 13.2.5)**
3. **When a new message is received, all data bytes are updated, even if the data length code (DLC) in the M_DLCm register is less than 8. The values of the data bytes that have not been received may be change undefined.**

(6) Message data length code registers 00 to 63 (M_DLC0 to M_DLC63)

The M_DLCm registers specify the data length code (DLC) of the corresponding message m (m = 00 to 63). The DLC determines how many data bytes have to be transmitted, or received respectively, for the corresponding data frame.

These registers can be read/written in 8-bit units.

Figure 13-29: Message Data Length Code Registers 00 to 63 (M_DLC00 to M_DLC63)

	7	6	5	4	3	2	1	0	Address Offset[Note 1]	Initial value
M_DLCm	RFU[Note 2]	RFU[Note 2]	RFU[Note 2]	RFU[Note 2]	DLC3	DLC2	DLC1	DLC0	004H + (m × 20H)	undef.

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>3 to 0</td><td>DLC3 to DLC0</td><td>Specifies the data length code of the transmit/receive message.
<table>
<tr><th>DLC3</th><th>DLC2</th><th>DLC1</th><th>DLC0</th><th>Data Length Code (DLC)</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>No data bytes (0)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1 data byte</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>2 data bytes</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>3 data bytes</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>4 data bytes</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>5 data bytes</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>6 data bytes</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>7 data bytes</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>8 data bytes</td></tr>
<tr><td colspan="4">Others than above</td><td>Setting not recommended[Note 3]</td></tr>
</table>
</td></tr>
</table>

Notes:
1. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset
2. RFU = Reserved for future use. Ensure to set these bits to 0 when writing to the M_DLCm register.
3. If a DLC is specified to a value greater 8 for a transmit message, 8-byte transfer is performed regardless of the DLC value.

Remark: m = 00 to 63

(7) Message control registers 00 to 63 (M_CTRL0 to M_CTRL63)

The M_CTRLm registers control the behaviour on reception or transmission of the corresponding message buffer m (m = 00 to 63).
These registers can be read/written in 8-bit units.

Figure 13-30: Message Control Registers 00 to 63 (M_CTRL00 to M_CTRL63) (1/2)

	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
M_CTRLm	RMDE1	RMED0	ATS	IE	MOVR	R1	R0	RTR	005H + (m × 20H)	undef.

Bit Position	Bit Name	Function
7	RMED1	Specifies the remote frame handling mode 1. 0: DN flag is not changed, when receiving a remote frame. 1: DN flag is set (1), when receiving a remote frame. **Remark:** The remote frame handling mode 1 is only valid for transmit messages and indicates how the DN flag is updated if a remote frame is received on that message buffer. (For details refer to chapter **13.2.8 Remote frame handling**)
6	RMED0	Specifies the remote frame handling mode 0. 0: Auto answering of remote frame is not active. 1: Auto answering of remote frame is active. **Remark:** The remote frame handling mode 0 is only valid for transmit messages and indicates how to respond if a remote frame is received on that message buffer. (For details refer to chapter **13.2.8 Remote frame handling**)
5	ATS	Controls appending of the time stamp. 0: No time stamp appending. 1: Append time stamp **Remark:** This bit is only handled for transmit messages. If ATS is set (1) and the data length code (DLC) is greater or equal 2, the last two data bytes are replaced by the 16-bit time stamp. The appended time stamp is the capture value of the CAN global time system counter (CGTSC) on the SOF for this message. The last two data bytes defined in the data area are ignored. (For further details refer to chapter **13.2.5 Time stamp**)

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Remark: m = 00 to 63

Figure 13-30: Message Control Registers 00 to 63 (M_CTRL00 to M_CTRL63) (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>4</td><td>IE</td><td>Enables message buffer m related interrupts.
0: Interrupts related to message buffer m disabled.
1: Interrupts related to message buffer m enabled.

<b>Remark:</b> If the message related interrupt is enabled, an interrupt is generated for any of the following conditions:
<table>
<tr><th>Condition</th><th>Related Interrupt</th></tr>
<tr><td>Data frame or remote frame is transmitted from transmit message buffer.</td><td>CANxTRX</td></tr>
<tr><td>Data frame or remote frame is received on receive message buffer.</td><td rowspan="2">CANxREC</td></tr>
<tr><td>Remote frame is received on transmit message without auto answering set (RMDE0 = 0).</td></tr>
</table>
An interrupt is not generated, even if enabled, for any of the following conditions:
<table>
<tr><th>Condition</th></tr>
<tr><td>Remote frame is received on a transmit message with auto answering mode (RMDE0 = 1).</td></tr>
<tr><td>Remote frame is transmitted from receive message buffer.</td></tr>
</table>
</td></tr>
<tr><td>3</td><td>MOVR</td><td>Indicates a message buffer overwrite.
0: No overwriting occurred.
1: Message buffer contents have been overwritten at least once since the DN flag of the M_STATm register has been cleared (0).

<b>Remark:</b> If the OVM bit of the CxCTRL register is cleared (0) a message buffer linked to this CAN module might be overwritten by new messages although the DN flag is already set. Checking the MOVR bit additionally, indicates whether the message buffer has been overwritten.</td></tr>
<tr><td>2</td><td>R1</td><td>Reserved bit (value of CAN bus bit r0 for receive message buffer)</td></tr>
<tr><td>1</td><td>R0</td><td>Reserved bit (value of CAN bus bit r1 for receive message buffer)</td></tr>
<tr><td>0</td><td>RTR</td><td>Specifies remote or data frame type of the message buffer.
0: Message received or to be sent is a data frame
1: Message received or to be sent is a remote frame.

<b>Remark:</b> When the RTR bit is set (1) for a transmit message, a remote frame is transmitted for the given identifier instead of a data frame. The RTR bit can be read for a receive message to determine whether a data frame or a remote frame was received.</td></tr>
</table>

Remarks: **1.** m = 00 to 63
2. x = 1 to 3

(8) Message time stamp registers 00 to 63 (M_TIME00 to M_TIME63)

The M_TIMEm registers store the captured time stamp value on reception of the corresponding message m (m = 00 to 63).
These registers can be read/written in 16-bit units.

Figure 13-31: Message Time Stamp Registers 00 to 63 (M_TIME00 to M_TIME63)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
M_TIMEm	TS15	TS14	TS13	TS12	TS11	TS19	TS9	TS8	TS7	TS6	TS5	TS4	TS3	TS2	TS1	TS0	006H + (m x 20H)	undef.

Bit Position	Bit Name	Function
15 to 0	TS15 to TS0	16-bit time stamp value captured on message reception. **Remark:** The trigger for the time stamp capture event is selected by the TMR flag of the CxCTRL register. (For details refer to chapter **13.2.5 Time stamp**)

Note: The address of a message time stamp register is calculated according to the following formula: effective address = PP_BASE + address offset

Remarks: **1.** m = 00 to 63
2. x = 1 to 3

(9) Message event registers m0, m1, and m3 (M_EVTm0, M_EVTm1, M_EVTm3) (m = 00 to 63)

The message event registers M_EVTm0, MEVTm1, and M_EVTm3 imply the event pointers for event processing with the CAN bridge ELISA (m = 0 to 63).
These register can be read/written in 8-bit units.

Figure 13-32: Message Event Registers m0, m1, and m3 (M_EVTm0, M_EVTm1, M_EVm3) (m = 00 to 63)

	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
M_EVTm0	PTR07	PTR06	PTR05	PTR04	PTR03	PTR02	PTR01	PTR00	000H + (m × 20H)	undef.
M_EVTm1	PTR17	PTR16	PTR15	PTR14	PTR13	PTR12	PTR11	PTR10	001H + (m × 20H)	undef.
M_EVTm3	PTR37	PTR36	PTR35	PTR34	PTR33	PTR32	PTR31	PTR30	003H + (m × 20H)	undef.

Bit Position	Bit Name	Function
7 to 0 (M_EVTm0)	PTR07 to PTR00	8-bit event pointer 0 for processing with the CAN bridge ELISA. CAN bridge ELISA uses this pointer when receiving or sending a data frame in the message buffer m or from the message buffer m.
7 to 0 (M_EVTm1)	PTR17 to PTR10	8-bit event pointer 1 for processing with the CAN bridge ELISA. CAN bridge ELISA uses this pointer when receiving or sending a remote frame in the message buffer m or from the message buffer m, if the message buffer is defined as transmit message buffer.
7 to 0 (M_EVTm3)	PTR37 to PTR30	8-bit event pointer 3 for processing with the CAN bridge ELISA. CAN bridge ELISA uses this pointer for the error recovery mechanism while processing events launched in conjunction with message buffer m.

Remarks: **1.** m = 00 to 63
2. For further details refer to chapter **13.5 CAN Bridge ELISA**

13.3.5 CAN Module Registers

(1) CAN 1 to 3 mask 0 to 3 registers L, H (CxMASKL0 to CxMASKL3, CxMASKH0 to CxMASKH3) (x = 1 to 3)

The CxMASKL0 to CxMASKL3, and CxMASKH0 to CxMASKH3 registers specify the four acceptance masks for each CAN module x (x = 1 to 3). (For more details refer to chapter **13.2.7 Mask handling**)
These registers can be read/written in 8-bit and 16-bit units.

Figure 13-33: CAN 1 to 3 Mask 0 to 3 Registers L, H (CxMASKL0 to CxMASKL3, CxMASKH0 to CxMASKH3) (x = 1 to 3)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset	Initial value
CxMASKHn	CMIDE	0	0	CMID28	CMID27	CMID26	CMID25	CMID24	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16	see Table 13-17	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
CxMASKLn	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0	see Table 13-17	undef.

Bit Position	Bit Name	Function
15 (CxMASKHn)	CMIDE	Sets the CAN module mask option for the identifier type of the receive message. 0: Check identifier type of a received message. 1: Do not check identifier type. **Remark:** When CMIDE is cleared (0), the specified identifier type (standard or extended) of the message buffer linked to this CAN mask register must match the identifier type of the received message, in order to accept it for that message buffer.
12 to 0 (CxMASKHn) 15 to 0 (CxMASKLn)	CMID28 to CMID0	Sets the CAN module mask option for the corresponding identifier bit (ID28 to ID0) of the receive message. 0: Check identifier bit of a received message. 1: Do not check identifier bit. **Remarks:** **1.** When CMIDn is cleared (0), the specified identifier bit of the message buffer linked to this CAN mask register must match the identifier bit of the received message, in order to accept it for that message buffer. **2.** When a receive message buffer is linked to a mask, always 29 bits of the specified identifier in the M_IDHm, M_IDLm registers of the message buffer are compared with the identifier of the received message, even if a standard format (11 identifier bits) is set. In case standard format identifier is selected (IDE = 0) the lower 18 bits in the M_IDm register contain a copy of data field bits, so that an address extensions by means of data field bits is possible. **3.** When a mask is exclusively intended for a standard format identifier the irrelevant mask bits CMID17 to CMID0 have to be set (1).

Remarks: **1.** n = 0 to 3 (mask number)
2. x = 1 to 3

Table 13-17: Address Offsets of the CAN 1 to 3 Mask Registers

Symbol[Note]	Address Offset[Note 2]		
	x = 1	x = 2	x = 3
CxMASKL0	840H	880H	8C0H
CxMASKH0	842H	882H	8C2H
CxMASKL1	844H	884H	8C4H
CxMASKH1	846H	886H	8C6H
CxMASKL2	848H	888H	8C8H
CxMASKH2	84AH	88AH	8CAH
CxMASKL3	84CH	88CH	8CCH
CxMASKH3	84EH	88EH	8CEH

Notes: **1.** CAN module number: x = 1 to 3
2. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

(2) CAN 1 to 3 control registers (C1CTRL to C3CTRL)

The CxCTRL registers control the operating modes and indicate the operating status of the corresponding CAN module x (x = 1 to 3).

These registers can be read in 8-bit and 16-bit units. It can be written in 16-bit units only. For setting and clearing certain bits a special set/clear method applies (refer to chapter 13.3.1).

Figure 13-34: CAN 1 to 3 Control Registers (C1CTRL to C3CTRL) (1/4)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1CTRL	TECS1	TECS0	RECS1	RECS0	BOFF	TSTAT	RSTAT	ISTAT	0	DLEVR	DLEVT	OVM	TMR	STOP	SLEEP	INIT	850H	0101H
C2CTRL	TECS1	TECS0	RECS1	RECS0	BOFF	TSTAT	RSTAT	ISTAT	0	DLEVR	DLEVT	OVM	TMR	STOP	SLEEP	INIT	890H	0101H
C3CTRL	TECS1	TECS0	RECS1	RECS0	BOFF	TSTAT	RSTAT	ISTAT	0	DLEVR	DLEVT	OVM	TMR	STOP	SLEEP	INIT	8D0H	0101H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
C1CTRL	0	ST_DLEVT	ST_DLEVT	ST_OVM	ST_TMR	ST_STOP	ST_SLEEP	ST_INIT	0	CL_DLEVR	CL_DLEVT	CL_OVM	CL_TMR	CL_STOP	CL_SLEEP	CL_INIT	850H
C2CTRL	0	ST_DLEVT	ST_DLEVT	ST_OVM	ST_TMR	ST_STOP	ST_SLEEP	ST_INIT	0	CL_DLEVR	CL_DLEVT	CL_OVM	CL_TMR	CL_STOP	CL_SLEEP	CL_INIT	890H
C3CTRL	0	ST_DLEVT	ST_DLEVT	ST_OVM	ST_TMR	ST_STOP	ST_SLEEP	ST_INIT	0	CL_DLEVR	CL_DLEVT	CL_OVM	CL_TMR	CL_STOP	CL_SLEEP	CL_INIT	8D0H

Read (1/3)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15, 14</td><td>TECS1,
TECS0</td><td>Indicates the transmission error counter status.
<table>
<tr><th>TECS1</th><th>TECS0</th><th>Transmission Error Counter Status</th></tr>
<tr><td>0</td><td>0</td><td>Transmission error counter below warning level (< 96)</td></tr>
<tr><td>0</td><td>1</td><td>Transmission error counter in warning level range (96 to 127)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved (not possible)</td></tr>
<tr><td>1</td><td>1</td><td>Transmission error counter above warning level (≥ 128)</td></tr>
</table></td></tr>
<tr><td>13, 12</td><td>RECS1,
RECS0</td><td>Indicates the reception error counter status.
<table>
<tr><th>RECS1</th><th>RECS0</th><th>Reception Error Counter Status</th></tr>
<tr><td>0</td><td>0</td><td>Reception error counter below warning level (< 96)</td></tr>
<tr><td>0</td><td>1</td><td>Reception error counter in warning level range (96 to 127)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved (not possible)</td></tr>
<tr><td>1</td><td>1</td><td>Reception error counter above warning level (≥ 128)</td></tr>
</table></td></tr>
</table>

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 13-34: CAN 1 to 3 Control Registers (C1CTRL to C3CTRL) (2/4)

Read (2/3)

11	BOFF	Indicates a bus-off status of the CAN module. 0: CAN module is not in bus-off state (transmission error counter < 256) 1: CAN module is in bus-off state (transmission error counter = 256)
10	TSTAT	Indicates the transmission status. 0: No transmission activity on the CAN bus. 1: Transmission activity on the CAN bus.
9	RSTAT	Indicates the reception status. 0: No reception activity on the CAN bus. 1: Reception activity on the CAN bus.
8	ISTAT	Indicates the initialisation mode. 0: CAN module is in normal operation mode. 1: CAN module is stopped and set into initialisation mode. **Remarks:** **1.** The ISTAT bit is set when the setting of the INIT bit is acknowledged by the CAN protocol layer. It is cleared automatically when the INIT bit is cleared. **2.** In initialisation mode the level of the corresponding CAN transmit output is recessive (logical high). **3.** Data manipulation of the CxSYNC and CxBRP registers is only possible during INIT state. **4.** In INIT state the transmission and reception error counters are cleared and any error status is reset.
6	DLEVR	Specifies the dominant level of the CAN receive input pin. 0: Low level at the receive input is interpreted as a dominant bit (0). 1: High level at the receive input is interpreted as a dominant bit (0). **Remark:** From software point of view a dominant bit is always a “0” value.
5	DLEVT	Specifies the dominant level of the CAN transmit output pin. 0: A dominant bit (0) results in a low level output. 1: A dominant bit (0) results in a high level output. **Remark:** From software point of view a dominant bit is always a “0” value.
4	OVM	Specifies the CAN message buffer overwrite mode. 0: A new CAN message overwrites a message buffer with DN flag set (1). 1: A new CAN message is discarded, if it would be stored in a message buffer with DN bit set (1). **Remark:** The OVM bit determines how to handle a receive message in case this message would overwrite the corresponding receive message buffer.
3	TMR	Specifies the time stamp mode for reception. 0: CGTSC counter is captured into the corresponding M_TIMEm register at SOF signal of the receive message. 1: CGTSC counter is captured into the corresponding M_TIMEm register, when the valid receive message is copied into the message buffer. **Remark:** For details refer to chapter **13.2.5 Time stamp**

Figure 13-34: CAN 1 to 3 Control Registers (C1CTRL to C3CTRL) (3/4)

Read (3/3)

2	STOP	Selects the CAN stop mode. 0: CAN module is not stop mode. 1: CAN module stop mode selected. **Remarks:** **1.** The CAN stop mode can be entered only if the CAN module is already in sleep mode (SLEEP = 1). **2.** In CAN stop mode the CAN module is disabled (protocol layer activities stopped, and set in suspend mode), and wake up of the CAN module is only possible by CPU (CPU clears STOP bit). **3.** Releasing the STOP mode enters the initialisation mode.
1	SLEEP	Selects the CAN sleep mode. 0: Normal operation mode. 1: CAN module sleep mode selected. **Remarks:** **1.** Entering the CAN sleep mode from normal operating mode is just possible when the CAN bus is idle. **2.** In CAN sleep mode the CAN module does not process any transmit request submitted by the CPU. **3.** In case there is activity on the CAN bus and in parallel the SLEEP bit is set (1), the CAN module remains in normal operating mode and the SLEEP bit is cleared (0) automatically. **4.** The CAN sleep mode is released and normal operating mode is entered under the following conditions: (a) CPU clears the SLEEP bit (i.e. internal wake up by CPU) (b) first dominant bit on the idle CAN bus (i.e. external wake up by CAN bus activity) **5.** After releasing the CAN sleep mode the WAKE bit of the CxDEF register is set (1), and an error interrupt is generated upon external wake up by CAN bus activity.
0	INIT	Requests entering the initialisation mode. 0: Normal operation mode 1: Initialisation mode request **Remark:** The INIT flag is used to set the CAN module in initialisation mode. The CAN module acknowledges the transition into initialisation state by setting the ISTAT flag (1).

Write (1/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>14, 6</td><td>ST_DLEVR,
CL_DLEVR</td><td>Sets/clears the DLEVR bit.
<table>
<tr><th>ST_DLEVR</th><th>CL_DLEVR</th><th>Status of DLEVR bit</th></tr>
<tr><td>0</td><td>1</td><td>DLEVR bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>DLEVR bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in DLEVR bit value.</td></tr>
</table>
</td></tr>
</table>

Figure 13-34: CAN 1 to 3 Control Registers (C1CTRL to C3CTRL) (4/4)

Write (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>13, 5</td><td>ST_DLEVT,
CL_DLEVT</td><td>Sets/clears the DLEVT bit.
<table>
<tr><th>ST_DLEVT</th><th>CL_DLEVT</th><th>Status of DLEVT bit</th></tr>
<tr><td>0</td><td>1</td><td>DLEVT bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>DLEVT bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in DLEVT bit value.</td></tr>
</table></td></tr>
<tr><td>13, 4</td><td>ST_OVM,
CL_OVM</td><td>Sets/clears the OVM bit.
<table>
<tr><th>ST_OVM</th><th>CL_OVM</th><th>Status of OVM bit</th></tr>
<tr><td>0</td><td>1</td><td>OVM bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>OVM bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in OVM bit value.</td></tr>
</table></td></tr>
<tr><td>12, 3</td><td>ST_TMR,
CL_TMR</td><td>Sets/clears the TMR bit.
<table>
<tr><th>ST_TMR</th><th>CL_TMR</th><th>Status of TMR bit</th></tr>
<tr><td>0</td><td>1</td><td>TMR bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>TMR bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in TMR bit value.</td></tr>
</table></td></tr>
<tr><td>11, 2</td><td>ST_STOP,
CL_STOP</td><td>Sets/clears the STOP bit.
<table>
<tr><th>ST_STOP</th><th>CL_STOP</th><th>Status of STOP bit</th></tr>
<tr><td>0</td><td>1</td><td>STOP bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>STOP bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in STOP bit value.</td></tr>
</table></td></tr>
<tr><td>10, 1</td><td>ST_SLEEP,
CL_SLEEP</td><td>Sets/clears the SLEEP bit.
<table>
<tr><th>ST_SLEEP</th><th>CL_SLEEP</th><th>Status of SLEEP bit</th></tr>
<tr><td>0</td><td>1</td><td>SLEEP bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>SLEEP bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in SLEEP bit value.</td></tr>
</table></td></tr>
<tr><td>9, 0</td><td>ST_INIT,
CL_INIT</td><td>Sets/clears the INIT bit.
<table>
<tr><th>ST_INIT</th><th>CL_INIT</th><th>Status of INIT bit</th></tr>
<tr><td>0</td><td>1</td><td>INIT bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>INIT bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in INIT bit value.</td></tr>
</table></td></tr>
</table>

(3) CAN 1 to 3 definition registers (C1DEF to C3DEF)

The CxDEF registers define normal and diagnostic operation and indicate CAN bus error and states of the corresponding CAN module x (x = 1 to 3).

These registers can be read in 8-bit and 16-bit units. It can be written in 16-bit units only. For setting and clearing certain bits a special set/clear method applies (refer to chapter 13.3.1).

Figure 13-35: CAN 1 to 3 Definition Registers (C1DEF to C3DEF) (1/3)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1DEF	0	0	0	0	0	0	0	0	DGM	MOM	SSHT	PBB	BERR	VALID	WAKE	OVR	852H	0101H
C2DEF	0	0	0	0	0	0	0	0	DGM	MOM	SSHT	PBB	BERR	VALID	WAKE	OVR	892H	0101H
C3DEF	0	0	0	0	0	0	0	0	DGM	MOM	SSHT	PBB	BERR	VALID	WAKE	OVR	8D2H	0101H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
C1DEF	ST_ DGM	ST_ MOM	ST_ SSHT	ST_ PBB	0	0	0	0	CL_ DGM	CL_ MOM	CL_ SSHT	CL_ PBB	CL_ BERR	CL_ VALID	CL_ WAKE	CL_ OVR	852H
C2DEF	ST_ DGM	ST_ MOM	ST_ SSHT	ST_ PBB	0	0	0	0	CL_ DGM	CL_ MOM	CL_ SSHT	CL_ PBB	CL_ BERR	CL_ VALID	CL_ WAKE	CL_ OVR	892H
C3DEF	ST_ DGM	ST_ MOM	ST_ SSHT	ST_ PBB	0	0	0	0	CL_ DGM	CL_ MOM	CL_ SSHT	CL_ PBB	CL_ BERR	CL_ VALID	CL_ WAKE	CL_ OVR	8D2H

Read (1/2)

Bit Position	Bit Name	Function
7	DGM	Specifies the storage of receive message in diagnostic mode. 0: receive only and store valid message in message buffer type 7 1: receive only and store valid message as in normal operation mode **Remark:** The settings of the DGM bit are only effective in diagnostic mode (MOM = 1). In normal operation mode (MOM = 0) the DGM bit settings have no meaning.
6	MOM	Defines the module operating mode. 0: Normal operating mode 1: Diagnostic mode **Remarks:** **1.** The diagnostic mode provides the following functional behavior: (a) Transmission of data frames and remote frames is not possible. (b) No acknowledge is generated upon reception of a valid message. (c) On reception of a valid message the VALID flag is set (0). (d) Receive and transmit error counters remain unchanged on errors. **2.** The diagnostic mode can be used for baud rate detection and diagnostic purposes. **Caution:** **When the diagnostic mode (MOM = 1) is defined for a CAN module, the CxBRP register is only accessible in the initialisation state (ISTAT = 1). While ISTAT is cleared (0) write access to the CxBRP is prohibited and reading the address of the CxBRP register returns the status of the CxDINF register.**

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 13-35: CAN 1 to 3 Definition Registers (C1DEF to C3DEF) (2/3)

Read (2/2)

Bit Position	Bit Name	Function
5	SSHT	Defines the single-shot mode for a CAN module. 0: Normal operating mode 1: Single-shot mode **Remarks:** **1.** In single shot mode the CAN module tries to transmit a message only once, and the TRQ flag of the corresponding message is cleared regardless whether the transmission was successful (no error occurred), or not. **2.** In case of an error frame caused during a transmission in single-shot mode, the CAN module does not launch a re-transmission. However, error management according to the CAN Protocol is executed (i.e. generation of error interrupt, incrementing of error counters). **3.** The CPU can switch between the normal operating mode and the single-shot mode while the CAN module is active without causing any error on the CAN bus. **Caution:** **According to the CAN protocol upon a loss of arbitration a transmitter attempts to re-transmit the message, though loss of arbitration is not defined as an error.** **When single shot mode is set (SSHT = 1), a loss of arbitration is signaled by setting the BERR flag (1). Since the BERR flag signals a bus error in normal operation, the user must check it in conjunction with the values of the error counter, in order to judge whether it was caused by an error or a loss of arbitration.**
4	PBB	Defines the priority by message buffer numbers. 0: Transmission priority is given by message identifier. 1: Transmission priority is given by the number of the message buffer. **Remark:** Normally the message identifier defines the transmission priority. If the PBB flag is set, the location of a message defines the priority – the lower the message buffer number the higher the transmission priority.
3	BERR	Indicates a CAN bus error. 0: No CAN bus error occurred since the bit was cleared last. 1: At least one CAN bus error occurred since the flag has been cleared last. **Remark:** For single shot mode (SSHT bit = 1) this flag indicates a loss of the arbitration.
2	VALID	Indicates valid protocol activity. 0: No valid message was detected by the CAN protocol layer. 1: At least one valid message was received on the CAN bus since the flag has been cleared last.
1	WAKE	Indicates the wake-up condition from CAN sleep mode. 0: No wake-up, or sleep mode has been terminated by CPU (normal operation). 1: CAN sleep mode has been terminated by detection of CAN bus activity.
0	OVR	Indicates an overrun error. 0: No overrun (normal operation) 1: An overrun occurred during access to the CAN memory. **Remark:** The OVR flag is set, if the CAN message handler is not able to scan all the message areas defined for the CAN module due to timing problems. The error interrupt CxINT6 is generated at the same time. Possible cause for an overrun situation: The CAN memory access clock f_{MEM} selection is too slow for the selected CAN baud rate.

Figure 13-35: CAN 1 to 3 Definition Registers (C1DEF to C3DEF) (3/3)

Write

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15, 7</td><td>ST_DGM,
CL_DGM</td><td>Sets/clears the DGM bit.
<table>
<tr><th>ST_DGM</th><th>CL_DGM</th><th>Status of DGM bit</th></tr>
<tr><td>0</td><td>1</td><td>DGM bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>DGM bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in DGM bit value.</td></tr>
</table></td></tr>
<tr><td>14, 6</td><td>ST_MOM,
CL_MOM</td><td>Sets/clears the MOM bit.
<table>
<tr><th>ST_MOM</th><th>CL_MOM</th><th>Status of MOM bit</th></tr>
<tr><td>0</td><td>1</td><td>MOM bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>MOM bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in MOM bit value.</td></tr>
</table></td></tr>
<tr><td>13, 5</td><td>ST_SSHT,
CL_SSHT</td><td>Sets/clears the SSHT bit.
<table>
<tr><th>ST_SSHT</th><th>CL_SSHT</th><th>Status of SSHT bit</th></tr>
<tr><td>0</td><td>1</td><td>SSHT bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>SSHT bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in SSHT bit value.</td></tr>
</table></td></tr>
<tr><td>12, 4</td><td>ST_PPB,
CL_PPB</td><td>Sets/clears the PPB bit.
<table>
<tr><th>ST_PPB</th><th>CL_PPB</th><th>Status of PPB bit</th></tr>
<tr><td>0</td><td>1</td><td>PPB bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>PPB bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in PPB bit value.</td></tr>
</table></td></tr>
<tr><td>3</td><td>CL_BERR</td><td>Clears the BERR bit.
0: No change of BERR bit.
1: BERR bit is cleared (0).</td></tr>
<tr><td>2</td><td>CL_VALID</td><td>Clears the VALID bit.
0: No change of VALID bit.
1: VALID bit is cleared (0).</td></tr>
<tr><td>1</td><td>CL_WAKE</td><td>Clears the WAKE bit.
0: No change of WAKE bit.
1: WAKE bit is cleared (0).</td></tr>
<tr><td>0</td><td>CL_OVR</td><td>Clears the OVR bit.
0: No change of OVR bit.
1: OVR bit is cleared (0).</td></tr>
</table>

(4) CAN 1 to 3 information registers (C1LAST to C3LAST)

The CxLAST registers return the number of the last received message and last CAN protocol error of the corresponding CAN module x (x = 1 to 3).
These registers can be read-only in 8-bit and 16-bit units.

Figure 13-36: CAN 1 to 3 Information Registers (C1LAST to C3LAST)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note 1]	Initial value
C1LAST	0	0	0	0	LERR3	LERR2	LERR1	LERR0	LREC7	LREC6	LREC5	LREC4	LREC3	LREC2	LREC1	LREC0	854H	00FFH
C2LAST	0	0	0	0	LERR3	LERR2	LERR1	LERR0	LREC7	LREC6	LREC5	LREC4	LREC3	LREC2	LREC1	LREC0	894H	00FFH
C3LAST	0	0	0	0	LERR3	LERR2	LERR1	LERR0	LREC7	LREC6	LREC5	LREC4	LREC3	LREC2	LREC1	LREC0	8D4H	00FFH

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>11 to 8</td><td>LERR3 to LERR0</td><td>Holds the code of the last CAN protocol error.
<table>
<tr><th>LERR3</th><th>LERR2</th><th>LERR1</th><th>LERR0</th><th>Code of Last CAN Protocol Error</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>No error (reset state only)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>CAN bus bit error</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>CAN bus stuff error</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>CAN bus CRC error</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>CAN bus form error</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>CAN bus acknowledgement error</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>CAN bus arbitration lost [Note 2]</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>CAN module overrun error</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>CAN wake-up from CAN bus</td></tr>
<tr><td colspan="4">Others than above</td><td>Reserved</td></tr>
</table>
<b>Remark:</b> The LERR3 to LERR0 bits cannot be cleared. Thus these bits remain unchanged until the next error occurs.</td></tr>
<tr><td>7 to 0</td><td>LREC7 to LREC0</td><td>Holds the message buffer number of the last received message.
<table>
<tr><th>LREC7 to LREC0</th><th>Receive Message Buffer Number</th></tr>
<tr><td>0 to 63</td><td>Message buffer number of the last received message</td></tr>
<tr><td>64 to 254</td><td>Reserved (not possible)</td></tr>
<tr><td>255</td><td>No message has been received</td></tr>
</table></td></tr>
</table>

Notes:
1. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset
2. This error code only occurs in single-shot mode (SSHT bit of the CxDEF register = 1).

(5) CAN 1 to 3 error counter registers (C1ERC to C3ERC)

The CxERC registers reflect the status of the transmit and the receive error counters of the corresponding CAN module x (x = 1 to 3).
These registers can be read-only in 8-bit and 16-bit units.

Figure 13-37: CAN 1 to 3 Error Counter Registers (C1ERC to C3ERC)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1ERC	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	856H	00FFH
C2ERC	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	896H	00FFH
C3ERC	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	8D6H	00FFH

Bit Position	Bit Name	Function
15 to 8	REC7 to REC0	The receive error counter (REC) holds the status of the error counter of reception errors as defined in the CAN protocol.
7 to 0	TEC7 to TEC0	The transmit error counter (TEC) holds the status of the error counter of transmission errors as defined in the CAN protocol.

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

(6) CAN 1 to 3 interrupt enable registers (C1IE to C3IE)

The CxIE registers enable the transmit, receive and error interrupts of the corresponding CAN module x (x = 1 to 3).
These registers can be read in 8-bit and 16-bit units. It can be written in 16-bit units only. For setting and clearing certain bits a special set/clear method applies (refer to chapter 13.3.1).

Figure 13-38: CAN 1 to 3 Interrupt Enable Registers (C1IE to C3IE) (1/2)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1IE	0	0	0	0	0	0	0	0	0	E_INT6	E_INT5	E_INT4	E_INT3	E_INT2	E_INT1	E_INT0	858H	0000H
C2IE	0	0	0	0	0	0	0	0	0	E_INT6	E_INT5	E_INT4	E_INT3	E_INT2	E_INT1	E_INT0	898H	0000H
C3IE	0	0	0	0	0	0	0	0	0	E_INT6	E_INT5	E_INT4	E_INT3	E_INT2	E_INT1	E_INT0	8D8H	0000H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
C1IE	0	ST_E_INT6	ST_E_INT5	ST_E_INT4	ST_E_INT3	ST_E_INT2	ST_E_INT1	ST_E_INT0	0	CL_E_INT6	CL_E_INT5	CL_E_INT4	CL_E_INT3	CL_E_INT2	CL_E_INT1	CL_E_INT0	858H
C2IE	0	ST_E_INT6	ST_E_INT5	ST_E_INT4	ST_E_INT3	ST_E_INT2	ST_E_INT1	ST_E_INT0	0	CL_E_INT6	CL_E_INT5	CL_E_INT4	CL_E_INT3	CL_E_INT2	CL_E_INT1	CL_E_INT0	898H
C3IE	0	ST_E_INT6	ST_E_INT5	ST_E_INT4	ST_E_INT3	ST_E_INT2	ST_E_INT1	ST_E_INT0	0	CL_E_INT6	CL_E_INT5	CL_E_INT4	CL_E_INT3	CL_E_INT2	CL_E_INT1	CL_E_INT0	8D8H

Read

Bit Position	Bit Name	Function
6	E_INT6	Enables CAN module error interrupt (INT6). 0: Interrupt disabled 1: Interrupt enabled
5	E_INT5	Enables CAN bus error interrupt (INT5). 0: Interrupt disabled 1: Interrupt enabled
4	E_INT4	Enables wake-up from CAN sleep mode interrupt (INT4). 0: Interrupt disabled 1: Interrupt enabled
3	E_INT3	Enables interrupt for error passive on reception(INT3). 0: Interrupt disabled 1: Interrupt enabled
2	E_INT2	Enables interrupt for error passive or bus-off on transmission (INT2). 0: Interrupt disabled 1: Interrupt enabled
1	E_INT1	Enables reception completion interrupt (INT1). 0: Interrupt disabled 1: Interrupt enabled
0	E_INT0	Enables transmission completion interrupt (INT0). 0: Interrupt disabled 1: Interrupt enabled

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 13-38: CAN 1 to 3 Interrupt Enable Registers (C1IE to C3IE)(2/2)

Write

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>14, 6</td><td>ST_E_INT6,
CL_E_INT6</td><td>Sets/clears the E_INT6 bit.
<table>
<tr><th>ST_E_INT6</th><th>CL_E_INT6</th><th>Status of E_INT6 bit</th></tr>
<tr><td>0</td><td>1</td><td>E_INT6 bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>E_INT6 bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in E_INT6 bit value.</td></tr>
</table></td></tr>
<tr><td>13, 5</td><td>ST_E_INT5,
CL_E_INT5</td><td>Sets/clears the E_INT5 bit.
<table>
<tr><th>ST_E_INT5</th><th>CL_E_INT5</th><th>Status of E_INT5 bit</th></tr>
<tr><td>0</td><td>1</td><td>E_INT5 bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>E_INT5 bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in E_INT5 bit value.</td></tr>
</table></td></tr>
<tr><td>12, 4</td><td>ST_E_INT4,
CL_E_INT4</td><td>Sets/clears the E_INT4 bit.
<table>
<tr><th>ST_E_INT4</th><th>CL_E_INT4</th><th>Status of E_INT4 bit</th></tr>
<tr><td>0</td><td>1</td><td>E_INT4 bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>E_INT4 bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in E_INT4 bit value.</td></tr>
</table></td></tr>
<tr><td>11, 3</td><td>ST_E_INT3,
CL_E_INT3</td><td>Sets/clears the E_INT3 bit.
<table>
<tr><th>ST_E_INT3</th><th>CL_E_INT3</th><th>Status of E_INT3 bit</th></tr>
<tr><td>0</td><td>1</td><td>E_INT3 bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>E_INT3 bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in E_INT3 bit value.</td></tr>
</table></td></tr>
<tr><td>10, 2</td><td>ST_E_INT2,
CL_E_INT2</td><td>Sets/clears the E_INT2 bit.
<table>
<tr><th>ST_E_INT2</th><th>CL_E_INT2</th><th>Status of E_INT2 bit</th></tr>
<tr><td>0</td><td>1</td><td>E_INT2 bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>E_INT2 bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in E_INT2 bit value.</td></tr>
</table></td></tr>
<tr><td>9, 1</td><td>ST_E_INT1,
CL_E_INT1</td><td>Sets/clears the E_INT1 bit.
<table>
<tr><th>ST_E_INT1</th><th>CL_E_INT1</th><th>Status of E_INT1 bit</th></tr>
<tr><td>0</td><td>1</td><td>E_INT1 bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>E_INT1 bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in E_INT1 bit value.</td></tr>
</table></td></tr>
<tr><td>8, 0</td><td>ST_E_INT0,
CL_E_INT0</td><td>Sets/clears the E_INT0 bit.
<table>
<tr><th>ST_E_INT0</th><th>CL_E_INT0</th><th>Status of E_INT0 bit</th></tr>
<tr><td>0</td><td>1</td><td>E_INT0 bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>E_INT0 bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in E_INT0 bit value.</td></tr>
</table></td></tr>
</table>

(7) CAN 1 to 3 bus activity registers (C1BA to C3BA)

The CxBA registers indicate the status of the CAN bus activities of the corresponding CAN module x (x = 1 to 3).
These registers can be read-only in 8-bit and 16-bit units.

Figure 13-39: CAN 1 to 3 Bus Activity Registers (C1BA to C3BA) (1/2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1BA	0	0	0	CACT4	CACT3	CACT2	CACT1	CACT0	TMNO7	TMNO6	TMNO5	TMNO4	TMNO3	TMNO2	TMNO1	TMNO0	85AH	00FFH
C2BA	0	0	0	CACT4	CACT3	CACT2	CACT1	CACT0	TMNO7	TMNO6	TMNO5	TMNO4	TMNO3	TMNO2	TMNO1	TMNO0	89AH	00FFH
C3BA	0	0	0	CACT4	CACT3	CACT2	CACT1	CACT0	TMNO7	TMNO6	TMNO5	TMNO4	TMNO3	TMNO2	TMNO1	TMNO0	8DAH	00FFH

Bit Position	Bit Name	Function
12 to 8	CACT4 to CACT0	Indicates the CAN module activity.

CACT4	CACT3	CACT2	CACT1	CACT0	CAN Module Activity
0	0	0	0	0	Reset state
0	0	0	0	1	Waiting for bus idle
0	0	0	1	0	Bus idle
0	0	0	1	1	Start of frame (SOF)
0	0	1	0	0	Standard format ID section
0	0	1	0	1	Data length code section
0	0	1	1	0	Data field section
0	0	1	1	1	CRC field section
0	1	0	0	0	CRC delimiter
0	1	0	0	1	Acknowledge slot
0	1	0	1	0	Acknowledge delimiter
0	1	0	1	1	End of frame section (EOF)
0	1	1	0	0	Intermission state
0	1	1	0	1	Suspend transmission
0	1	1	1	0	Error frame
0	1	1	1	1	Waiting for error delimiter
1	0	0	0	0	Error delimiter
1	0	0	0	1	Error bus-off
1	0	0	1	0	Extended format ID section
Others than above					Reserved

Remark: The CACT4 to CACT0 bits reflect the status of the CAN protocol layer.

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 13-39: CAN 1 to 3 Bus Activity Registers (C1BA to C3BA) (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>7 to 0</td><td>TMNO7 to TMNO0</td><td>Indicates the message buffer, which is either waiting to be transmitted or in transmission progress.
<table>
<tr><th>TMNO7 to TMNO0</th><th>Number of Transmit Message Buffer</th></tr>
<tr><td>0 to 63</td><td>Current transmit message buffer (waiting for transmission, or in transmission progress)</td></tr>
<tr><td>64 to 254</td><td>Reserved (not possible)</td></tr>
<tr><td>255</td><td>No message waiting for transmission, or in transmission progress.</td></tr>
</table>
</td></tr>
</table>

(8) CAN 1 to 3 bit rate prescaler registers (C1BRP to C3BRP)

The CxBRP registers specify the bit rate prescaler and CAN bus speed of the corresponding CAN module x (x = 1 to 3). The register layout depends on the TLM bit (bit 15), and distinguishes between 6-bit prescaler (TLM bit = 0) and 8-bit prescaler (TLM bit = 1).
These registers can be read/written in 8-bit and 16-bit units. However, write access is only permitted in initialisation mode (ISTAT bit of the CxCTRL register = 1)

Figure 13-40: CAN 1 to 3 Bit Rate Prescaler Registers (C1BRP to C3BRP) (1/2)

TLM = 0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1BRP	TLM	0	0	0	0	0	0	0	0	BTYPE	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	85CH	0000H
C2BRP	TLM	0	0	0	0	0	0	0	0	BTYPE	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	89CH	0000H
C3BRP	TLM	0	0	0	0	0	0	0	0	BTYPE	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	8DCH	0000H

TLM = 1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1BRP	TLM	0	0	0	0	0	0	BTYPE	BRP7	BRP6	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	85CH	
C2BRP	TLM	0	0	0	0	0	0	BTYPE	BRP7	BRP6	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	89CH	
C3BRP	TLM	0	0	0	0	0	0	BTYPE	BRP7	BRP6	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	8DCH	

Bit Position	Bit Name	Function
15	TLM	Specifies the transfer layer mode. 0: Transfer layer uses 6-bit prescaler setting. 1: Transfer layer uses 8-bit prescaler setting.
8 (TLM = 1) 6 (TLM = 0)	BTYPE	Specifies the CAN bus type. 0: CAN bus type is low speed bus (≤ 125 kbps) 1: CAN bus type is high speed bus (> 125 kbps)

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Remarks: **1.** Writing to this register is only possible if CAN module is set into initialisation mode.
2. CPU can read CxBRP register at any time.

Caution: In diagnostic mode the CxBRP register is hidden, and the CxDINF register appears instead of it at the same address.

Figure 13-40: CAN 1 to 3 Bit Rate Prescaler Registers (C1BRP to C3BRP) (2/2)

Bit Position	Bit Name	Function
7 to 0 (TLM = 1) 5 to 0 (TLM = 0)	BRP7 to BRP0 (TLM = 1) BRP5 to BRP0 (TLM = 0)	Specifies the bit rate prescaler for the CAN protocol layer.

TLM = 0:

BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Bit Rate Prescaler $f_{BTL} = f_{MEM} / 2\times(k+1)$	k
0	0	0	0	0	0	$f_{BTL} = f_{MEM} / 2$	0
1	0	0	0	0	1	$f_{BTL} = f_{MEM} / 4$	1
0	0	0	0	1	0	$f_{BTL} = f_{MEM} / 6$	2
1	0	0	0	1	1	$f_{BTL} = f_{MEM} / 8$	3
0	0	0	1	0	0	$f_{BTL} = f_{MEM} / 10$	4
⋮						⋮	⋮
1	1	1	1	1	0	$f_{BTL} = f_{MEM} / 126$	62
1	1	1	1	1	1	$f_{BTL} = f_{MEM} / 128$	63

TLM = 1:

BRP7	BRP6	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Bit Rate Prescaler $f_{BTL} = f_{MEM} / (k+1)$	k
0	0	0	0	0	0	0	0	$f_{BTL} = f_{MEM}$	0
0	0	1	0	0	0	0	1	$f_{BTL} = f_{MEM} / 2$	1
0	1	0	0	0	0	1	0	$f_{BTL} = f_{MEM} / 3$	2
0	1	1	0	0	0	1	1	$f_{BTL} = f_{MEM} / 4$	3
1	0	0	0	0	1	0	0	$f_{BTL} = f_{MEM} / 5$	4
⋮								⋮	⋮
1	1	1	1	1	1	1	0	$f_{BTL} = f_{MEM} / 255$	254
1	1	1	1	1	1	1	1	$f_{BTL} = f_{MEM} / 256$	255

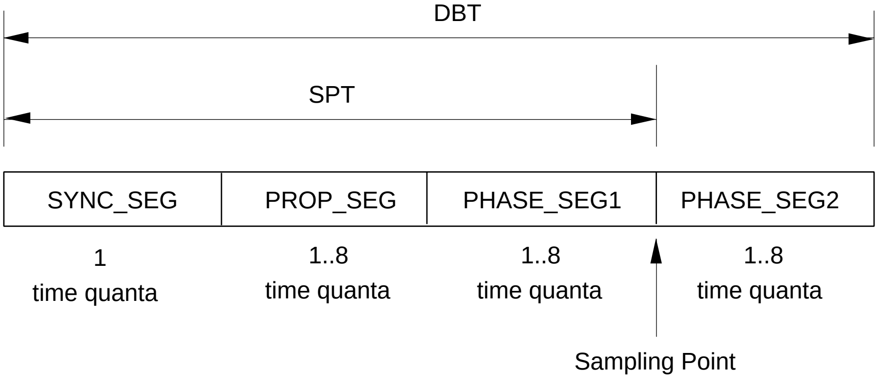
Remark: The BRP defines the period of the base clock f_{BTL} for the protocol layer of a CAN module. It determines the number of FCAN system clocks f_{MEM} per time quantum TQ. A time quantum TQ is the basic unit of a bit in a CAN frame:

$$TQ = \frac{1}{f_{BTL}}$$

(9) CAN 1 to 3 synchronization control registers (C1SYNC to C3SYNC)

A bit in a CAN frame is built by a programmable number of time quanta (TQ), as shown in the Figure 13-41 below.

Figure 13-41: CAN Bus Bit Timing



For the CAN modules in the FCAN system the bit length of segments SYNC_SEG, PROP_SEG, PHASE_SEG1 and PHASE_SEG2 must not be defined explicitly. All necessary CAN bit timings are programmed by defining

- the total number of time quanta TQ per CAN bit (i.e. DBT).
- the location of the sample point (i.e. SPT) as a number of TQ.

The CAN protocol segmentation is done by the CAN module automatically.

Due to re-synchronisation mechanisms the CAN module may lengthen PHASE_SEG1 or shorten PHASE_SEG2 by one or more TQ. The total number of TQ for which the CAN module is permitted to lengthen or shorten the phase segments is called synchronisation jump width (SJW). The SJW value must be less or equal the difference of DBT and SPT, which corresponds to PHASE_SEG2, and can be specified in the range of 1 TQ to 4 TQ.

For additional information on the CAN bus bit timing please refer to ISO 11898.

The relation between CAN memory clock and CAN bus baud rate is:

$$f_{CANBUS} = \frac{1}{DBT \times TQ} = \frac{f_{BTL}}{DBT} = \frac{f_{MEM}}{DBT \times BRP}$$

Valid values for DBT and BRP are:

TLM bit	DBT [TQ]	BRP[Note]	
0	8, 9, 10, ... ,25	2, 4, 6, ... ,128	2 × (k + 1)
1	8, 9, 10, ... ,25	1, 2, 3, ... ,256	k + 1

Note: BRP is the resulting bit rate prescaler value specified in the CxBRP register, where the variable k corresponds to the contents of bits BRP5 to BRP0 when TLM bit = 0, and bits BRP7 to BRP0 bits when TLM bit = 1, respectively (x = 1 to 3).

The CxSYNC registers specify the data bit time (DBT), sampling point position (SPT) and synchronisation jump width (SJW) of the corresponding CAN module x (x = 1 to 3).
These registers can be read/written in 8-bit and 16-bit units. However, write access is only permitted in initialisation mode (ISTAT bit of the CxCTRL register = 1)

Figure 13-42: CAN 1 to 3 Synchronization Control Registers (C1SYNC to C3SYNC) (1/2))

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1SYNC	0	0	0	SAMP	SJW1	SJW0	SPT4	SPT3	SPT2	SPT1	SPT0	DBT4	DBT3	DBT2	DBT1	DBT0	85EH	0218H
C2SYNC	0	0	0	SAMP	SJW1	SJW0	SPT4	SPT3	SPT2	SPT1	SPT0	DBT4	DBT3	DBT2	DBT1	DBT0	89EH	0218H
C3SYNC	0	0	0	SAMP	SJW1	SJW0	SPT4	SPT3	SPT2	SPT1	SPT0	DBT4	DBT3	DBT2	DBT1	DBT0	8DEH	0218H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>12</td><td>SAMP</td><td>Specifies the bit sampling.
0: Sample receive data one time at sampling point.
1: Sample receive data three times and take majority decision at sampling point.</td></tr>
<tr><td>11, 10</td><td>SJW1,
SJW0</td><td>Specifies the synchronization jump width.
<table>
<tr><th>SJW1</th><th>SJW0</th><th>Synchronization Jump Width</th></tr>
<tr><td>0</td><td>0</td><td>1 TQ</td></tr>
<tr><td>0</td><td>1</td><td>2 TQ</td></tr>
<tr><td>1</td><td>0</td><td>3 TQ</td></tr>
<tr><td>1</td><td>1</td><td>4 TQ</td></tr>
</table></td></tr>
<tr><td>9 to 5</td><td>SPT4 to
SPT0</td><td>Specifies the sampling point position.
<table>
<tr><th>SPT4</th><th>SPT3</th><th>SPT2</th><th>SPT1</th><th>SPT0</th><th>Sampling Point Position
SPT = (p + 1) TQ</th><th>p</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td rowspan="2">Setting prohibited</td><td rowspan="2"></td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>3 TQ</td><td>2</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>4 TQ</td><td>3</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>5 TQ</td><td>4</td></tr>
<tr><td colspan="5">.
.
.</td><td>.
.
.</td><td>.
.
.</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>17 TQ</td><td>16</td></tr>
<tr><td colspan="5">Other than above</td><td>Setting prohibited</td><td></td></tr>
</table></td></tr>
</table>

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 13-42: CAN 1 to 3 Synchronization Control Registers (C1SYNC to C3SYNC) (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>4 to 0</td><td>DBT4 to DBT0</td><td>Specifies the number of TQ per bit.
<table>
<tr><th>DBT4</th><th>DBT3</th><th>DBT2</th><th>DBT1</th><th>DBT0</th><th>Data Bit Time
DBT = (q + 1) TQ</th><th>q</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td rowspan="3">Setting prohibited</td><td rowspan="3">–</td></tr>
<tr><td colspan="5">.
.
.</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>8 TQ</td><td>7</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>9 TQ</td><td>8</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>10 TQ</td><td>9</td></tr>
<tr><td colspan="5">.
.</td><td>.
.
.</td><td>.
.</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>25 TQ</td><td>24</td></tr>
<tr><td colspan="5">Other than above</td><td>Setting prohibited</td><td></td></tr>
</table>
</td></tr>
</table>

Remarks: **1.** CPU can read the CxSYNC register at any time (x = 1 to 3).
2. Writing to the register is only possible when the CAN module is set to INIT mode.
3. For setting the DBT and SPT bits some rules must be observed, otherwise the CAN module will malfunction (for details refer to chapter 13.4).

(10) CAN 1 to 3 bus diagnostic information registers (C1DINF to C3DINF)

The CxDINF registers reflect the last transmission on CAN bus of the corresponding CAN module x (x = 1 to 3).

These registers can be read-only in 1-bit, 8-bit and 16-bit units. It is only accessible when diagnostic mode is set (CxDEF register's MOM bit = 1).

Figure 13-43: CAN 1 to 3 Bus Diagnostic Information Registers (C1DINF to C3DINF)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1DINF	DINF15	DINF14	DINF13	DINF12	DINF11	DINF10	DINF9	DINF8	DINF7	DINF6	DINF5	DINF4	DINF3	DINF2	DINF1	DINF0	85DH	0000H
C2DINF	DINF15	DINF14	DINF13	DINF12	DINF11	DINF10	DINF9	DINF8	DINF7	DINF6	DINF5	DINF4	DINF3	DINF2	DINF1	DINF0	89DH	0000H
C3DINF	DINF15	DINF14	DINF13	DINF12	DINF11	DINF10	DINF9	DINF8	DINF7	DINF6	DINF5	DINF4	DINF3	DINF2	DINF1	DINF0	8DDH	0000H

Bit Position	Bit Name	Function
15 to 8	DINF15 to DINF8	Number of bits detected on the CAN bus since the last occurrence of a SOF signal.
7 to 0	DINF7 to DINF0	Reflects the value of the last 8 bits transmitted on the CAN bus, where DINF0 holds the very last bit.

Remarks: **1.** The CAN bus diagnostic information shows all CAN bus bits including stuff bits, delimiters, etc.

2. The storage of the last 8 bits is automatically stopped either when detecting an error on the CAN bus or when detecting a valid message (acknowledge delimiter). It is automatically reset whenever a SOF is detected on the CAN bus.

Caution: **In normal operating mode the CxDINF register is hidden, and the corresponding CxBRP register appears instead of it at the same address.**

13.4 Operating Condsiderations

13.4.1 Rules to be observed for correct baud rate settings

Observing the following rules for the baud rate setting assures correct operation of a CAN module and compliance to the CAN protocol specification.

(1) Rule for sampling point (SPT) setting:

The sample point position needs to be programmed between 3 TQ and 17 TQ, which corresponds to the SPT4 to SPT0 bits of the CxSYNC register (x = 1 to 3):

$$3\ \mathrm{TQ} \leq \mathrm{SPT} = (p+1)\ \mathrm{TQ} \leq 17\ \mathrm{TQ}$$

$$2 \leq p \leq 16$$

$$p = \text{decimal value of bits SPT4 to SPT0}$$

(2) Rule for data bit time (DBT) setting:

The number of TQ per CAN frame bit is restricted to a range from 8 TQ to 25 TQ, which corresponds to the DBT4 to DBT0 bits of the CxSYNC register:

$$8\ \mathrm{TQ} \leq \mathrm{DBT} = (q+1)\ \mathrm{TQ} \leq 25\ \mathrm{TQ}$$

$$7 \leq q \leq 24$$

$$q = \text{decimal value of bits DBT4 to DBT0}$$

(3) Rule for synchronization jump width (SJW) setting:

The number of TQ allowed for soft-synchronization must not exceed the number of TQ for PHASE_SEG2. The length of PHASE_SEG2 is given by the difference of data bit time (DBT) and the sampling point position (SPT). Converted to register values the following condition applies:

$$\mathrm{SJW} = (s+1)\ \mathrm{TQ} \leq \mathrm{DBT} - \mathrm{SPT}$$

$$s \leq q - p - 1$$

$$s = \text{decimal value of bits SJW1, SJW0}$$

Remark: The time quantum (TQ) is determined by the base clock f_{BTL} for the CAN protocol layer, which is defined in the CxBRP register:

$$\mathrm{TQ} = \frac{1}{f_{BTL}}$$

Caution: The rules above represent CAN protocol limits. Violating these rules may cause erroneous operation.

13.4.2 Example for baudrate setting of CAN module

To illustrate how to calculate the correct setting of the registers CxBRP and CxSYNC the following example is given to 3:

Requirements from CAN bus:

- FCAN system global frequency f_{MEM} = 16 MHz
- CAN bus baud rate $f_{CANBUS} = (83^1/_3)$ kHz
- Sampling point 75% or above
- Synchronization jump width 3 TQ

First the frequency ratio between the global frequency and the CAN bus baud rate is calculated:

$$\frac{f_{MEM}}{f_{CANBUS}} = \frac{16\ \text{MHz}}{(83^1/_3)\ \text{KHz}} = 192 = 3 * 2^6$$

The register descriptions show that the prescaler must be an even number between 2 and 128, the data bit time must be a value in the range 8 to 25.

As the synchronization jump width (SJW) is defined as 3 TQ, the maximum setting for the sampling point (SPT) can be only 3 TQ less than the setting for the data bit time (DBT) and also less than 17 TQ:

$$SPT \le \min\{DBT - 1, 17\}$$

Based on the restrictions and assumptions above, the four settings are basically possible:

Prescaler (BRP)	Data Bit Time (DBT)	Max. Sampling Point (SPT)	Calculated Sampling Point
24	8 TQ	5 TQ	5/8 = 62.5%
16	12 TQ	9 TQ	9/12 = 75%
12	16 TQ	13 TQ	13/16 = 81.25%
8	24 TQ	17 TQ	17/24 = 71%

Regarding the maximum sampling point setting and the resulting sampling point, two settings meet all the requirements above. Therefore the correct settings are:

(1) TLM=0:

BRP5 to BRP0 =	000101B	(5)	(prescaler BRP = 12 TQ)
DBT4 to DBT0 =	01111B	(15)	(data bit time DBT = 16 TQ)
SPT4 to SPT0 =	01100B	(12)	(sampling point SPT = 13 TQ)

or

BRP5 to BRP0 =	000111B	(7)	(prescaler BRP = 16 TQ)
DBT4 to DBT0 =	01011B	(11)	(data bit time DBT = 12 TQ)
SPT4 t oSPT0 =	01000B	(8)	(sampling point SPT = 9 TQ)

(2) TLM=1:

BRP7 to BRP0 =	00001011B	(11)	(prescaler BRP = 12)
DBT4 to DBT0 =	01111B	(15)	(data bit time DBT = 16 TQ)
SPT4 to SPT0 =	01100B	(12)	(sampling point SPT = 13 TQ)

or

BRP7 to BRP0 =	00001111B	(15)	(prescaler BRP = 16)
DBT4 to DBT0 =	01011B	(11)	(data bit time DBT = 12 TQ))
SPT4 t oSPT0 =	01000B	(8)	(sampling point SPT = 9 TQ)

13.4.3 Ensuring data consistency

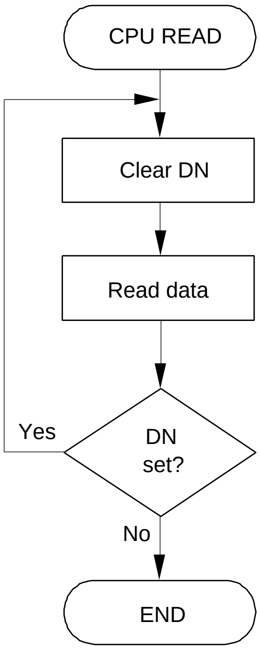
If the CPU reads data from the CAN message buffers, the consistency of data read has to be ensured. Therefore two mechanisms are provided:

- Sequential data read
- Burst mode data read

(1) Sequential data read

If the data is read by the CPU by sequential accesses to the CAN message buffers, the following sequence has to be observed:

Figure 13-44: Sequential CAN Data Read by CPU



As the DN flag is only set by the CAN module and cleared by the CPU only, it is ensured that the CPU can recognize that new data is stored in the message buffer during the read operation.

Remark: If the CPU reads the data by only one read access, the data consistency is always ensured. Therefore the check of the DN flag after reading the data is not necessary.

(2) Burst Mode Data Read

For faster access to a complete message the burst read mode is applicable.

In burst read mode the complete message is copied from the internal message buffer to a temporary read buffer located outside the CAN memory section. This allows read access without any wait, if the CAN memory is accessed by the CAN modules while the CPU tries to read data.

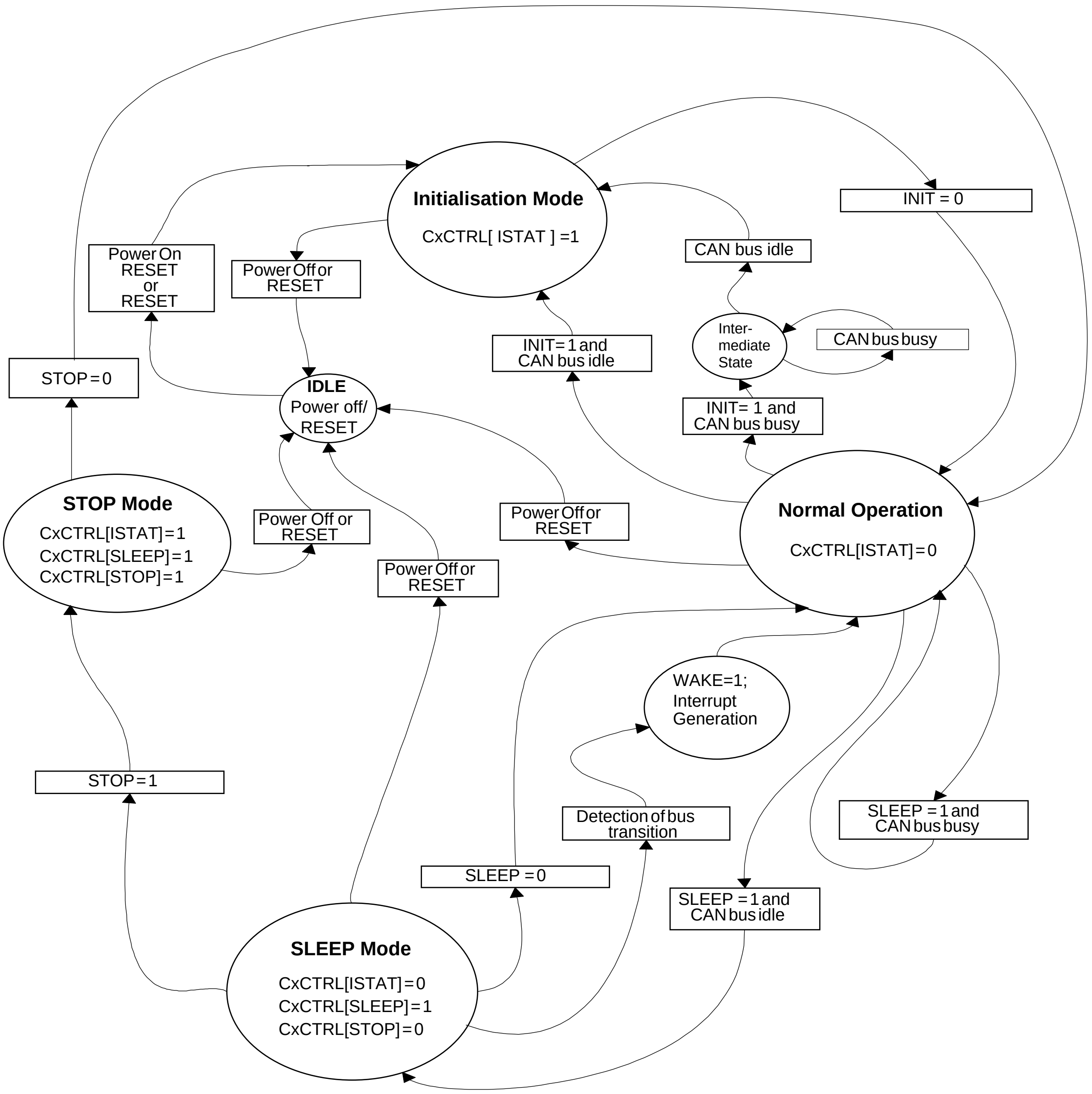
The copy of the message is automatically started whenever the data length code from the M_DLCm register is read by the CPU, and the data is copied from the message buffer into the temporary buffer. As long as the CPU reads 16-bit data from consecutive addresses (that means 16 -bit burst read sequence M_DLCm/M_CTRLm → M_TIMEm → M_DATAm0/m1 → M_DATAm2/m3 → M_DATAm4/m5 → M_DATAm6/m7 → M_IDLm → M_IDHm) the data is read from the temporary buffer.

Caution: The burst read requires consecutive 16-bit read accesses to the memory area. Any 8-bit access (byte read operation), even if not violating the linear address rule, causes that the read is performed from the register instead of the temporary buffer.

13.4.4 Operating states of the CAN modules

The different operating states and the state transitions of the CAN modules are shown in the state transition diagram in Figure 13-45.

Figure 13-45: State Transition Diagram for CAN Modules



Remark: x = 1 to 3

13.4.5 Initialisation routines

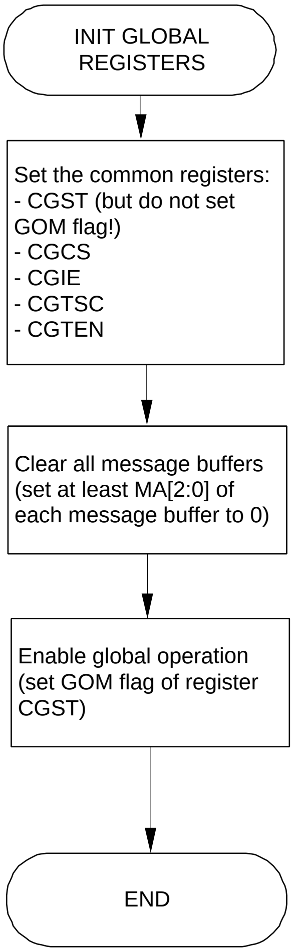
Below the necessary steps for correct start-up of the CAN interface are explained.

Caution: It is very important that the software programmer observes the sequence given in the following paragraphs. Otherwise unexpected operation of the CAN interface or any CAN module can occur.

(1) Global initialisation sequence for the CAN interface

Before any operation on the CAN memory can be done, it is essential that the common control register are initialised. The general initialisation sequence is shown in Figure 13-46.

Figure 13-46: General Initialisation Sequence for the CAN Interface


INIT GLOBAL REGISTERS
Set the common registers:
- CGST (but do not set GOM flag!)
- CGCS
- CGIE
- CGTSC
- CGTEN
Clear all message buffers (set at least MA[2:0] of each message buffer to 0)
Enable global operation (set GOM flag of register CGST)
END

Remark: Enabling the global operation does not automatically enable any CAN module. Each CAN module must be initialised and enabled separately.

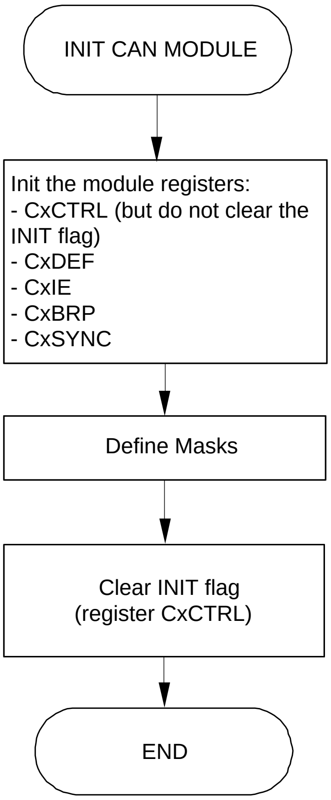
Example for C routine:

```
int CAN_GlobalInit (void)
{
        unsigned char i;

        CGST = 0x00FF;                      // clear all flags of CGST
        CGIE = 0x00FF;                      // disable global interrupts
        CGCS = 0x0000;                      // define internal clock
        CGTSC = 0x0000;                     // clear CAN global time system counter
        CGTEN = 0x0000;                     // disable all timer events

        for (i=0; i<CAN_MESSAGES; i++)
                CAN_ClearMessage(i);        // clear all message buffers

        CGST = 0x0100;                      // set GOM bit
        return 0;
}
```

(2) Initialisation sequence for a CAN Module

Each CAN module must be initialised by the sequence according to Figure 13-47.

Figure 13-47: Initialisation Sequence for a CAN module



Remark: x = 1 to 3

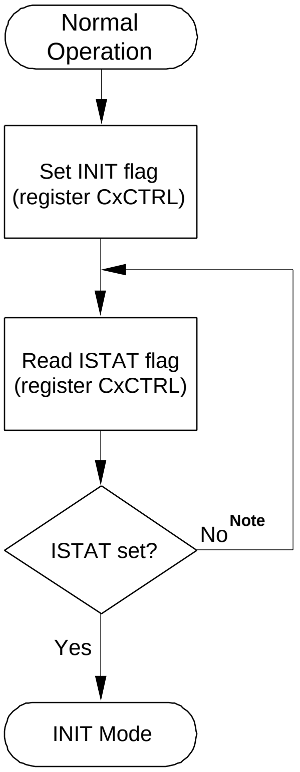
Example for C routine:

```
int CAN_ModuleInit (unsigned char module_no,
                    unsigned short brp_value,
                    unsigned short sync_value)
{
    can_module_type *can_mod_ptr;               // define ptr
    can_mod_ptr = &can_module[module_no];       // load ptr

    can_mod_ptr->CxCTRL = 0x00FE;               // clear CxCTRL
                                                // except INIT
    can_mod_ptr->CxDEF = 0x00FF;                // clear CxDEF
    can_mod_ptr->CxIE   = 0x00FF;               // clear CxIE
    can_mod_ptr->CxBRP = brp_value;             // set CxBRP
    can_mod_ptr->CxSYNC = sync_value;           // set CxSYNC
    can_mod_ptr->mask0_low = 0x0000;            // clear mask0
    can_mod_ptr->mask0_high = 0x0000;
    can_mod_ptr->mask1_low = 0x0000;            // clear mask1
    can_mod_ptr->mask1_high = 0x0000;
    can_mod_ptr->mask2_low = 0x0000;            // clear mask2
    can_mod_ptr->mask2_high = 0x0000;
    can_mod_ptr->mask3_low = 0x0000;            // clear mask3
    can_mod_ptr->CxCTRL = 0x0001;               // clear INIT flag

    return 0;
}
```

(3) Setting a CAN Module into initialisation state

The following routine is required if a CAN module has to be set from normal operation into initialisation mode.
Please notice that all CAN modules are automatically set to initialisation mode after reset. Therefore the sequence is only required if the CAN module is already in normal operation.

Figure 13-48: Setting CAN Module into Initialisation State


Normal Operation
Set INIT flag (register CxCTRL)
Read ISTAT flag (register CxCTRL)
ISTAT set?
No Note
Yes
INIT Mode

Note: In case of permanent bus activity the program loops for a long time. Therefore, a timeout mechanism should be provided in order to limit the runtime of the routine.

Remark: x = 1 to 3

Example for C routine:

```
int CAN_ModuleStop (unsigned char module_no)
{
    can_module_type *can_mod_ptr;               // define CAN module ptr
    can_mod_ptr = &can_module[module_no];       // load CAN module ptr

    if ((can_mod_ptr->CxCTRL & 0x0001)==0)      // if INIT flag not yet set:
        can_mod_ptr->CxCTRL=0x0100;             // set INIT flag

    while ((can_mod_ptr->CxCTRL & 0x0100)==0);  // wait until initialisation state is confirmed
                                                // (ISTAT bit = 1)
    return 0;
}
```

(4) Shutdown of the FCAN system

If the clock to the CAN interface should be switched off for power saving, the following sequence has to be executed for correct termination of any CAN bus activity:

<1> For each CAN module x (x = 1 to 3)
<a> Enter sleep mode
Set SLEEP bit = 1 (CxCTRL register)
or
<b> Enter initialisation mode
Set INIT bit = 1 (CxCTRL register) and wait for ISTAT bit = 1

<2> Disable event processing
Clear EVM flag (CGST register)

<3> Stop the CAN global time system counter
Clear the TSM flag (CGST register)

<4> Stop the global CAN operation
Clear GOM flag (CGST register)

<5> Switch off the CAN clock
Set CSTP bit (CSTOP register)

Caution: If the sequence is not observed, any active CAN module may cause malfunction on the corresponding CAN bus.

13.5 CAN Bridge ELISA

13.5.1 Principle of operation

ELISA is the name of an event processor, a sophisticated machine that supports the CPU on data transfer between different CAN buses and on time management functions.

The operations done by ELISA are based on user-programmable sequences of commands. The commands are stored in the CAN memory. The commands are numbered by their location in the CAN memory. The command with the lowest physical address is defined to be command 1, the next to be command 2, etc. ELISA executes commands sequentialy from lower command number to the higher command number.

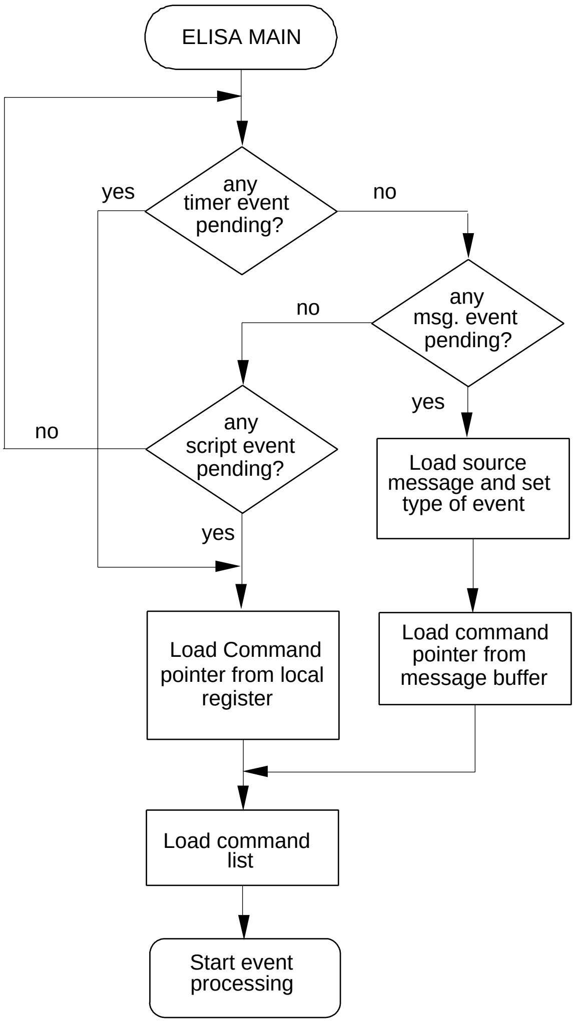
The event pointers (refer to chapter **13.3.4 (9)Message event registers m0, m1, and m3 (M_EVTm0, M_EVTm1, M_EVTm3) (m = 00 to 63)**) link a message object to the start of a command sequence to be executed if an event occurs. ELISA continuously checks the event request flag (ERQ) of all message buffers (M_STATm register, m = 00 to 63). Whenever the ERQ flag of a particular message buffer is set, ELISA loads the related event pointer and executes commands starting with the command that is addressed by the event pointer. ELISA will end its processing when executing a command with the END flag set.

Remark: In the V850E/CA1 (Atomic) an event command list executed by a certain event must only contain 8 event commands. In case the list contains more than 8 commands the CAN bridge ELISA stops execution after the 8th command automatically.

(1) Operation Flow Charts

For better understanding the following flow charts will explain how ELISA operates.

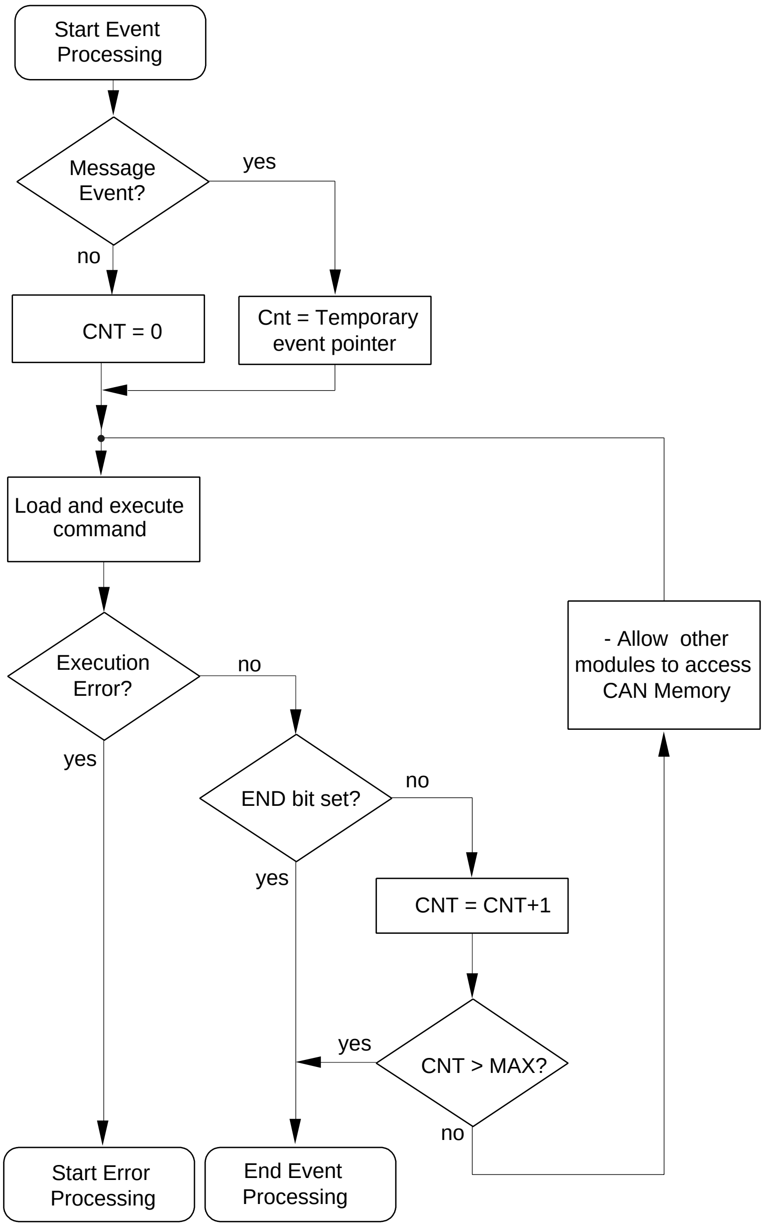
Figure 13-49: Main Operation of ELISA


ELISA MAIN
any timer event pending?
yes
no
any msg. event pending?
no
yes
any script event pending?
no
yes
Load source message and set type of event
Load Command pointer from local register
Load command pointer from message buffer
Load command list
Start event processing

The flowchart in Figure 13-49 illustrates the main loop of ELISA. The highest priority for event processing is given for events generated by the timer, followed by message events. The lower the message number the higher the priority of the message event. The lowest priority is given to the script event. The script event is only executed if no other event is pending.

If several timer events are waiting to be processed at the same time, the pending timer event with the lowest number is processed first.

Figure 13-50: Event Processing by ELISA

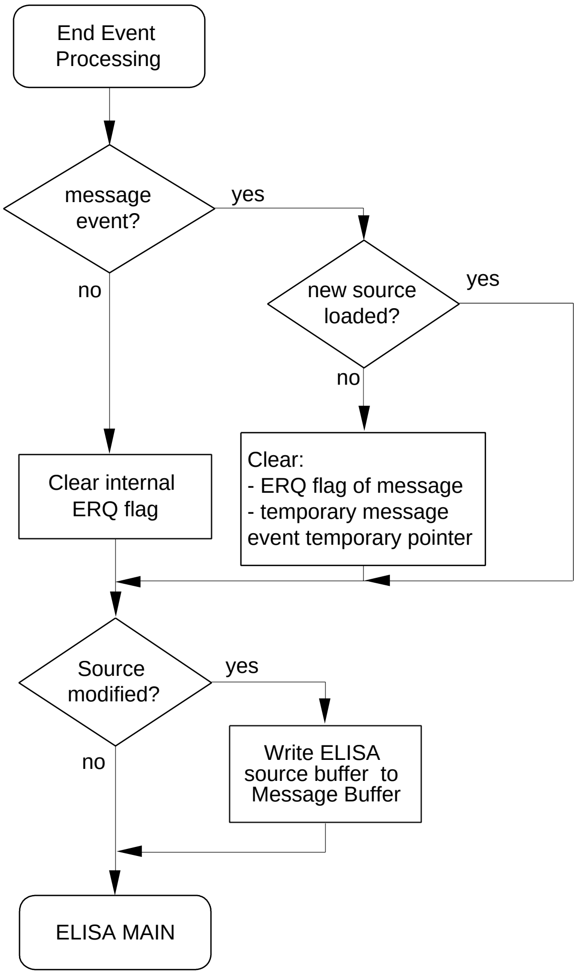


The flowchart in Figure 13-50 shows the event processing. Some implementations of ELISA limit the maximum number of commands, which can be executed within one event processing.

Similar to a microcomputer's program counter, ELISA has a temporary command pointer (M_EVTm3). The temporary command pointer stores the number of the command under execution. Contrary to a program counter, which holds an absolute address, the command number is an offset relative to the first command of the command list. The absolute number of the first command is defined by M_EVTm0. Under normal termination of a command list M_EVTm3 is cleared. In case of a command execution error, M_EVTm3 contains the number of the command that led to the erroneous execution.
At the beginning of processing a command list ELISA adds the value of M_EVTm3 to the value of M_EVTm0. That technique allows the user to design a convenient error recovery procedure.

For example: In an application where the full execution of a command list is required to assure data consistency, a command list once terminated can be continued at exactly the same command, where the error occurred. Details are described in the following example for event processing of ELISA.

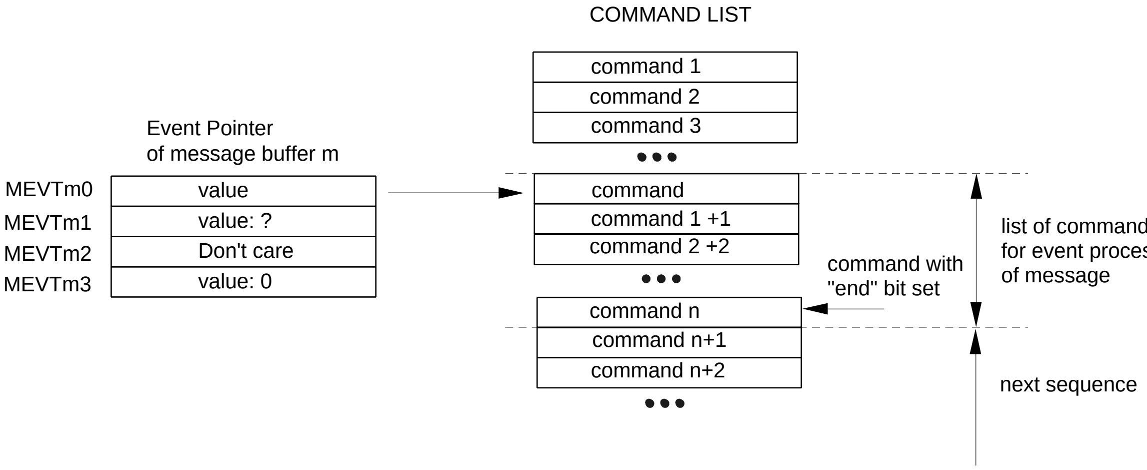
Figure 13-51: End of Event Processing



(2) Example

As an example for message event processing please look at the Figure 13-52 below. A message requires event processing of ELISA. Therefore a set of commands for that event processing is defined in the command list section, and the event pointer of the message contains the number of the first command ELISA should execute whenever the event processing for the message is requested.

Figure 13-52: Example for Event Processing of ELISA



When ELISA detects a request for event processing of the message (ERQ flag = 1), ELISA loads the event pointer of the message buffer first (in the figure above it is M_EVTm0). ELISA then adds the value of M_EVTm3 to determine the first command where it starts its execution. Under normal conditions M_EVTm3 is 0, so ELISA starts at the command given by the event pointer.

After execution of a command where an END flag is set, ELISA ends the processing by saving the data, writing 0 to M_EVTm3 and clearing the ERQ flag of the message.

M_EVTm3 is basically desired for error recovery, but it can also be used by the application for special conditions. Whenever an error occurs, ELISA does not write back a 0 to M_EVTm3, but the offset of the command that caused the error. If for example the second command (in the figure above it is command m+1) caused an error, ELISA writes a 1 to M_EVTm3.

By an Error Handling bit (EHDL bit in each command) the termination of an erroneous command can be determined. It is selectable that the event request flag ERQ is not cleared at an error.

When ELISA recognises a further event request and starts the event processing for the message again, it will not start its operation at the first command, but at the command which caused the error.

(3) Data consistency for CPU access

To ensure that ELISA does not overwrite or handle messages which are under use of the CPU, the CPU has to observe the following rule for modifying any message which is also handled by ELISA:

- clear RDY flag of message
- modify message
- set RDY flag of message

If ELISA tries to access a message with RDY flag = 0, an interrupt is generated and ELISA stops its operation. The CPU must do the error recovery then.

Cautions: 1. Any violation of this rule could cause data inconsistency-
2. Never modify by CPU any message with ERQ flag set (1) while ELISA is in operation (EER flag = 0)

(4) Syntax check of commands

There is no syntax check implemented in ELISA that takes care:

that temporary sums and counters are not mixed
that copy commands (especially bit-string copy) parameters are within the valid range

The used programming tool has to observe that the code generated for ELISA meets the specification of the ELISA commands. Any command not meeting the command requirements leads into malfunction of ELISA.

The programming tools provided by NEC take care that no commands with illegal parameters are generated.

13.5.2 Implementation details

The event pointers are used to indicate the start of a command sequence that should be executed whenever an event occurs. If ELISA detects an ERQ flag for a message buffer, it loads the related event pointer. ELISA will then execute commands starting with the command that number given by the event pointer. ELISA will end its processing when executing a command with the END flag set (1).

(1) General flags and buffers

(a) Condition flag (CFLG)

ELISA contains an internal condition flag (CFLG), which is used for conditional execution of commands. It is cleared whenever a new event script is processed. It is set when executing some special commands (e.g. decrement counter, test bit). For each command conditional execution can be selected. If conditional execution of a command is selected (by setting the COND flag in the command) the command is only executed if CFLG is set (1).
If no conditional execution is selected for a command, the execution of the command automatically clears the CFLG flag (0).

(b) 64-bit temporary buffer

For temporary storage of data by the event scripts (e.g. counters are required or if a temporary sum is calculated) a 64-bit temporary buffer is available within ELISA.

The 64-bit can either be used as 8-bit counters or as temporary 16-bit sums. The table below shows how the 64-bit temporary buffer can be split:

Table 13-18: Format of 64-bit Temporary Buffer

Bit7 to 0	Bit 15 to 8	Bit 23 to 16	Bit 31 to 24	Bit 39 to 32	Bit 47 to 40	Bit 55 to 48	Bit 63 to 56
Counter 0	Counter 1	Counter 2	Counter 3	Counter 4	Counter 5	Counter 6	Counter 7
Temporary sum 0		Temporary sum 1		Temporary sum 2		Temporary sum 3	
Low byte	High byte	Low byte	High byte	Low byte	High byte	Low byte	High byte

The counters can be set or decremented by certain commands. If a decrement command is executed on a counter already set to 0, nothing happens. If a decrement command is executed on a counter not equal to 0, the counter will be decremented and the new value stored. If the value becomes zero, the CFLG flag is set, otherwise the CFLG flag is cleared.

The 16-bit temporary sums can be used to add up to 16 bytes by command. As the resulting sum of 16 bytes can be only a 12-bit value the upper 4 bits are used to show the number of bytes added to the buffer. If 16 bytes in total are added, the upper 4 bits become 0 again. The lower 12 bits are used to store the total sum.

Table 13-19: Format of Temporary Sum

Bit 15 to 12	Bit 11 to 8	Bit 7 to 4	Bit 3 to 0
No. of adds	Sum (12-bit)		
High byte		Low byte	

(2) Command format

Each action command consists of a 32-bit word of the following format.

Figure 13-53: ELISA Command Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
END	EHDL	COND	CMD4	CMD3	CMD2	CMD1	CMD0	DEST7	DEST6	DEST5	DEST4	DEST3	DEST2	DEST1	DEST0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAR15	PAR14	PAR13	PAR12	PAR11	PAR10	PAR9	PAR8	PAR7	PAR6	PAR5	PAR4	PAR3	PAR2	PAR1	PAR0

Bit Position	Bit Name	Function
31	END	End of action command list 0: Actual command is not the last command of action command list 1: Actual command is the last command of the action command list **Remarks:** **1.** The END flag indicates that the actual command is the last one of the command list. Please notice that each command is executed before the END flag is tested. Thus the command where the END flag is set is the last command which is executed. **2.** The END flag has no influence on the execution of the command, meaning that the command is executed in any case. It is used to indicate that no further commands need to be executed after finishing the execution of the actual command **3.** After executing 8 consecutive commands without any END flag set, the execution module automatically stops the execution. (**First implementations only**)
30	EHDL	Error Handling 0: On error store source message and command number back to message buffer 1: On error do not store anything back to message buffer **Remark:** If an error occurs during the event processing of ELISA the EHDL bit defines how ELISA stops its execution. If EHDL bit is cleared (0) ELISA writes the actual status of the source message buffer and the number of the command (as offset to the start pointer) to the source message buffer, but it does not clear the ERQ bit of the source message buffer. If EHDL bit is set (1) no writing to the source message buffer is done.
29	COND	Conditional execution 0: Always execute command and clear CFLG 1: Execute command only if CFLG is set **Remark:** If commands should only be executed if a special condition occurs (e.g. if a counter is decremented to zero or if a special bit is set) then the COND flag can be used.
28 to 24	CMD4 to CMD0	Command Available commands depending on the implementation.
23 to16	DEST7 to DEST0	Destination message buffer Number of message buffer the command uses as target (0 to 255).
15 to 0	PAR15 to PAR0	Command parameter Additional parameter(s) for the corresponding command. Format depends on implementation of command handler.

(3) Bit location definition for bit commands

If the location of a bit or a bit-string has to be defined for a command, the bit location is given as a 6-bit or 5-bit value (named in to following as “BIT_POS”) using the following formula. If a 6-bit value is given, the location is absolute without any offset for the byte location. If a 5-bit value is given an offset has to be added. The type of command defines the offset of the byte location, i.e. if the command targets the bytes 4-7 then the offset for the byte location is 4.

Formula for start byte and start bit:
byte: BIT_POS div 8 [+ byte offset for 5-bit value]
bit: BIT_POS modulo 8

Example 1:

BIT_POS is set to 43 (as 6-bit value)
→ BIT_POS div 8 = 43 div 8 = 5 … byte location
→ BIT_POS modulo 8 = 3 … bit location
⇒ bit/bit-string is located at data byte 5, bit 3

Example 2:

BIT_POS is set to 16 (as 5-bit value) for a command that targets bytes 4-7
→ byte offset = 4 (command targets bytes 4-7)
→ BIT_POS div 8 = 16 div 8 = 2 … relative byte location
→ BIT_POS modulo 8 = 0 … bit location
⇒ bit/bit-string is located at data byte 6, bit 0

Example 3:

BIT_POS is set to 5 (as 5-bit value) for a command that targets bytes 0-3
→ byte offset = 0 (command targets bytes 0-3)
→ BIT_POS div 8 = 5 div 8 = 0 … relative byte location
→ BIT_POS modulo 8 = 5 … bit location
⇒ bit/bit-string is located at data byte 0, bit 5

(4) ELISA specific events

(a) Timer event processing by ELISA

There are four timer events available (periodic intervals). Each of the timer events can be enabled or disabled individually. Whenever a timer event occurs, the event is reported to ELISA. ELISA will start the event processing similar to events given by messages, but the event pointers are read from pointer buffers located in ELISA.

(b) Script Events

For better flexibility a script event is available. The script event can be used if an event script requires more than 8 commands. Basically ELISA limits the total number of commands executed for one event to 8. If more commands are required the primary script can generate an additional event, so-called script event. When generating a script event the pointer on the first command must be defined at the same time.

The application software can also generate a script event.

(c) Data Consistency

As ELISA accesses the message buffers as well as the CPU and the CAN interfaces, some mechanisms were implemented to ensure that data inconsistency is detected.

To reduce the risk of data inconsistency the application software should not operate any message with the ERQ flag set without ensuring that ELISA is not handling that message. The CPU can ensure that by checking which event is under processing and clearing the ERQ flag if the message is not under processing.

ELISA stops its operation and generates and interrupt under the following conditions:

- any of the M_STATm register flags of the source message m handled by ELISA is written (DN, ERQ, TRQ, RDY) (m = 00 to 63)
- ELISA tries to write to a message with RDY flag cleared (0) or TRQ flag set (1)
- ELISA tries to read from a message with RDY flag cleared (0)

The EHDL bit defines how ELISA terminates its operation (writing back source message or not).

The application software must restart ELISA by writing to a special register.

(d) Debugging

For debugging the operation of ELISA the BREAK command can be used (refer to chapter 13.5.3).

13.5.3 Event processing commands

(1) Command Overview

Code	Description	Mnemonic
00H	no operation	NOP 0,0
01H	define and load new source	SET_SRC <DEST>,<PAR>
02H	set script event	SET_EVT <DEST>,0
03H	generate interrupt and stop operation	BREAK 0,0
04H to 07H	reserved for future implementations	
08H	write value to counter	SET_CNT <DEST>,<PAR>
09H	decrement counter and update CFLG flag	DECR_CNT <DEST>,0
0AH	test source bit and update CFLG flag	TST_BIT <DEST>,<PAR>
0BH	reserved for future implementations	
0CH	set and/or clear message flags	WR_FLG <DEST>,<PAR>
0DH	add value of source byte to temporary sum	ADD_SUM <DEST>,<PAR>
0EH	reserved for future implementations	
0FH		
10H	write source data to destination data bytes 0-3	WR_DLO <DEST>,<PAR>
11H	write source data to destination data bytes 4-7	WR_DHI <DEST>,<PAR>
12H	copy all source data to destination (including DLC)	WR_DALL <DEST>,<PAR>
13H	write source bit-string to destination bit-string, data bytes 0-3	WR_DBLO <DEST>,<PAR>
14H	write source bit-string to destination bit-string, data bytes 4-7	WR_DBHI <DEST>,<PAR>
15H to 17H	reserved for future implementations	
18H	get data bytes from destination data bytes 0-3	LD_DLO <DEST>,<PAR>
19H	get data bytes from destination data bytes 4-7	LD_DHI <DEST>,<PAR>
1AH	copy all destination data to source (including DLC)	LD_DALL <DEST>,<PAR>
1BH	get source bit-string from destination bit-string data bytes 0-3	LD_DBLO <DEST>,<PAR>
1CH	get source bit-string from destination bit-string data bytes 4-7	LD_DBHI <DEST>,<PAR>
1DH to 1FH	reserved for future implementations	

(a) No operation (NOP)

Command	No Operation
Code	00H
Mnemonic	NOP 0,0

(b) Define and load new source (SET_SRC)

Command	Define and Load New Source
Code	01H
Mnemonic	SET_SRC <DEST>,<PAR>

DEST7 to DEST0	Destination
0 to 255	Number of new source message

PAR0	Parameter
0	Do not handle old source.
1	Write old source to message buffer.

(c) Set script event (SET_EVT)

Command	Set Script Event
Code	02H
Mnemonic	SET_EVT <DEST>,0

DEST7 to DEST0	Destination
0 to 255	Pointer to first command for script event processing

(d) Generate interrupt and stop operation (BREAK)

Command	Generate Interrupt and Stop Operation
Code	03H
Mnemonic	BREAK 0,0
	Remark: If the command is executed ELISA generates an interrupt, handles the source message as defined by the EHDL flag and stops its operation. The application software has to restart ELISA.

(e) Set Counter (SET_CNT)

Command	Set Counter
Code	08H
Mnemonic	SET_CNT <DEST>,<PAR>

DEST2 to DEST0	Destination
0 to 7	Number of counter

PAR7 to PAR0	Parameter
0 to 255	Value written to counter

(f) Decrement counter and update CFLG flag (DECR_CNT)

<table>
<tr><th>Command</th><th>Decrement Counter and Update CFLG Flag</th></tr>
<tr><td>Code</td><td>09H</td></tr>
<tr><td>Mnemonic</td><td>DECR_CNT <DEST>,0</td></tr>
<tr><td></td><td>Effect of the DECR_CNT command is as follows.
<table>
<tr><th>Counter Value Before Executing Command</th><th>Counter Value After Executing Command</th><th>CFLG at End of Command</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>n (2 ≤ n ≤ 255)</td><td>n-1</td><td>0</td></tr>
</table>
</td></tr>
</table>

DEST2 to DEST0	Destination
0 to 7	number of counter

(g) Test Source Bit and Update CFLG Flag (TST_BIT)

Command	Decrement Counter and Update CFLG Flag
Code	0AH
Mnemonic	TST_BIT <DEST>,<PAR>
	Remark: The test bit number is set as described in **13.5.2 (3)Bit location definition for bit commands**

DEST5 to DEST0	Destination
0 to 63	Number of source data bit to be tested

<table>
<tr><th>PAR0</th><th>Parameter</th></tr>
<tr><td>0</td><td>Test for 0
<table>
<tr><th>Status of Test Bit</th><th>CFLG at End of Command</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table></td></tr>
<tr><td>1</td><td>Test for 1
<table>
<tr><th>Status of Test Bit</th><th>CFLG at End of Command</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table></td></tr>
</table>

(h) Set and/or clear message flags (WR_FLG)

Command	Set and/or Clear Message Flags (M_STATm Register)
Code	0CH
Mnemonic	WR_FLG <DEST>,<PAR>

DEST5 to DEST0	Destination
0 to 254	Destination message number (absolute)
255	Manipulate message flags (in M_STATm register) of source message

PAR15 to PAR8	Parameter (Upper Byte)
0 to 255	Bit-set parameter for M_STATm **Remark:** The parameter has the same format as if the CPU sets any bit of the M_STATm register.

PAR7 to PAR0	Parameter (Lower Byte)
0 to 255	Bit-clear parameter for M_STATm **Remark:** The parameter has the same format as if the CPU clears any bit of the M_STATm register.

(i) Add value of source byte to temporary sum (ADD_SUM)

Command	Add Value of Source Byte to Temporary Sum
Code	0DH
Mnemonic	ADD_SUM <DEST>,<PAR>

DEST2, DEST1	Destination
0 to 3	Number of temporary sum

PAR6 to PAR4	Parameter
0 to 7	Number of source byte to be added to temporary sum **Remark:** When the total numbers of adds are done, an interrupt is generated and ELISA stops its operation. The user can define by the EHDL flag how ELISA stops its operation.

PAR3 to PAR0	Parameter
0	Maximum 16 adds to temporary sum
1 to 15	Maximum number of adds to temporary sum

(j) Write source data to destination data bytes 0 to 3 (WR_DLO)

Command	Write Source Data to Destination Data Bytes 0 to 3
Code	10H
Mnemonic	WR_DLO <DEST>,<PAR>

DEST7 to DEST0	Destination
0 to 255	Number of message buffer the command uses as target

PAR15 to PAR12	Parameter for Destination Data Byte 0
0 to 7	Number of source data byte to be copied to destination data byte 0
8	Copy source time stamp byte 0 to destination data byte 0
9	Copy source time stamp byte 1 to destination data byte 0
10 to 14	Setting prohibited
15	Do not copy any data to destination data byte 0

PAR11 to PAR8	Parameter for Destination Data Byte 1
0 to 7	Number of source data byte to be copied to destination data byte 1
8	Copy source time stamp byte 0 to destination data byte 1
9	Copy source time stamp byte 1 to destination data byte 1
10 to 14	Setting prohibited
15	Do not copy any data to destination data byte 1

PAR7 to PAR4	Parameter for Destination Data Byte 2
0 to 7	Number of source data byte to be copied to destination data byte 2
8	Copy source time stamp byte 0 to destination data byte 2
9	Copy source time stamp byte 1 to destination data byte 2
10 to 14	Setting prohibited
15	Do not copy any data to destination data byte 2

PAR3 to PAR0	Parameter for Destination Data Byte 3
0 to 7	Number of source data byte to be copied to destination data byte 3
8	Copy source time stamp byte 0 to destination data byte 3
9	Copy source time stamp byte 1 to destination data byte 3
10 to 14	Setting prohibited
15	Do not copy any data to destination data byte 3

(k) Write source data to destination data bytes 4 to 7 (WR_DHI)

Command	Write Source Data to Destination Data Bytes 4 to 7
Code	11H
Mnemonic	WR_DHI <DEST>,<PAR>

DEST7 to DEST0	Destination
0 to 255	Number of message buffer the command uses as target

PAR15 to PAR12	Parameter for Destination Data Byte 4
0 to 7	Number of source data byte to be copied to destination data byte 4
8	Copy source time stamp byte 4 to destination data byte 4
9	Copy source time stamp byte 1 to destination data byte 4
10 to 14	Setting prohibited
15	Do not copy any data to destination data byte 4

PAR11 to PAR8	Parameter for Destination Data Byte 5
0 to 7	Number of source data byte to be copied to destination data byte 5
8	Copy source time stamp byte 0 to destination data byte 5
9	Copy source time stamp byte 5 to destination data byte 5
10 to 14	Setting prohibited
15	Do not copy any data to destination data byte 5

PAR7 to PAR4	Parameter for Destination Data Byte 6
0 to 7	Number of source data byte to be copied to destination data byte 6
8	Copy source time stamp byte 0 to destination data byte 6
9	Copy source time stamp byte 1 to destination data byte 6
10 to 14	Setting prohibited
15	Do not copy any data to destination data byte 6

PAR3 to PAR0	Parameter for Destination Data Byte 7
0 to 7	Number of source data byte to be copied to destination data byte 7
8	Copy source time stamp byte 0 to destination data byte 7
9	Copy source time stamp byte 1 to destination data byte 7
10 to 14	Setting prohibited
15	Do not copy any data to destination data byte 7

(l) Copy all source data to destination (WR_DALL)

Command	Copy All Source Data to Destination (Including DLC)
Code	12H
Mnemonic	WR_DALL <DEST>,<PAR>

DEST7 to DEST0	Destination
0 to 255	Number of message buffer the command uses as target

PAR0	Parameter
0	Copy all data except source identifier to destination
1	Copy all data including source identifier to destination

(m) Write Source Bit-string to Destination Bit-string, Data Bytes 0 to 3 (WR_DBLO)

Command	Write Source Data to Destination Bit-string, Data Bytes 0 to 3
Code	13H
Mnemonic	WR_DBLO <DEST>,<PAR>

DEST7 to DEST0	Destination
0 to 255	Number of message buffer the command uses as target

PAR14 to PAR10	Parameter for Start Position in Source
0 to 31	Start position (bit number) of bit-string in source

PAR9 to PAR5	Parameter for Bit-string Length
0 to 31	Length of bit-string in bits decremented by 1. **Remark:** A value of 0 means a length of 1, a value of 1 a length of 2, etc.

PAR4 to PAR0	Parameter for Start Position in Destination
0 to 7	Start position (bit number) of bit-string in destination (byte 0 to 3)

(n) Write source bit-string to destination bit-string, data bytes 4 to 7 (WR_DBHI)

Command	Write Source Data to Destination Bit-string, Data Bytes 4 to 7
Code	14H
Mnemonic	WR_DBHI <DEST>,<PAR>

DEST7 to DEST0	Destination
0 to 255	Number of message buffer the command uses as target

PAR14 to PAR10	Parameter for Start Position in Source
0 to 31	Start position (bit number) of bit-string in source

PAR9 to PAR5	Parameter for Bit-string Length
0 to 31	Length of bit-string in bits decremented by 1. **Remark:** A value of 0 means a length of 1, a value of 1 a length of 2, etc.

PAR4 to PAR0	Parameter for Start Position in Destination
0 to 7	Start position (bit number) of bit-string in destination (byte 4 to 7)

(o) Get data bytes from destination data bytes 0 to 3 (LD_DLO)

Command	Get Data Bytes from Destination Data Bytes 0 to 3
Code	18H
Mnemonic	LD_DLOI <DEST>,<PAR>

DEST7 to DEST0	Destination
0 to 255	Number of message buffer the command uses as target

PAR15 to PAR12	Parameter for Destination Data Byte 0
0 to 7	Number of destination data byte to be copied to source data byte 0
8 to 14	Setting prohibited
15	Do not copy the destination data byte

PAR11 to PAR8	Parameter for Destination Data Byte 1
0 to 7	Number of destination data byte to be copied to source data byte 1
8 to 14	Setting prohibited
15	Do not copy the destination data byte

PAR7 to PAR4	Parameter for Destination Data Byte 2
0 to 7	Number of destination data byte to be copied to source data byte 2
8 to 14	Setting prohibited
15	Do not copy the destination data byte

PAR3 to PAR0	Parameter for Destination Data Byte 7
0 to 7	Number of destination data byte to be copied to source data byte 3
8 to 14	Setting prohibited
15	Do not copy the destination data byte

(p) Get data bytes from destination data bytes 4 to 7 (LD_DHI)

Command	Get Data Bytes from Destination Data Bytes 4 to 7
Code	19H
Mnemonic	LD_DHI <DEST>,<PAR>

DEST7 to DEST0	Destination
0 to 255	Number of message buffer the command uses as target

PAR15 to PAR12	Parameter for Destination Data Byte 4
0 to 7	Number of destination data byte to be copied to source data byte 4
8 to 14	Setting prohibited
15	Do not copy the destination data byte

PAR11 to PAR8	Parameter for Destination Data Byte 5
0 to 7	Number of destination data byte to be copied to source data byte 5
8 to 14	Setting prohibited
15	Do not copy the destination data byte

PAR7 to PAR4	Parameter for Destination Data Byte 6
0 to 7	Number of destination data byte to be copied to source data byte 6
8 to 14	Setting prohibited
15	Do not copy the destination data byte

PAR3 to PAR0	Parameter for Destination Data Byte 7
0 to 7	Number of destination data byte to be copied to source data byte 7
8 to 14	Setting prohibited
15	Do not copy the destination data byte

Copy all destination data to source (LD_DALL)

Command	Copy all Destination Data to Source (Including DLC)
Code	1AH
Mnemonic	LD_DALL <DEST>,<PAR>

DEST7 to DEST0	Destination
0 to 255	Number of message buffer the command uses as target

PAR0	Parameter
0	Copy all data except source identifier to source
1	Copy all data including source identifier to source

(q) Get source bit-string from destination bit-string, data bytes 0 to 3 (LD_DBLO)

Command	Get Source Bit-string from Destination Bit-string, Data Bytes 0 to 3
Code	1BH
Mnemonic	LD_DBLO <DEST>,<PAR>

DEST7 to DEST0	Destination
0 to 255	Number of message buffer the command uses as target

PAR14 to PAR10	Parameter for Start Position in Source
0 to 31	Start position (bit number) of bit-string in source

PAR9 to PAR5	Parameter for Bit-string Length
0 to 31	Length of bit-string in bits decremented by 1. **Remark:** A value of 0 means a length of 1, a value of 1 a length of 2, etc.

PAR4 to PAR0	Parameter for Start Position in Destination
0 to 7	Start position (bit number) of bit-string in destination (byte 0 to 3)

(r) Get source bit-string from destination bit-string, data bytes 4 to 7 (LD_DBHI)

Command	Get Source Bit-string from Destination Bit-string, Data Bytes 4 to 7
Code	1CH
Mnemonic	LD_DBHI <DEST>,<PAR>

DEST7 to DEST0	Destination
0 to 255	Number of message buffer the command uses as target

PAR14 to PAR10	Parameter for Start Position in Source
0 to 31	Start position (bit number) of bit-string in source

PAR9 to PAR5	Parameter for Bit-string Length
0 to 31	Length of bit-string in bits decremented by 1. **Remark:** A value of 0 means a length of 1, a value of 1 a length of 2, etc.

PAR4 to PAR0	Parameter for Start Position in Destination
0 to 7	Start position (bit number) of bit-string in destination (byte 4 to 7)

(2) ELISA Register Area

(a) Timer event pointer registers 0, 1 and 3 (TEP0, TEP1, TEP3)

The TEPn registers set the corresponding timer event pointers (n = 0, 1, 3).
This register can be read and written in 8-bit units only.

Figure 13-54: Timer Event Pointer Registers 0, 1 and 3 (TEP0, TEP1, TEP3)

	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
TEP0	TEP07	TEP06	TEP05	TEP04	TEP03	TEP02	TEP01	TEP00	910H	0000H
TEP1	TEP17	TEP16	TEP15	TEP14	TEP13	TEP12	TEP11	TEP10	911H	0000H
TEP3	TEP37	TEP36	TEP35	TEP34	TEP33	TEP32	TEP31	TEP30	913H	0000H

Bit Position	Bit Name	Function
7 to 0	TEPn7 to TEPn0	Pointer to command list for timer event n (n = 0, 1, 3)

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = PP_BASE + address offset

(b) Script event pointer and command counter register (SEPCC)

The SEPCC register controls the script pointer and commands

This register can be read and written in 8-bit and 16-bit units.

Figure 13-55: Script Event Pointer And Command Counter Register (SEPCC)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
SEPCC	CMDC7	CMDC6	CMDC5	CMDC4	CMDC3	CMDC2	CMDC1	CMDC0	SEP7	SEP6	SEP5	SEP4	SEP3	SEP2	SEP1	SEP0	914H	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15 to 8</td><td>CMDC7 to CMDC0</td><td>Pointer on command list for script event.
<table>
<tr><th>CMDC7 to CMDC0</th><th>Last Executed Command</th></tr>
<tr><td>0</td><td>1st command</td></tr>
<tr><td>1</td><td>2nd command</td></tr>
<tr><td>2</td><td>3rd command</td></tr>
<tr><td colspan="2">•
•
•</td></tr>
<tr><td>255</td><td>256th command</td></tr>
</table>
<b>Remark:</b> The CMDC7 to CMDC0 bits show the (offset) number of the command executed last. If an error condition occurs the offset value indicates the command, which has caused the error.</td></tr>
<tr><td>7 to 0</td><td>SEP7 to SEP0</td><td>Pointer to command list for script event.
<table>
<tr><th>SEP7 to SEP0</th><th></th></tr>
<tr><td>0 to 255</td><td>Number of first command for script event</td></tr>
</table></td></tr>
</table>

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = PP_BASE + address offset

(c) ELISA event processing status register (EEPS)

The EEPS register indicates the ELISA processing status.
This register can be read and written in 8-bit and 16-bit units.

Figure 13-56: ELISA Event Processing Status Register (EEPS)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
EEPS	EERC7	EERC6	EERC5	EERC4	EERC3	EERC2	EERC1	EERC0	EPS7	EPS6	EPS5	EPS4	EPS3	EPS2	EPS1	EPS0	916H	00FFH

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15 to 8</td><td>EERC7 to EERC0</td><td>Indicates the ELISA error code.
<table>
<tr><th>EERC7 to EERC0</th><th>ELISA Error Code</th></tr>
<tr><td>0</td><td>No error occurred since EER register was cleared last.</td></tr>
<tr><td>1 to 255</td><td>Error code</td></tr>
</table>
<b>Remark:</b> The EERC7 to EERC0 bits are read-only.</td></tr>
<tr><td>7 to 0</td><td>EPS7 to EPS0</td><td>Indicates the event processing source
<table>
<tr><th>EPS7 to EPS0</th><th>Event Processing Source</th></tr>
<tr><td>0 to 254</td><td>Number of message defined as source for event processing</td></tr>
<tr><td>255</td><td>No message has been defined to be source for actual event processing</td></tr>
</table></td></tr>
</table>

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = PP_BASE + address offset

(d) ELISA status register (ELSR)

The ELSR register indicates the ELISA status. The bits 7 to 0 can be set and cleared by writing to the register according to the special bit-set/clear method. (Refer to chapter 13.3.1)
This register can be read in 8-bit and 16-bit units. It can be written in 16-bit units only.

Figure 13-57: ELISA Event Processing Status Register (ELSR) (1/2)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
ELSR	0	0	0	0	1	1	0	1	CFLG	TBS	EER	PSE	PTE3	PTE2	PTE1	PTE0	918H	0920H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ELSR	0	ST_TBS	0	0	0	0	0	0	0	CL_TBS	CL_EER	CL_PSE	CL_PTE3	CL_PTE2	CL_PTE1	CL_PTE0	918H

Read

Bit Position	Bit Name	Function
7	CFLG	Determines the execution of commands with COND bit set (1). CFLG bit is modified by ELISA in dependence of the result when executing the commands DECR_CNT, TST_BIT. 0: Executing the DECR_CNT or TST_BIT command does not meet the criterion to set conditional execution. 1: Executing the DECR_CNT or TST_BIT command does meet the criterion to set conditional execution. **Remarks:** **1.** The execution of commands other than DECR_CNT or TST_BIT does always clear the CFLG bit. **2.** The value of CFLG bit does only influence the execution of the command which is executed directly after the DECR_CNT command or TST_BIT command and for which conditional execution is set by the COND bit.
6	TBS	Selects the temporary buffer bank. 0: Temporary buffer bits 0-31 are accessible by CPU 1: Temporary buffer bits 63-32 are accessible by CPU **Remarks:** **1.** As the local register address area is limited, the temporary buffer (64-bits) is split into 2 banks. The TBS bit indicates which of the buffer banks can be accessed by CPU. **2.** For operation of ELISA and handling of the temporary buffers by ELISA the TBS has no influence, that means the TBS is only relevant for the CPU, not for ELISA
5	EER	Indicates an ELISA error. Indicates a pending script. 0: ELISA operates without any error 1: An error occurred and ELISA stopped its operation **Remark:** Whenever an error occurs the EER flag is set and ELISA stops its operation. The CPU has to clear the EER flag to continue operation of ELISA. The error code can be read by EERC.
4	PSE	Indicates a pending script. 0: Script event is not pending. 1: Script event is pending.
3	PTE3	Indicates a pending timer event 3. 0: Timer event 3 is not pending. 1: Timer event 3 is pending.

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = PP_BASE + address offset

Figure 13-57: ELISA Event Processing Status Register (ELSR) (2/2)

Read

Bit Position	Bit Name	Function
2	PTE2	Indicates a pending timer event 2. 0: Timer event 2 is not pending. 1: Timer event 2 is pending.
1	PTE1	Indicates a pending timer event 1. 0: Timer event 1 is not pending. 1: Timer event 1 is pending.
0	PTE0	Indicates a pending timer event 0. 0: Timer event 0 is not pending. 1: Timer event 0 is pending.

Write

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>14, 6</td><td>ST_TBS,
CL_TBS</td><td>Sets/clears the TBS bit.
<table>
<tr><th>ST_TBS</th><th>CL_TBS</th><th>Status of TBS bit</th></tr>
<tr><td>0</td><td>1</td><td>TBS bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>TBS bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in TBS bit value.</td></tr>
</table></td></tr>
<tr><td>5</td><td>CL_ERR</td><td>Clears the ERR flag.
0: No change of ERR flag.
1: ERR flag is cleared (0).</td></tr>
<tr><td>4</td><td>CL_PSE</td><td>Clears the PSE flag.
0: No change of PSE flag.
1: PSE flag is cleared (0).</td></tr>
<tr><td>3</td><td>CL_PTE3</td><td>Clears the PTE3 flag.
0: No change of PTE3 flag.
1: PTE3 flag is cleared (0).</td></tr>
<tr><td>2</td><td>CL_PTE2</td><td>Clears the PTE2 flag.
0: No change of PTE2 flag.
1: PTE2 flag is cleared (0).</td></tr>
<tr><td>1</td><td>CL_PTE1</td><td>Clears the PTE1 flag.
0: No change of PTE1 flag.
1: PTE1 flag is cleared (0).</td></tr>
<tr><td>0</td><td>CL_PTE0</td><td>Clears the PTE0 flag.
0: No change of PTE0 flag.
1: PTE0 flag is cleared (0).</td></tr>
</table>

(e) ELISA last processed command register (ELC)

The ELC register indicates the upper 16-bit of the last processed command by ELISA.
The register is can be read in 16-bit units only.

Figure 13-58: ELISA Last Processed Command Register (ELC)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
ELC	ELC15	ELC14	ELC13	ELC12	ELC11	ELC10	ELC9	ELC8	ELC7	ELC6	ELC5	ELC4	ELC3	ELC2	ELC1	ELC0	91AH	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15 to 0</td><td>ELC15 to ELC0</td><td>Indicates the last processed command by ELISA.
<table>
<tr><th>ELC15 to ELC0</th><th>Last Processed Command by ELISA</th></tr>
<tr><td>0000H to FFFFH</td><td>Upper 16-bit of command processed last</td></tr>
</table>
<b>Remark:</b> The ELC shows only the upper 16-bit of the command processed last (END, EHDL, COND, CMD4 to CMD0, and DEST7 to DEST0 bits). If it necessary to check the lower 16-bits (PAR15 to PAR0) the command must be read from the command buffer.</td></tr>
</table>

Note: The address of an interrupt pending register is calculated according to the following formula:
effective address = PP_BASE + address offset

(f) ELISA temporary buffer registers (ETBL, ETBLH)

The ETBL and ETBH registers contain the temporary buffer
The registers can be read and written in 8-bit and 16-bit units.

Figure 13-59: ELISA Temporary Buffer Registers (ETBL, ETBH)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
ETBL	ETB15	ETB14	ETB13	ETB12	ETB11	ETB10	ETB9	ETB8	ETB7	ETB6	ETB5	ETB4	ETB3	ETB2	ETB1	ETB0	91CH	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
ETBH	ETB31	ETB30	ETB29	ETB28	ETB27	ETB26	ETB25	ETB24	ETB23	ETB22	ETB21	ETB20	ETB19	ETB18	ETB17	ETB16	91EH	0000H

when TBS = 0

Bit Position	Bit Name	Function
15 to 8 (ETBH)	ETB31 to ETB24	These bits reflect either the counter 3, or the upper byte of temporary sum 1.
7 to 0 (ETBH)	ETB23 to ETB16	These bits reflect either the counter 2, or the lower byte of temporary sum 1.
15 to 8 (ETBL)	ETB15 to ETB8	These bits reflect either the counter 1, or the upper byte of temporary sum 0.
7 to 0 (ETBL)	ETB7 to ETB0	These bits reflect either the counter 0, or the lower byte of temporary sum 0.

when TBS = 1

Bit Position	Bit Name	Function
15 to 8 (ETBH)	ETB31 to ETB24	These bits reflect either the counter 7, or the upper byte of temporary sum 3.
7 to 0 (ETBH)	ETB23 to ETB16	These bits reflect either the counter 6, or the lower byte of temporary sum 3.
15 to 8 (ETBL)	ETB15 to ETB8	These bits reflect either the counter 5, or the upper byte of temporary sum 2.
7 to 0 (ETBL)	ETB7 to ETB0	These bits reflect either the counter 4, or the lower byte of temporary sum 2.

Note: The address of an interrupt pending register is calculated according to the following formula:
effective address = PP_BASE + address offset

Chapter 14 A/D Converter

14.1 Features

- 10-bit resolution on-chip A/D converter
- Analog inputs: 12 channels
- Separate on-chip A/D conversion result registers for each analog input
  - 10 bits × 12 registers
- A/D conversion trigger modes
  - A/D trigger mode
  - A/D trigger polling mode
  - Timer trigger mode
- Successive approximation technique
- Voltage detection mode

14.2 Configuration

The A/D converter, which employs a successive approximation technique, performs A/D conversion operation using A/D scan mode registers 0 and 1 (ADSCM0, ADSCM1) and A/D conversion result registers ADCRm (m = 0 to 11).

(1) Input circuit

The input circuit selects an analog input (ANIm) according to the mode set in the ADSCM0 register and sends it to the sample and hold circuit (m = 0 to 11).

(2) Sample and hold circuit

The sample and hold circuit individually samples analog inputs sent sequentially from the input circuit and sends them to the comparator. It holds sampled analog inputs during A/D conversion.

(3) Voltage comparator

The voltage comparator compares the analog input voltage from the input with the output voltage of the D/A converter.

(4) D/A converter

The D/A converter is used to generate a voltage that matches an analog input.
The output voltage of the D/A converter is controlled by the successive approximation register (SAR).

(5) Successive approximation register (SAR)

The SAR is a 10-bit register that controls the output value of the D/A converter for comparing with an analog input voltage value. When an A/D conversion terminates, the current contents of the SAR (conversion result) are stored in an A/D conversion result register (ADCRm) (m = 0 to 11). When all specified A/D conversions terminate, there also is an A/D conversion termination interrupt (INTAD).

(6) A/D conversion result registers n (ADCRn) (n = 0 to 11)

ADCRn are 10-bit registers that hold A/D conversion results (n = 0 to 11). Whenever an A/D conversion terminates, the conversion result from the successive approximation register (SAR) is loaded. $\overline{\text{RESET}}$ input sets this to 0000H.

(7) Controller

The controller selects an analog input, generates sample and hold circuit operation timing, controls conversion triggers, specifies the conversion operation time, and sets the low power consumption mode according to the mode set in the ADSCM0 or ADSCM1 register.

(8) ANIm pins (m = 0 to 11)

The ANIm pins are the 12-channel analog input pins to analog converter.

Caution: Use input voltages to ANIm that are within the range of the ratings. In particular, if a voltage higher than AV_{DD} or lower than AV_{SS} (even one within the range of absolute maximum ratings) is input, the conversion value of that channel is undefined, and the conversion values of other channels also may be affected.

(9) AV_{REF} pin

The AV_{REF} pin is used to input reference voltage to the A/D converter. A signal input to the ANIm pin is converted to a digital signal based on the voltage applied between AV_{REF} and AV_{SS} (m = 0 to 11). If not using the AV_{REF} pin, connect it to AV_{DD} or AV_{SS}**Note**.

Note: When connecting the AV_{REF} pin to AV_{SS} the power consumption will be reduced.

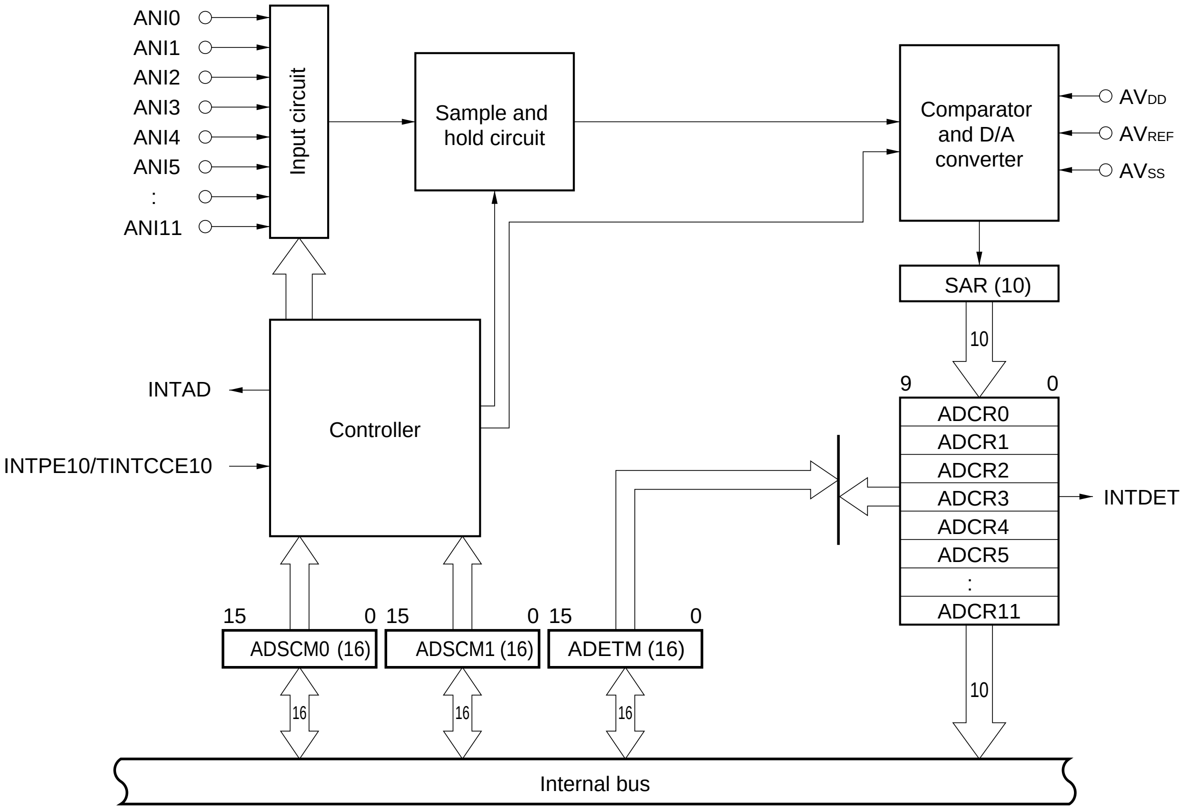
(10) AV_{SS} pin

The AV_{SS} pin is the ground voltage pin of the A/D converter. Even if not using A/D converter, always ensure that this pin has the same DC potential as the V_{SS5} pin.

(11) AV_{DD} pin

The AV_{DD} pin is the analog power supply pin of A/D converter. Even if not using A/D converter, always ensure that this pin has the same potential as the V_{DD5} pin.

Figure 14-1: Block Diagram of A/D Converter



Cautions: 1. Noise at an analog input pin (ANIm) or reference voltage input pin (AV_{REF}) may give rise to an invalid conversion result.

Software processing is needed in order to prevent this invalid conversion result from adversely affecting the system.

The following are examples of software processing.

•Use the average value of the results of multiple A/D conversions as the A/D conversion result.

•Perform A/D conversion multiple consecutive times and use conversion results with the exception of any abnormal conversion results that are obtained.

•If an A/D conversion result from which it is judged that an abnormality occurred in the system is obtained, do not perform abnormality processing at once but perform it upon reconfirming the occurrence of an abnormality.

2. Be sure that voltages outside the range [AV_{SS} to AV_{REF}] are not applied to pins being used as A/D converter and input pins.

14.3 Control Registers

(1) A/D scan mode register 0 (ADSCM0)

The ADSCM0 register is a 16-bit register that selects analog input pins, specifies operation modes, and controls conversion operation.

It can be read or written in 1-bit, 8-bit or 16-bit units. However, writing to the ADSCM0 register during A/D conversion operation interrupts the conversion operation and the data is lost. The conversion operation restarts as specified.

Figure 14-2: A/D Scan Mode Register 0 (ADSCM0) (1/2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ADSCM0	CE	CS	0	MS	PLM	TRG2	TRG1	TRG0	SANI3	SANI2	SANI1	SANI0	ANIS3	ANIS2	ANIS1	ANIS0	FFFFF200H	0000H

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>15</td><td>CE</td><td>Specifies enabling or disabling A/D conversion.
0: Disable
1: Enable</td></tr>
<tr><td>14</td><td>CS</td><td>Shows status of A/D converter. This bit is read-only.
0: Stopped
1: Operating</td></tr>
<tr><td>12</td><td>MS</td><td>Specifies operation mode of A/D converter.
0: Scan mode
1: Select mode</td></tr>
<tr><td>11 to 8</td><td>PLM,
TRG2 to TRG0</td><td>PLM: Specifies polling mode.
TRG2 toTRG0: Specifies trigger mode.
<table>
<tr><th>PLM</th><th>TRG2</th><th>TRG1</th><th>TRG0</th><th>Trigger Mode</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>A/D trigger mode</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>Timer trigger mode Note</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>A/D trigger polling mode</td></tr>
<tr><td colspan="4">Other than above</td><td>Setting prohibited</td></tr>
</table>
<b>Note:</b> The interrupt signal that can be selected as trigger is the timer E interrupt INTPE10/TINTCCE10.</td></tr>
</table>

Figure 14-2: A/D Scan Mode Register 0 (ADSCM0) (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7 to 4</td><td>SANI3 to SANI0</td><td>The bits SANI3 to SANI0 specify the analog input pin mode for which the 1st conversion is performed in scan mode.
These bits are ignored in select mode.
<table>
<tr><th>SANI3</th><th>SANI2</th><th>SANI1</th><th>SANI0</th><th>Scan Start Analog Input Pin</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>ANI0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>ANI1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>ANI2</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>ANI3</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>ANI4</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>ANI5</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>ANI6</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>ANI7</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>ANI8</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>ANI9</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>ANI10</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>ANI11</td></tr>
<tr><td colspan="4">Other than above</td><td>Setting prohibited</td></tr>
</table>
<b>Caution:</b> <b>Always set the conversion start analog input pin number that is set by bits SANI3-SANI0 to a smaller pin number than the conversion end analog input pin number that is set by bits ANIS3-ANIS0.</b></td></tr>
<tr><td>3 to 0</td><td>ANIS3 to ANIS0</td><td>ANIS3 to ANIS0 specifies the analog input pin in select mode.
In scan mode it specifies the last analog input pin for which a conversion is issued. The range of consecutive conversions is defined by the setting of SANI3 to SANI0 and ANIS3 to ANIS0, which lead to a number of conversions defined by: n= ANIS3 to ANIS0 - SANI3 to SANI0 + 1.
<table>
<tr><th>ANIS3</th><th>ANIS2</th><th>ANIS1</th><th>ANIS0</th><th>in Select Mode</th><th>In Scan Mode</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>ANI0</td><td>ANI0 [Note]</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>ANI1</td><td>ANI0 [Note] → ANI1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>ANI2</td><td>SANI[3-0] → ANI2</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>ANI3</td><td>SANI[3-0] → ANI3</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>ANI4</td><td>SANI[3-0] → ANI4</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>ANI5</td><td>SANI[3-0] → ANI5</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>ANI6</td><td>SANI[3-0] → ANI6</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>ANI7</td><td>SANI[3-0] → ANI7</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>ANI8</td><td>SANI[3-0] → ANI8</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>ANI9</td><td>SANI[3-0] → ANI9</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>ANI10</td><td>SANI[3-0] → ANI10</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>ANI11</td><td>SANI[3-0] → ANI11</td></tr>
<tr><td colspan="4">Other than above</td><td colspan="2">Setting prohibited</td></tr>
</table>
<b>Note:</b> Setting for SANI3 to SANI0 has to be set to 0 to start the scan mode at analog input pin ANI0.

<b>Remarks:</b> <b>1.</b> SANI < ANIm
<b>2.</b> m = 1 to 11</td></tr>
</table>

(2) A/D scan mode register 1 (ADSCM1)

The ADSCM1 register is a 16-bit register that sets the conversion time of the A/D converter.
It can be read or written in 1-bit, 8-bit, or 16-bit units.

Figure 14-3: A/D Scan Mode Register 1 (ADSCM1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ADSCM1	ADPS	SPC3	SPC2	SPC1	SPC0	FR2	FR1	FR0	0	0	0	0	0	0	0	0	FFFFF202H	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15</td><td>ADPS</td><td>A/D converter power saving Note
0: Disable power saving mode
1: Enable power saving mode
<b>Note:</b> Conversion with ADPS bit set requires stabilization time for analog circuit. Conversion in power saving mode is only allowed in select mode and select mode with polling mode. After conversion, within 5 µs (after post-stabilization time) the analog circuit turns in low power mode again. Writing CE bit must be delayed 5 µs to prevent the analog circuit from illegal action. In select-polling mode stabilization time is required for first conversion only. Conversion request during post-stabilization time is valid but result can not be guaranteed.
<b>Remark:</b> Power saving functions are only engaged, if this bit is set, after the select or select-polling mode has been activated.</td></tr>
<tr><td>13 to 11</td><td>SPC3 to SPC0</td><td>Specifies the sampling time (T_{SMPAD}).
<table>
<tr><th>SPC3</th><th>SPC2</th><th>SPC1</th><th>SPC0</th><th>Sampling Clocks</th><th>Sampling Time Setting (T_{SMPAD})</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>2 (TSTAGE/10)</td><td>2 (TSTAGE/10) / f_{CPU}</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>3 (TSTAGE/10)</td><td>3 (TSTAGE/10) / f_{CPU}</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>4 (TSTAGE/10)</td><td>4 (TSTAGE/10) / f_{CPU}</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>5 (TSTAGE/10)</td><td>5 (TSTAGE/10) / f_{CPU}</td></tr>
<tr><td colspan="4">Others</td><td colspan="2">Setting prohibited</td></tr>
</table></td></tr>
<tr><td>10 to 8</td><td>FR2 to FR0</td><td>Specifies the compare time (T_{CMPAD}).
<table>
<tr><th rowspan="2">FR2</th><th rowspan="2">FR1</th><th rowspan="2">FR0</th><th rowspan="2">Compare Clocks
(TSTAGE)</th><th colspan="2">Compare Time [µs]</th></tr>
<tr><th>f_{CPU} = 20 MHz</th><th>f_{CPU} = 16 MHz</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>140</td><td>7.00</td><td>8.75</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>100</td><td>5.00</td><td>6.25</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>70</td><td>-</td><td>4.375</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>50</td><td>-</td><td>-</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>40</td><td>-</td><td>-</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>30</td><td>-</td><td>-</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>20</td><td>-</td><td>-</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Setting prohibited</td><td>-</td><td>-</td></tr>
</table>
<b>Caution:</b> <b>Conversion time = Sampling time + Compare Time</b>
<b>Remark:</b> f_{CPU}: Internal system clock.</td></tr>
</table>

Caution: Do not write to the ADSCM1 register during A/D conversion operation. If a write is performed, conversion operation is suspended and subsequently terminates.

(a) Conversion time setting

In order to prevent a drastic change of A/D conversion time even when the oscillation frequency is changed, the conversion speed of an operation stage can be adjusted. By the selection bits FR2 to FR0 in the ADSCM1 register the SAR compare time T_{CMPAD} can be set in the range of $20/f_{CPU}$ to $140/f_{CPU}$. Furthermore the sample time T_{SMPAD} can be selected by the control bits SPC2 to SPC0 in the ADSCM1 register to alter the parameters of conversion speed and accuracy.

However, the settings modifying the compare time T_{CMPAD} must keep the following relation.

$$T_{CMPAD} \geq 4.16\ \mu s$$

The conversion time results by the addition of the sample time T_{SMPAD} and the compare time T_{CMPAD}.

$$T_{CONV} = T_{CMPAD} + T_{SMPAD}$$

Example:

Provided that	f_{CPU}	$= 16$ MHz
	T_{SMPAD}	$= 2\ (T_{STAGE}/10)\ /\ f_{CPU}$

The compare time has to be set to

$$T_{CMPAD} = 70/f_{CPU} = 4,375\mu s \geq 4.16\ \mu s$$

Therefore the setting of bits FR2 to FR0 = 010B will be chosen.
The sampling time is

$$T_{SMPAD} = \frac{2 \times T_{STAGE}}{10 \times f_{CPU}} = \frac{2 \times 70}{10 \times 16MHz} = 0,875\mu s$$

By this the conversion time results in

$$T_{CONV} = 5,25\mu s$$

Caution: When A/D conversion is started by internal timer interrupt signal (TINTCCE10) an additional setup time has to be added (refer to “Start of conversion trigger setup timing” on page 495.

(b) Start of conversion trigger setup timing

When starting the start of conversion by internal timer interrupt signal (TINTCCE10) an additional setup time T_{TRGAD} of 8 system clocks has to be considered.

$$T_{TRGAD} = 8/f_{CPU}$$

Thus the total time of A/D conversion T_{TOTAL} including the trigger setup time is as follows.

- Select mode

$$T_{TOTAL} = T_{TRGAD} + T_{CONV}$$

- Scan mode

$$T_{TOTAL} = N_{CHANNEL} \times (T_{TRGAD} + T_{CONV})$$

$$N_{CHANNEL} = \text{Number of channels to scan}$$

Example:

Provided that	f_{CPU} = 16 MHz
	T_{SMPAD} = 2 (T_{STAGE} / 10 / f_{CPU})
	N = 8
	Scan mode is selected

The total A/D conversion time becomes

$$T_{TOTAL} = N_{CHANNEL} \cdot (T_{TRGAD} + T_{CONV}) = 8 \cdot (0.5\ \mu s + 5.25\ \mu s) = 46 \mu s$$

(3) A/D voltage detection mode register (ADETM)

The ADETM register is a 16-bit register that sets voltage detection mode. In voltage detection mode a reference voltage value is compared with the analog input pin for which voltage detection is being performed, and an interrupt is set in response to the comparison result.

This register can be read or written in 1-bit, 8-bit, or 16-bit units.

Figure 14-4: A/D Voltage Detection Mode Register (ADETM)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ADETM	DETEN	DETLH	DET ANI3	DET ANI2	DET ANI1	DET ANI0	DET CMP9	DET CMP8	DET CMP7	DET CMP6	DET CMP5	DET CMP4	DET CMP3	DET CMP2	DET CMP1	DET CMP0	FFFFF204H	0000H

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>15</td><td>DETEN</td><td>Specifies voltage detection mode.
0: Operate in normal mode
1: Operate in voltage detection mode</td></tr>
<tr><td>14</td><td>DETLH</td><td>Sets voltage comparison detection.
0: Generate INTDET interrupt,
if reference voltage value > analog input pin voltage.
1: Generate INTDET interrupt,
if reference voltage value < analog input pin voltage.</td></tr>
<tr><td>13 to 10</td><td>DETANI3
to
DETANI0</td><td>Selects analog input pin to compare to reference voltage value set by bits DETCMP9-DETCMP0 when in voltage detection mode.
<table>
<tr><th>DETANI3</th><th>DETANI2</th><th>DETANI1</th><th>DETANI0</th><th>Voltage Detection Analog Input Pin</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>ANI0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>ANI1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>ANI2</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>ANI3</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>ANI4</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>ANI5</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>ANI6</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>ANI7</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>ANI8</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>ANI9</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>ANI10</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>ANI11</td></tr>
<tr><td colspan="4">Other than above</td><td>Setting prohibited</td></tr>
</table>
</td></tr>
<tr><td>9 to 0</td><td>DETCMP9
to
DETCMP0</td><td>Sets reference voltage value to compare with analog input pin selected by bits DETANI3 - DETANI0.</td></tr>
</table>

Caution: Do not write to the an ADETM register during A/D conversion operation. If a write is performed, conversion is suspended and it subsequently terminates. Also, the polling mode needs to be re-engaged by writing the CE and PLM bits of the ADSCM0 register.

(4) A/D conversion result registers 0 to 11 (ADCR0 to ADCR11)

The ADCRm registers are 10-bit registers that hold the results of A/D conversions (m = 0 to 11).
These registers can only be read in 16-bit units.
When reading 10 bits of data of an A/D conversion result from an ADCRm register, only the lower 10 bits are valid and the upper 6 bits always read 0.

Figure 14-5: A/D Conversion Result Registers 0 to 11 (ADCR0 to ADCR11)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ADCRm	0	0	0	0	0	0	ADCRm9	ADCRm8	ADCRm7	ADCRm6	ADCRm5	ADCRm4	ADCRm3	ADCRm2	ADCRm1	ADCRm0	see Table 14-1	0000H

Table 14-1: Correspondence between ADCRm (m = 0 to 11) Register Names and Addresses

Register name	Address
ADCR0	FFFFF210H
ADCR1	FFFFF212H
ADCR2	FFFFF214H
ADCR3	FFFFF216H
ADCR4	FFFFF218H
ADCR5	FFFFF21AH
ADCR6	FFFFF21CH
ADCR7	FFFFF21EH
ADCR8	FFFFF220H
ADCR9	FFFFF222H
ADCR10	FFFFF224H
ADCR11	FFFFF226H

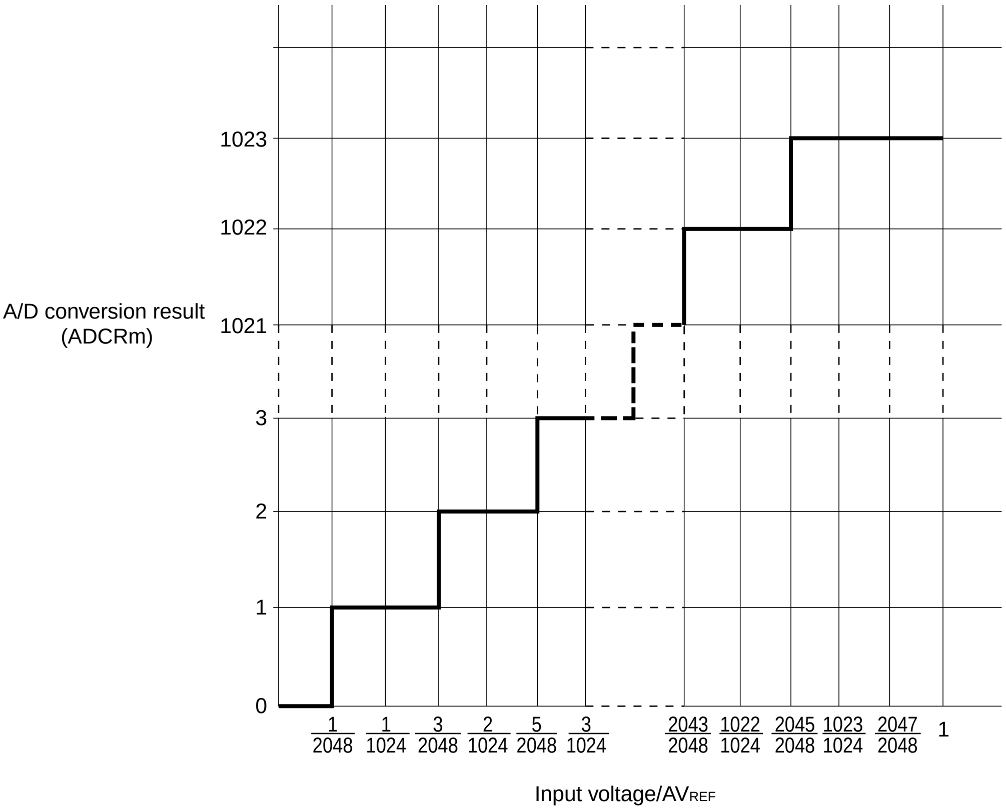
The correspondence between each analog input pin and the ADCRm registers is shown in Table 14-2.

Table 14-2: Correspondence between each Analog Input Pin and ADCRm Registers

Analog Input Pin	A/D Conversion Result Register
ANI0	ADCR0
ANI1	ADCR1
ANI2	ADCR2
ANI3	ADCR3
ANI4	ADCR4
ANI5	ADCR5
ANI6	ADCR6
ANI7	ADCR7
ANI8	ADCR8
ANI9	ADCR9
ANI10	ADCR10
ANI11	ADCR11

The following figure shows the relationship between analog input voltage and A/D conversion results.

Figure 14-6: Relationship Between Analog Input Voltages and A/D Conversion Results



Remark: m = 0 to 11

14.4 Interrupt Requests

The A/D converter generates two kinds of interrupts.

- A/D conversion termination interrupt (INTAD)
- Voltage detection interrupt (INTDET)

(1) A/D conversion termination interrupt (INTAD)

In A/D conversion enabled status, an A/D conversion termination interrupt is generated when a specified number of A/D conversions have terminated.

(2) Voltage detection interrupt (INTDET)

In voltage detection mode (DETEN bit of ADETM register = 1), the value of the ADCRm register of the relevant analog input pin is compared to the reference voltage set in the DETCMP[9:0] bits of the ADETM register and a voltage detection interrupt is generated in response to the value of the DETLH bit of the ADETM register (m = 0 to 11).

14.5 A/D Converter Operation

14.5.1 A/D converter basic operation

A/D conversion is performed using the following procedure.

(1) Set the analog input selection and the operation mode and trigger mode specifications using the ADSCM0 **Note 1**. Setting (1) the CE bit of the ADSCM0 register when in A/D trigger mode or A/D trigger polling mode starts the A/D conversion. In timer trigger mode, the status becomes trigger standby **Note 2**.

(2) When the A/D conversion starts, the analog input is compared with the voltage generated by the D/A converter.

(3) When the 10-bit comparison terminates, the conversion result is started in the ADCRm register. When the specified number of A/D conversions have terminated, an A/D conversion termination interrupt (INTAD) is generated.

Notes: **1.** If the contents of the ADSCM0 register are changed during A/D conversion operation, the A/D conversion operation preceding the change stops and a conversion result is not stored in the ADSCRm register. Conversion operation is initialized and conversion starts from the beginning.

2. In timer trigger mode, there is a transition to trigger standby status when the CE bit of the ADSCM0 register is set to 1. A/D conversion operation is activated by a trigger signal and there is a return to trigger standby status when the A/D conversion operation terminates.

Remark: m = 0 to 11

14.5.2 Operation modes and trigger modes

Several conversion operations can be specified for A/D converter by specifying operation modes and trigger modes. Operation modes and trigger modes are set using the ADSCM0 register.
The relationship between operation modes and trigger modes is shown below.

Trigger Mode	Operation Mode	Setting of ADSCM0
AD trigger	Select	×××10000××××××××B
	Scan	×××00000××××××××B
AD trigger polling	Select	×××11000××××××××B
	Scan	×××01000××××××××B
Timer trigger	Select	×××10001××××××××B
	Scan	×××00001××××××××B

(1) Trigger modes

The following trigger modes that serve as the start timing of A/D conversion processing are available: A/D trigger mode, A/D trigger polling mode and timer trigger mode.
These trigger modes are set using the ADSCM0 register.

(a) A/D trigger mode
In this mode, the A/D conversion is started by setting the CE bit of the register ADSCM0. This starts the conversion timing for the analog input(s) ANIm.
To restart the AD conversion after the INTAD interrupt (conversion finished; CS = 0), the CE bit has to be set again to engage the next conversion.

(b) A/D trigger polling mode
In this mode, the A/D conversion is started by setting the CE bit of the register ADSCM0. This starts the conversion timing for the analog input(s) ANIm.
A restart the AD conversion after the INTAD interrupt (conversion finished; CS = 1), is not necessary.
The specified analog input is converted serially until the CE bit is set to 0. An INTAD interrupt occurs each time, when a conversion terminates.

(c) Timer trigger mode
In this mode, the A/D conversion is started by a trigger signal derived from the INTPE10/ TINTCCE10 interrupt.

Remark: m = 0 to 11

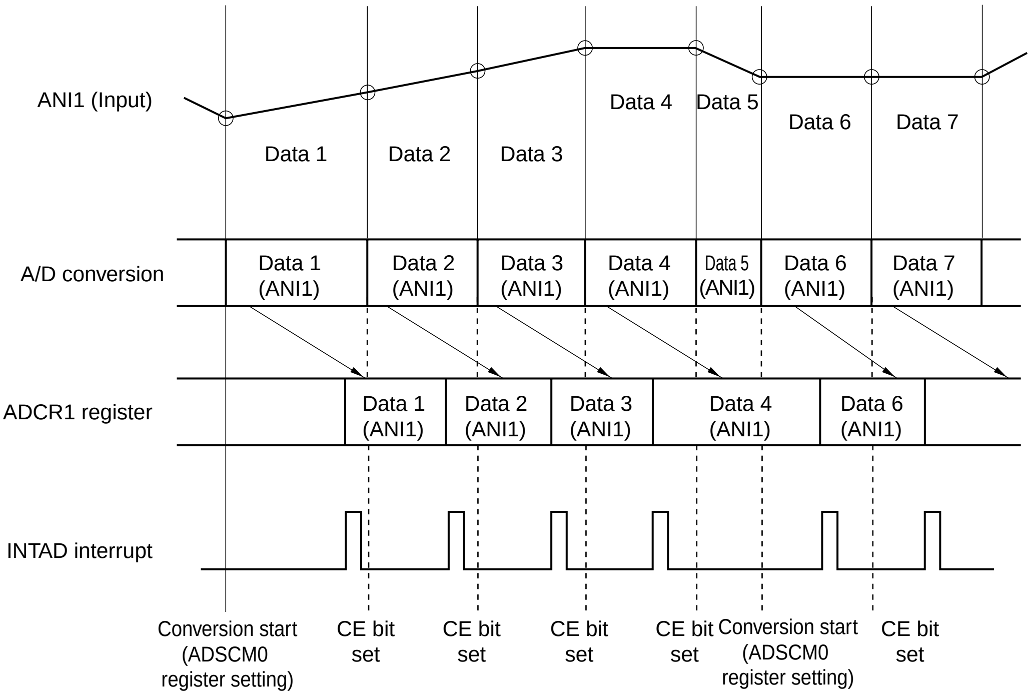
(2) Operation modes

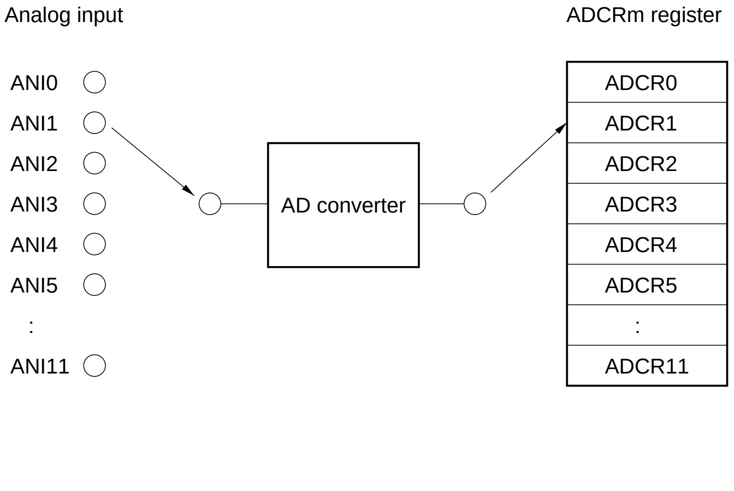
The two operation modes are select mode and scan mode. These modes are set using the ADSCM0 register.

(a) Select mode

In select mode the A/D converts one analog input specified in the ADSCM0 register. The conversion result is stored in the ADCRm register corresponding to the analog input (ANIm) (m = 0 to 11).

Figure 14-7: Example of Select Mode Operation Timing (ANI1)

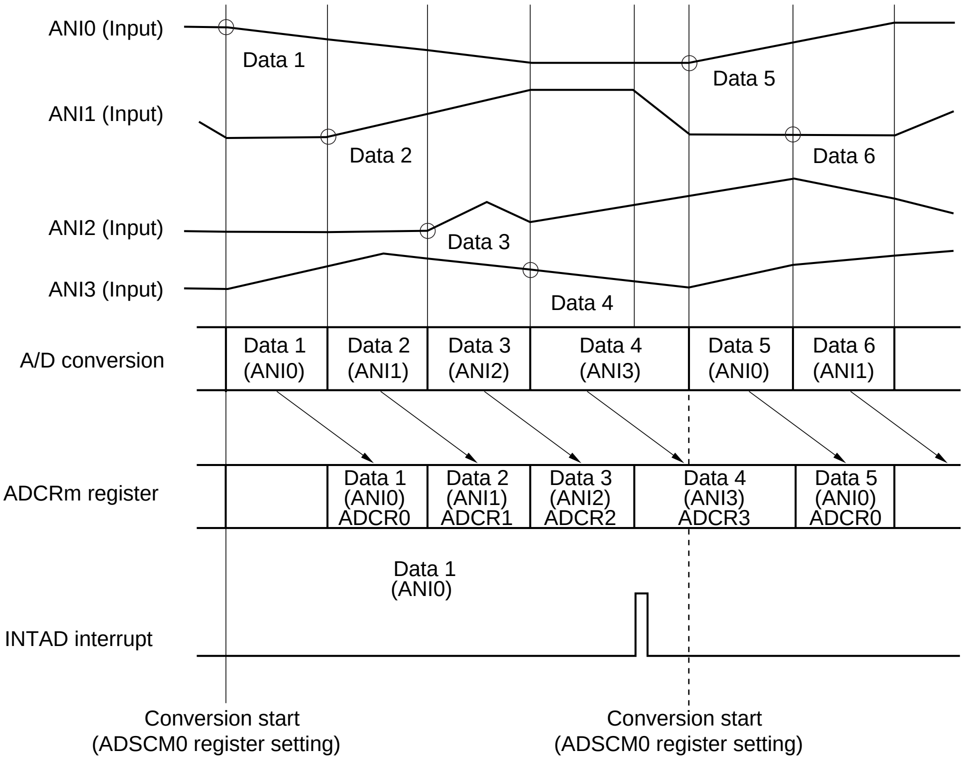


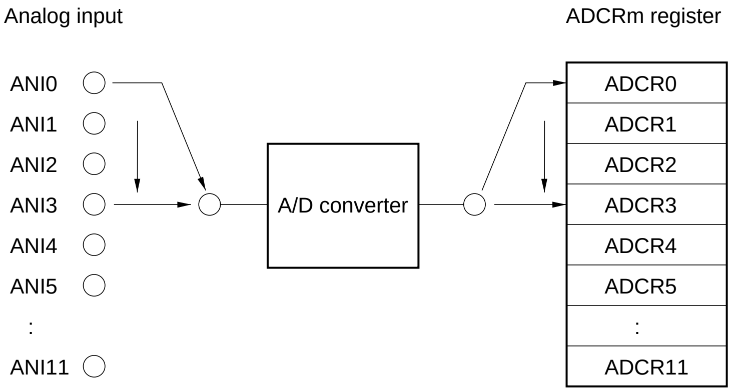


(b) Scan mode

The scan mode sequentially selects and converts pin input voltage from the A/D conversion start analog input pin through the A/D conversion termination analog input pin specified in the ADSCM0 register.
It stores the A/D conversion result in the ADCRm register corresponding to the analog input (m = 0 to 11). When the specified analog input conversion terminates, there is an A/D conversion termination interrupt (INTAD).

Figure 14-8: Example of Scan Mode Operation Timing (4-Channel Scan (ANI0 to ANI3))





14.6 Operation in A/D Trigger Mode

Setting the CE bit of the ADSCM0 register to 1 starts A/D conversion immediately.

14.6.1 Operation in select mode

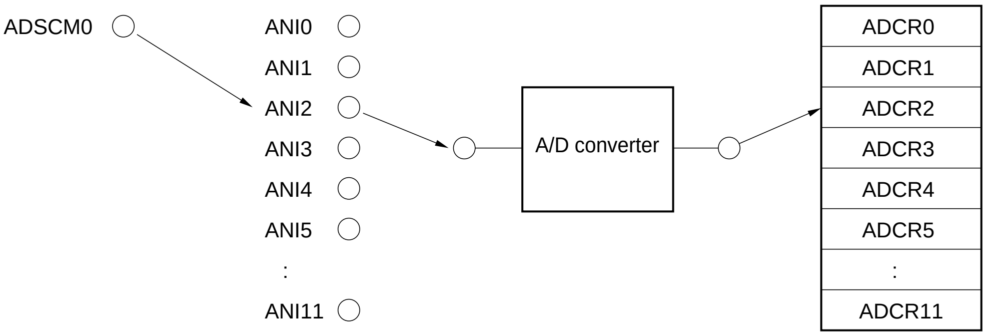
One analog input specified in the ADSCM0 register is A/D converted at a time and the result is stored in an ADCRm register. Analog inputs correspond one-to-one with ADCRm register (m = 0 to 11).
An A/D conversion termination interrupt (INTAD) is generated for each A/D conversion termination (CS bit = 0).

Analog input	A/D conversion result register
ANIm	ADCRm

To restart A/D conversion, set the CE bit of the ADSCM0 register.
This is optimal for an application that reads a result for each A/D conversion.

Remark: m = 0 to 11

Figure 14-9: Example of Select Mode (A/D Trigger Select) Operation (ANI2)



(1) CE bit of ADSCM0 = 1 (Enabled)
(2) A/D conversion of ANI2
(3) Store conversion result in ADCR2
(4) Generate INTAD interrupt

14.6.2 Operation in scan mode

The pins from the first analog input pin through the last analog input pin, specified in the ADSCM0 register, are sequentially selected and A/D converted. The A/D conversion result is stored in the ADCRm register corresponding to the analog input (m = 0 to 11). When conversion stops through the last analog input pin, an A/D conversion interrupt (INTAD) is generated, which terminates A/D conversion (CS bit of ADSCM0 register = 0).

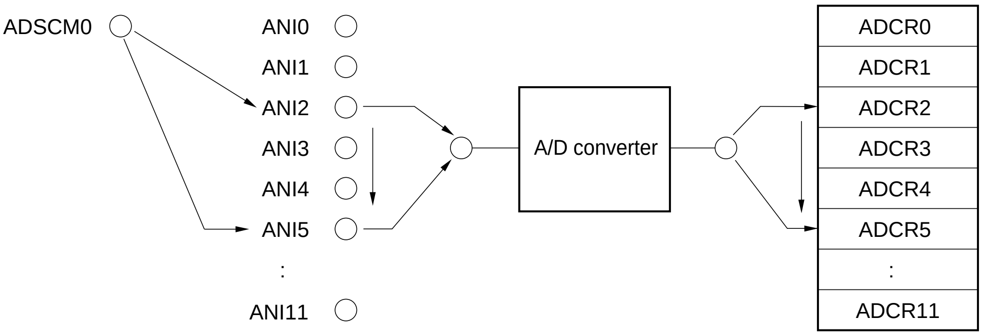
Analog input	A/D conversion result register
ANIn**Note 1**	ADCRn
:	:
ANIm**Note 2**	ADCRm

To restart A/D conversion, set the CE bit of the ADSCM0 register. This is optimal for an application that regularly monitors multiple analog inputs.

Notes: **1.** n = SANI [3:0] = 0 to 10
2. m = ANIS [3:0] = 1 to 11

Caution: Always set SANI[3:0] < ANIS[3:0]
Exception: When bits SANI[3:0] = ANIS[3:0] = 0, just the analog input ANI0 is scanned.

Figure 14-10: Example of Scan Mode (A/D Trigger Scan) Operation (ANI2-ANI5)



(1) CE bit of ADSCM0 = 1 (Enabled)
(2) A/D conversion of ANI2
(3) Store conversion result in ADCR2
(4) A/D conversion of ANI3
(5) Store conversion result in ADCR3
(6) A/D conversion of ANI4
(7) Store conversion result in ADCR4
(8) A/D conversion of ANI5
(9) Store conversion result in ADCR5
(10) Generate INTAD interrupt

14.7 Operation in A/D Trigger Polling Mode

Setting the CE bit of the ADSCM0 register to 1 starts the A/D conversion immediately.
Both select mode and scan mode can be continued with A/D trigger polling mode. Since the CS bit of the ADSCM0 register remains 1 after an INTAD interrupt in this mode, it is not necessary to set the CE bit to restart the A/D conversion.

14.7.1 Operation in select mode

The analog input specified in the ADSCM0 register is A/D converted. The conversion result is stored in the ADCRm register (m = 0 to 11).
One analog input is A/D converted at a time and the result is stored in one ADCRm register. Analog inputs correspond one-to-one with ADCRm register.
An A/D conversion termination interrupt (INTAD) is generated for each A/D conversion termination.
A/D conversion operation is repeated until the CE bit = 0 (CS bit = 1).

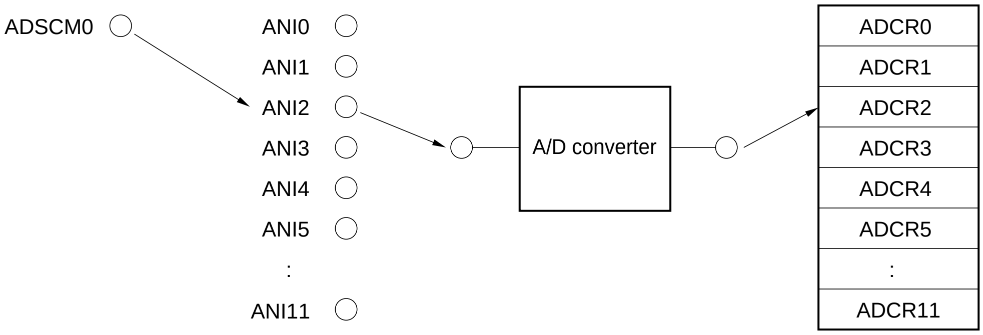
Analog input	A/D conversion result register
ANIm	ADCRm

In A/D trigger polling mode, it is not necessary to set the CE bit of the ADSCM0 register to restart the A/D conversion operation [Note].
This is optimal for applications that regularly read A/D conversion values.

Note: If the ADCRm register is not read before the next A/D conversion has been finished, its contents is overwritten.

Remark: m = 0 to 11

Figure 14-11: Example of Select Mode (A/D Trigger Polling Select) Operation (ANI2)



(1) CE bit of ADSCM0 = 1 (Enabled)
(2) A/D conversion of ANI2
(3) Store conversion result in ADCR2
(4) Generate INTAD interrupt
(5) Return to (2)

14.7.2 Operation in scan mode

The pins from the first analog input pin through the last analog input pin, specified in the ADSCM0 register, are sequentially selected and A/D converted. The A/D conversion result is stored in the ADCRm register corresponding to the analog input (m = 0 to 11). When conversion stops through the last analog input pin, an A/D conversion termination interrupt (INTAD) is generated.
A/D conversion operation repeats until the CE bit = 0 (CS bit = 1).

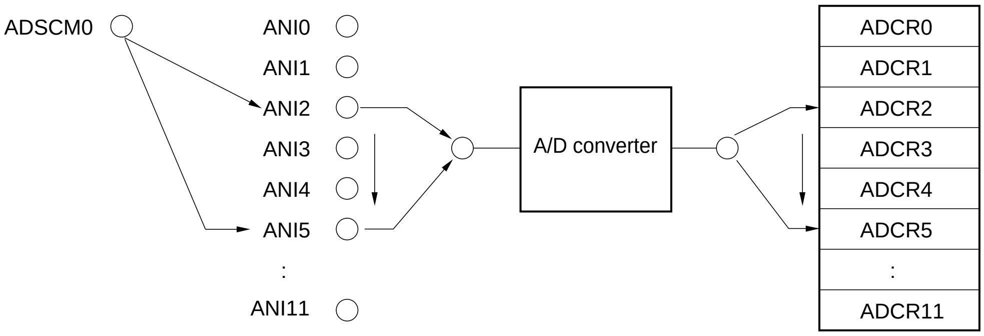
Analog input	A/D conversion result register
ANIn[Note 1]	ADCRn
:	:
ANIm[Note 2]	ADCRm

It is not necessary to set the CE bit of the ADSCM0 register as the A/D conversion restarts automatically [Note 3].
This is optimal for applications that regularly read A/D conversion values.

Notes: **1.** n = SANI [3:0] = 0 to 10
2. m = ANIS [3:0] = 1 to 11
3. If the ADCRm register is not read before the next A/D conversion has been finished, its contents is overwritten.

Caution: Always set SANI[3:0] < ANIS[3:0]
Exception: When bits SANI[3:0] = ANIS[3:0] = 0, just the analog input ANI0 is scanned.

Figure 14-12: Example of Scan Mode (A/D Trigger Polling Scan) Operation (ANI2 to ANI5)



(1) CE bit of ADSCM0 = 1 (Enabled)
(2) A/D conversion of ANI2
(3) Store conversion result in ADCR2
(4) A/D conversion of ANI3
(5) Store conversion result in ADCR3
(6) A/D conversion of ANI4
(7) Store conversion result in ADCR4
(8) A/D conversion of ANI5
(9) Store conversion result in ADCR5
(10) Generate INTAD interrupt
(11) Return to (2)

14.8 Operation in Timer Trigger Mode

The A/D converter can set an interrupt signal as a conversion trigger for up to 12 channels of analog input (ANI0 to ANI11).
The interrupt signal that can be selected as trigger is the Timer E 0 interrupt INTPE10/TINTCCE10.

14.8.1 Operation in select mode

One analog input (ANI0 to ANI) specified by the ADSCM0 register is A/D converted. The conversion result is stored in the ADCRm register corresponding to the analog input (m = 0 to 11).
This is optimal for applications that read A/D conversion values synchronized to a trigger.

(1) Timer trigger select mode

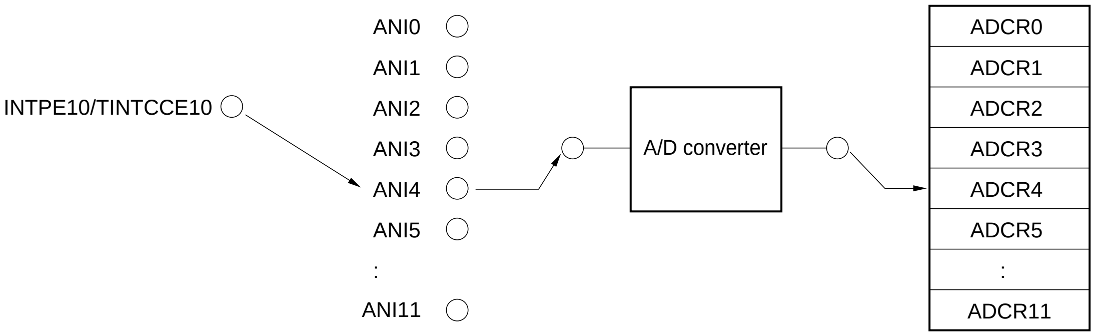
Taking the interrupt signal as a trigger, one analog input at a time is A/D converted and the result is stored in one ADCRm register. An A/D conversion termination interrupt (INTAD) is generated for each A/D conversion, which terminates A/D conversion (CS = 0).

Trigger	Analog Input	A/D Conversion Result Register
Timer E 0 Interrupt	ANIm	ADCRm

After A/D conversion termination, the A/D converter changes to trigger wait status (CE = 1). It performs A/D conversion operation again, when the interrupt signal occurs.

Remark: m = 0 to 11

Figure 14-13: Example of Timer Trigger Select Mode Operation (ANI4)



(1) CE bit of ADSCM0 = 1 (Enabled)
(2) INTPE10/TINTCCE10 interrupt generation
(3) A/D conversion of ANI4
(4) Store conversion result in ADCR4
(5) INTAD interrupt generation

14.8.2 Operation in scan mode

Analog input pins specified by register ADSCM0 are selected sequentially, and the specified number of A/D conversions are performed by using the Timer E 0 interrupt as a trigger. The conversion results are stored in the ADCRm registers corresponding to the analog inputs (m = 0 to 11).
This is optimal for applications that regularly monitor multiple analog inputs.

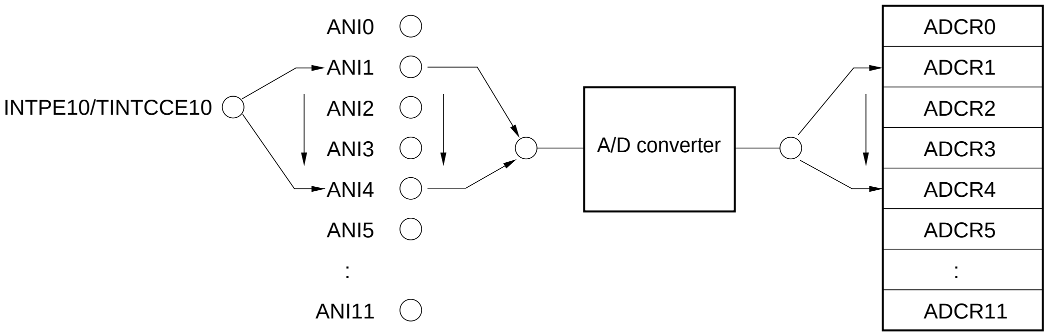
(1) Timer trigger scan mode

Using the interrupt signal from Timer E 0 as a trigger, the conversion start analog input pin through the conversion termination analog input pin specified by the ADSCM0 register are sequentially selected and A/D converted. Analog inputs correspond one-to-one with ADCRm register. When all of the specified A/D conversions have terminated, an A/D conversion termination interrupt (INTAD) is generated (CS = 0).

Trigger	Analog Input	A/D Conversion Result Register
INTPE10/TINTCCE10	ANI0	ADCR0
	ANI1	ADCR1
	ANI2	ADCR2
	ANI3	ADCR3
	ANI4	ADCR4
	ANI5	ADCR5
	ANI6	ADCR6
	ANI7	ADCR7
	ANI8	ADCR8
	ANI9	ADCR9
	ANI10	ADCR10
	ANI11	ADCR11

After all of the specified A/D conversions have terminated, the A/D converter changes to trigger wait status (CE = 1). It performs A/D conversion operation again, when the interrupt signal occurs.

Figure 14-14: Example of Timer Trigger Scan Mode Operation (ANI1 to ANI4)



(1) CE bit of ADSCM0 = 1 (Enabled)

(2) INTPE10/TINTCCE10 interrupt generation

(3) A/D conversion of ANI1

(4) Store conversion result in ADCR1

(5) A/D conversion of ANI2

(6) Store conversion result in ADCR2

(7) A/D conversion of ANI3

(8) Store conversion result in ADCR3

(9) A/D conversion of ANI4

(10) Store conversion result in ADCR4

(11) INTAD interrupt generation

Remark: Set to scan ANI1 to ANI4

14.9 Precautions

14.9.1 Stopping conversion operation

If the CE bit of the ADSCM0 register is cleared during conversion operation, all conversion operations are stopped, and a conversion result is not stored in the ADCRm register (m = 0 to 11).

14.9.2 Trigger input during conversion operation

If a trigger is input during conversion operation, that trigger input is ignored.

14.9.3 Timer trigger interval

Make the trigger interval (input time interval) in timer trigger mode longer than the conversion time specified by the FR2 to FR0 bits of the ADSCM1 register.

(1) When interval = 0

If multiple triggers are input simultaneously, the analog input whose ANIm pin number is smallest is converted. The other trigger signals input at the same time are ignored (m = 0 to 11).

(2) When 0 < interval < conversion time

If a timer trigger is input during conversion operation, that trigger input is ignored.
If conversion operation is suspended, a conversion result is not stored in the ADCRm register (m = 0 to 11).

(3) When interval = conversion time

If a timer trigger is input at the same time when the conversion terminates, interrupt generation and ADCRm register storage of the actual value are performed correctly.

14.9.4 Operation in standby modes

(1) HALT mode

The A/D converter continues the operation. When recover from HALT mode the ADSCM0, ADSCM1 and ADCRm registers maintain their values (m = 0 to 11).

(2) WATCH mode, IDLE mode, software STOP mode

Since clock supply to A/D converter stops, conversion operation is not performed.
If released by $\overline{\text{RESET}}$, NMI, or maskable interrupt input, the ADSCM0, ADSCM1 and ADCRm registers maintain their values.
However, if IDLE mode or software STOP mode is set during conversion operation, conversion operation stops. If released by NMI input, conversion resumes but the conversion result written in the ADCRm register becomes undefined (m = 0 to 11).

Remark: Connecting the AV_{REF} pin to AV_{SS} in IDLE mode, or software STOP mode will further reduce the power consumption.

[MEMO]

Chapter 15 LCD Controller/Driver

15.1 LCD Controller/Driver Functions

The functions of the LCD controller/driver incorporated in the µPD70F3123 subseries are shown below.

(1) Automatic output of segment signals and common signals is possible by automatic reading of the display data memory.

(2) Display mode

- 1/4 duty (1/3 bias)

(3) Any of four frame frequencies can be selected in each display mode.

(4) Maximum of 40 segment signal outputs (S0 to S39); 4 common signal outputs (COM0 to COM3). All segment outputs can be switched to input/output ports.
PAH7/S0 to PAH0/S7, PAL15/S8 to PAL0/S23 and PCS2/S24 to PCS4/S26 and PCT0/S27 to PCT4/S31 and PCM0/S32 to PCM1/S33 and P60/S34 to P65/S39 are bitwise switchable.

(5) The operation with the subsystem clock is not available.

Table 15-1: Maximum Number of Display Pixels

Bias Method	Time Division	Common Signals Used	Maximum Number of Display Pixels
1/3	4	COM0 to COM3	160 (40 segments x 4 commons)

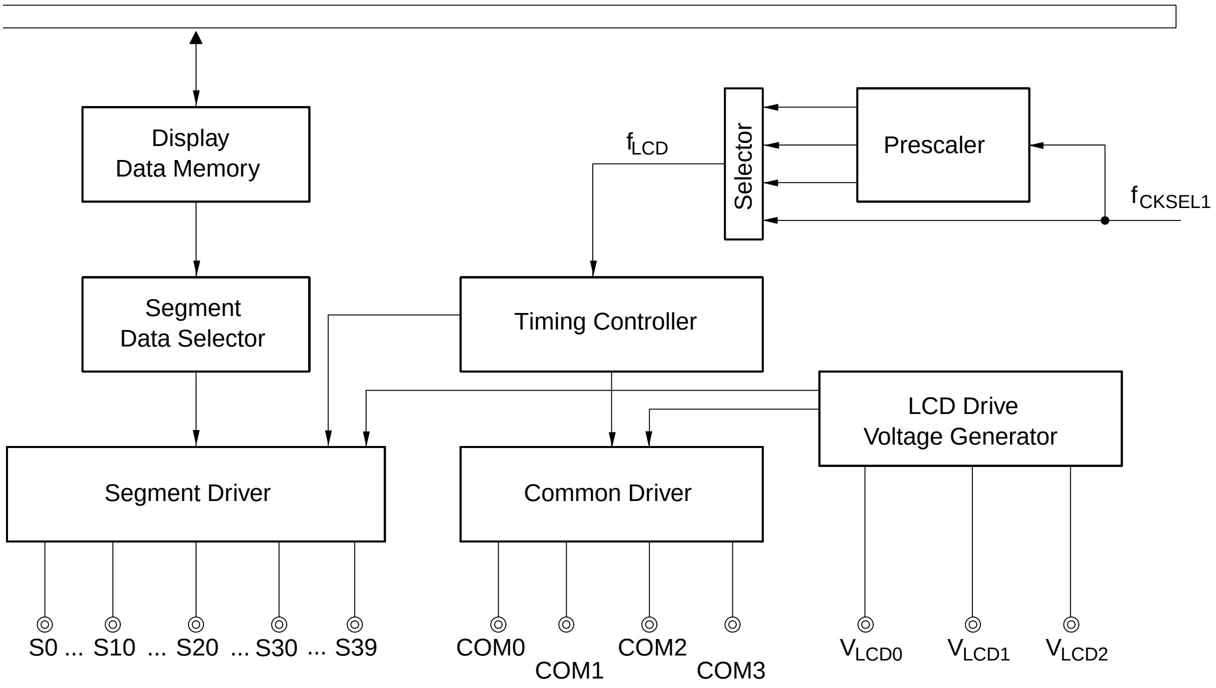
15.2 LCD Controller/Driver Configuration

The LCD controller/driver consists of the following hardware.

Table 15-2: LCD Controller/Driver Configuration

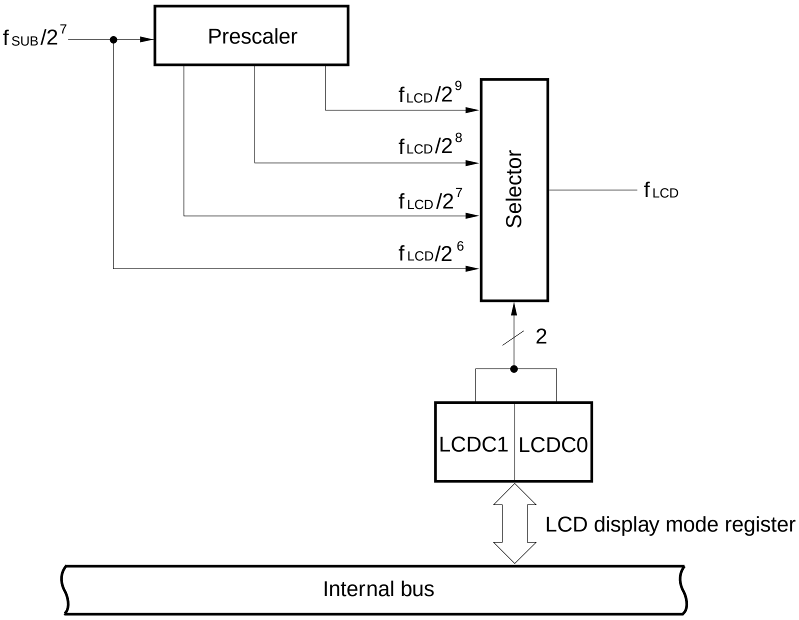
Item	Configuration
Display outputs	Segment signals: 40 Segment signal with alternate function: 40 Common signals: 4 (COM0 to COM3)
Control registers	LCD display mode register (LCDM) LCD display control register (LCDC)

Figure 15-1: LCD Controller/Driver Block Diagram



Note: Segment driver

Figure 15-2: LCD Clock Select Circuit Block Diagram



Remark: f_{LCD}: LCD clock frequency

15.3 LCD Controller/Driver Control Registers

The LCD controller/driver is controlled by the following register.

- LCD display mode register (LCDM)

(1) LCD display mode register (LCDM)

This register sets display operation enabling/disabling, the LCD clock, frame frequency.
The LCDM register can be read or written in 1-bit or 16-bit units.

Figure 15-3: LCD Display Mode Register (LCDM) Format

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	At Reset
LCDM	LCDON	ECOM	ASTOP	LIPS	0	LCDM2	LCDM1	LCDM0	0	0	0	0	0	0	LCDC1	LCDC0	FFFFF710H	0000H

<table>
<tr><th>Bit Name</th><th>Function</th></tr>
<tr><td>LCDON</td><td>Display control bit (Output enable of the displayed data)
0: Display OFF
1: Display ON
If the display is switched off, the non-selected waveform will be output from the segment pin regardless of the contents of the display data memory. If the display is switched on, either the selected waveform or the non-selected waveform will be output from the segment pin according to the contents of the display data memory.</td></tr>
<tr><td>ECOM</td><td>Common enable signal
0: Common disable
1: Common enable
This bit is used for common enabling. Before starting LCD make sure to set bit to 1. And in case application requires low power, this bit should be disabled.</td></tr>
<tr><td>ASTOP</td><td>Driving power supply
0: All analog macro is working
1: LEPSB being used for STOP signal for analog
<b>Note:</b> This is a bit which disable the whole analog macro. (Driving power supply.)</td></tr>
<tr><td>LIPS</td><td>Driving power supply
0: Internal driving power supply OFF
1: Internal driving power supply ON
<b>Note:</b> This is a bit which authorizes/inhibits the common output and the segment output by the internal power supply and the external power supply.</td></tr>
<tr><td>LCDM2,
LCDM1,
LCDM0</td><td>Display mode select bit
<table>
<tr><th>LCDM2</th><th>LCDM1</th><th>LCDM0</th><th>No of time divisions</th><th>Bias method</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>4</td><td>1/3</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>3</td><td>1/3</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2</td><td>1/2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3</td><td>1/2</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td colspan="2">Static</td></tr>
</table></td></tr>
<tr><td>LCDC1,
LCDC0</td><td>Selection of LCD frame frequency
<table>
<tr><th>LCDC1</th><th>LCDC0</th><th>Division of f_{CKSEL1}</th><th>Frame frequency(static)</th></tr>
<tr><td>0</td><td>0</td><td>2^{6} divisions</td><td>488 Hz</td></tr>
<tr><td>0</td><td>1</td><td>2^{7} divisions</td><td>244 Hz</td></tr>
<tr><td>1</td><td>0</td><td>2^{8} divisions</td><td>122 Hz</td></tr>
<tr><td>1</td><td>1</td><td>2^{9} divisions</td><td>61 Hz</td></tr>
</table>
<b>Note:</b> Frequency values if input clock of LCDC section is 31.25 KHz. (4 MHz/2^{7})</td></tr>
</table>

(2) Port LCD segment selector registers 0 to 4 (LSEG0 to LSEG4)

The LSEG0 to LSEG4 registers control the pin function of the corresponding pins.
These registers can be read or written in 1-bit or 8-bit units.

Figure 15-4: Port LCD Segment Selector Registers 0 to 4 (LSEG0 to LSEG4)

	7	6	5	4	3	2	1	0	Address	At Reset
LSEG0	SEGE7	SEGE6	SEGE5	SEGE4	SEGE3	SEGE2	SEGE1	SEGE0	FFFFF700H	00H

	7	6	5	4	3	2	1	0	Address	At Reset
LSEG1	SEGE15	SEGE14	SEGE13	SEGE12	SEGE11	SEGE10	SEGE9	SEGE8	FFFFF702H	00H

	7	6	5	4	3	2	1	0	Address	At Reset
LSEG2	SEGE23	SEGE22	SEGE21	SEGE20	SEGE19	SEGE18	SEGE17	SEGE16	FFFFF704H	00H

	7	6	5	4	3	2	1	0	Address	At Reset
LSEG3	SEGE31	SEGE30	SEGE29	SEGE28	SEGE27	SEGE26	SEGE25	SEGE24	FFFFF706H	00H

	7	6	5	4	3	2	1	0	Address	At Reset
LSEG4	SEGE39	SEGE38	SEGE37	SEGE36	SEGE35	SEGE34	SEGE33	SEGE32	FFFFF708H	00H

Bit Name	Function
SEGEn	SEGn pin function mode 0: Port / control function mode 1: LCD segment mode

Caution: To avoid high cross current, and to ensure that the LCD is working properly, make sure that Port is input mode and port mode before enabling LCD segment mode.

Remark: n = 0 to 15

15.4 LCD Memory Layout

The LCD display data memory is mapped onto addresses 3FFF720H to 3FFF746H. The data stored in the LCD display data memory can be displayed on an LCD panel by the LCD controller/driver.
Table 15-3 shows the relationship between the LCD display data memory contents and the segment outputs/common outputs.
Each of the 160 segments (4 time division mode) is represented by one bit in the segment registers SEGREGxy.

Table 15-3: Relationship between LCD Display Data Memory Contents and Segment/Common Outputs

The SEGREGxy registers are accessible in 1-bit, 8-bit and 16-bit units.

Address	Register Description	Name	Type	1bit	8bit	16bit	Reset
3FFF720	LCD segment register00	SEGREG00	R/W	Y	Y	Y	0000H
22	LCD segment register10	SEGREG10	R/W	Y	Y	Y	0000H
24	LCD segment register20	SEGREG20	R/W	Y	Y	Y	0000H
26	LCD segment register30	SEGREG30	R/W	Y	Y	Y	0000H
3FFF730	LCD segment register01	SEGREG01	R/W	Y	Y	Y	0000H
32	LCD segment register11	SEGREG11	R/W	Y	Y	Y	0000H
34	LCD segment register21	SEGREG21	R/W	Y	Y	Y	0000H
36	LCD segment register31	SEGREG31	R/W	Y	Y	Y	0000H
3FFF740	LCD segment register02	SEGREG30	R/W	Y	Y	Y	0000H
42	LCD segment register12	SEGREG12	R/W	Y	Y	Y	0000H
44	LCD segment register22	SEGREG22	R/W	Y	Y	Y	0000H
46	LCD segment register32	SEGREG32	R/W	Y	Y	Y	0000H

Bit setting	LCD Segment
0	transparent
1	black

Table 15-4: Memory Layout of LCD Segments

	seg39 - seg32	seg31 - seg16	seg15 - seg0
Com0	SEGREG32 FFFFF746$_H$	SEGREG31 FFFFF736$_H$	SEGREG30 FFFFF726$_H$
Com1	SEGREG22 FFFFF744$_H$	SEGREG21 FFFFF734$_H$	SEGREG20 FFFFF724$_H$
Com2	SEGREG12 FFFFF742$_H$	SEGREG11 FFFFF732$_H$	SEGREG10 FFFFF722$_H$
Com3	SEGREG02 FFFFF740$_H$	SEGREG01 FFFFF730$_H$	SEGREG00 FFFFF720$_H$

In 3 time division mode only the registers connected to Com1 - Com3 will be used.
In 2 time division mode only the registers connected to Com2 - Com3 will be used.

15.5 Common Signals and Segment Signals

An individual pixel on an LCD panel lights when the potential difference of the corresponding common signal and segment signal reaches or exceeds a given voltage (the LCD drive voltage VLCD). The light goes off when the potential difference becomes VLCD or lower.
As an LCD panel deteriorates if a DC voltage is applied in the common signals and segment signals, it is driven by AC voltage.

(1) Common signals

For common signals, the selection timing order is as shown in Table 15-5, and operations are repeated with these as the cycle.

Table 15-5: COM Signals

COM Signal Time Division	COM0	COM1	COM2	COM3
4-Time Division				

(2) Segment signals

Segment signals correspond to a 40-byte LCD display data memory (3FFF720H to 3FFF746H). Each display data memory bit 0, bit 1, bit 2, and bit 3 is read in synchronization with the COM0, COM1, COM2 and COM3 timings respectively, and if the value of the bit is 1, it is converted to the selection voltage. If the value of the bit is 0, it is converted to the non-selection voltage and output to a segment pin (S0 to S39) (S39 to S0 have an alternate function as input/output port pins).
Consequently, it is necessary to check what combination of front surface electrodes (corresponding to the segment signals) and rear surface electrodes (corresponding to the common signals) of the LCD panel to be used form the display pattern, and then write bit data corresponding on a one-to-one basis with the pattern to be displayed.

(3) Common signal and segment signal output waveforms

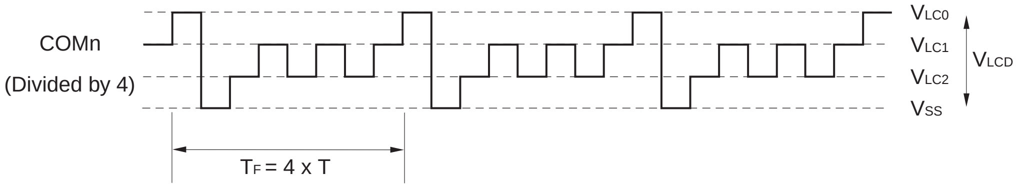
The voltages shown in Table 15-6 are output in the common signals and segment signals.
The ±VLCD ON voltage is only produced when the common signal and segment signal are both at the selection voltage; other combinations produce the OFF voltage.

Table 15-6: LCD Drive Voltage

Common	Segment	Select Level	Non-Select Level
		V_{SS1}, V_{LC0}	V_{LC1}, V_{LC2}
Select Level	V_{LC0}, V_{SS1}	$-V_{LCD}$, $+V_{LCD}$	$-1/3\ V_{LCD}$, $+1/3\ V_{LCD}$
Non-Select Level	V_{LC2}, V_{LC1}	$-1/3\ V_{LCD}$, $+1/3\ V_{LCD}$	$-1/3\ V_{LCD}$, $+1/3\ V_{LCD}$

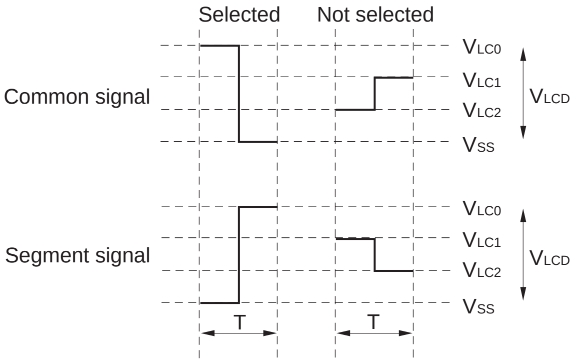
“Common Signal Waveform” on page 519 shows the common signal waveform, and “Common Signal and Segment Signal Voltages and Phases” on page 519 shows the common signal and segment signal voltages and phases.

Figure 15-5: Common Signal Waveform



T: One LCD clock cycle
TF: Frame frequency

Figure 15-6: Common Signal and Segment Signal Voltages and Phases



T: One LCD clock cycle

15.6 Supplying LCD Drive Voltage V_{LC0}, V_{LC1}, and V_{LC2}

The µPD703123 Subseries have a split resistor to create an LCD drive voltage, and the drive voltage is fixed to 1/3 bias.
To supply various LCD drive voltages, internal V_{DD} or external V_{LCD} supply voltage can be selected.

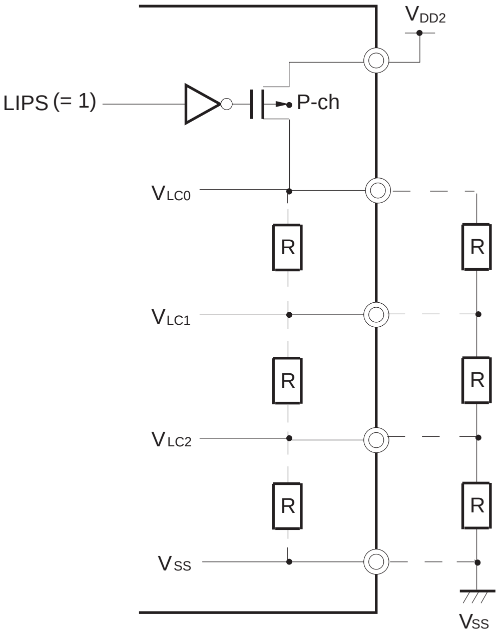
Table 15-7: Drive Voltage Supply

Bias Method LCD Drive Voltage	1/3 Bias Method
V_{LC0}	V_{LC0}
V_{LC1}	2/3 V_{LC0}
V_{LC2}	1/3 V_{LC0}

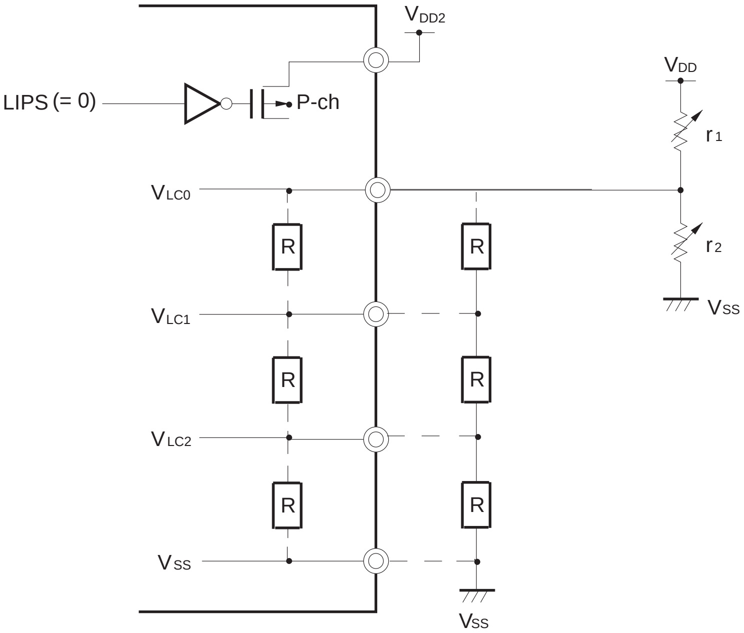
“Example of Connection of LCD Drive Power Supply” on page 521 shows an example of supplying an LCD drive voltage from an internal source according to “Drive Voltage Supply” on page 520 By using variable resistors r1 and r2, a non-stepwise LCD drive voltage can be supplied.

Figure 15-7: Example of Connection of LCD Drive Power Supply

(a) To supply LCD drive voltage from V_{DD}



(b) To supply LCD drive voltage from external source



15.7 Display Mode

15.7.1 4-time-division display example

Figure 15-9 shows the connection of a 4-time-division type 10-digit LCD panel with the display pattern shown in Figure 15-8 with the µPD703123 Subseries segment signals (S0 to S19) and common signals (COM0 to COM3). The display example is “1234567890,” and the display data memory contents (addresses FA59H to FA6CH) correspond to this.
An explanation is given here taking the example of the 5th digit “6” (). In accordance with the display pattern in Figure 15-8, selection and non-selection voltages must be output to pins S8 and S9 as shown in Table 15-8 at the COM0 to COM3 common signal timings.

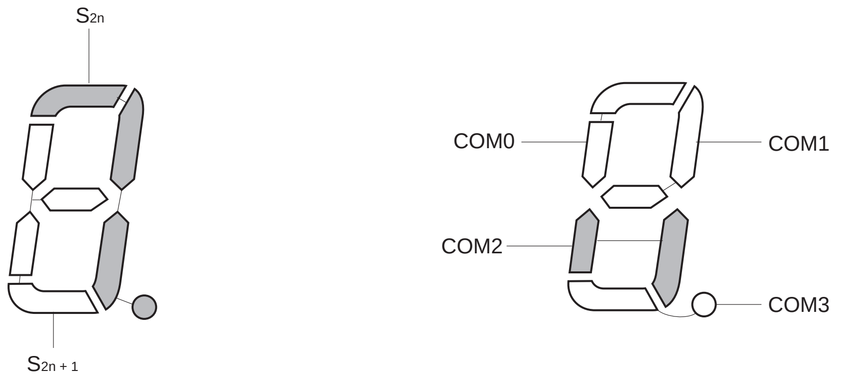
Table 15-8: Selection and Non-Selection Voltages (COM0 to COM3)

Segment Common	S8	S9
COM0	S	S
COM1	NS	S
COM2	S	S
COM3	NS	S

S: Selection, NS: Non-selection

From this, it can be seen that 0101 must be prepared in the display data memory (address H) corresponding to S8.
Examples of the LCD drive waveforms between S8 and the COM0 and COM1 signals are shown in Figure 15-10 (for the sake of simplicity, waveforms for COM2 and COM3 have been omitted). When S8 is at the selection voltage 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 15-8: 4-Time-Division LCD Display Pattern and Electrode Connections



Remark: n = 0 to 9

Figure 15-9: 4-Time-Division LCD Panel Connection Example

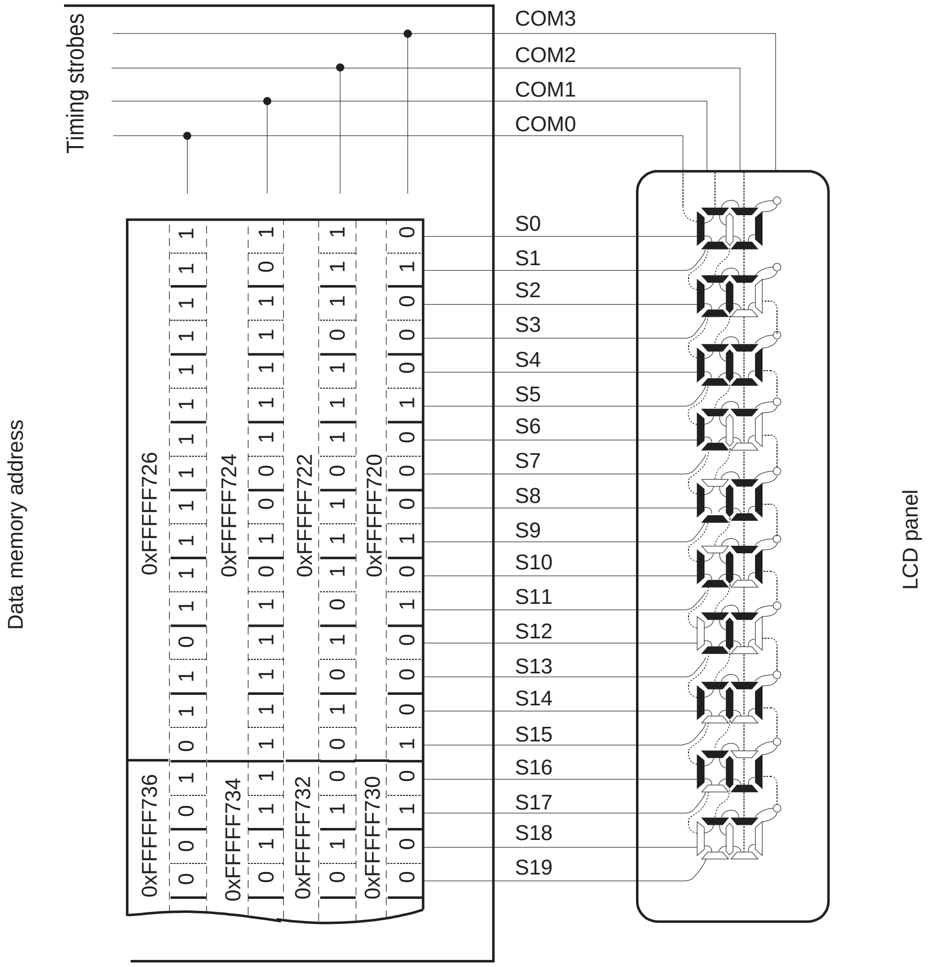
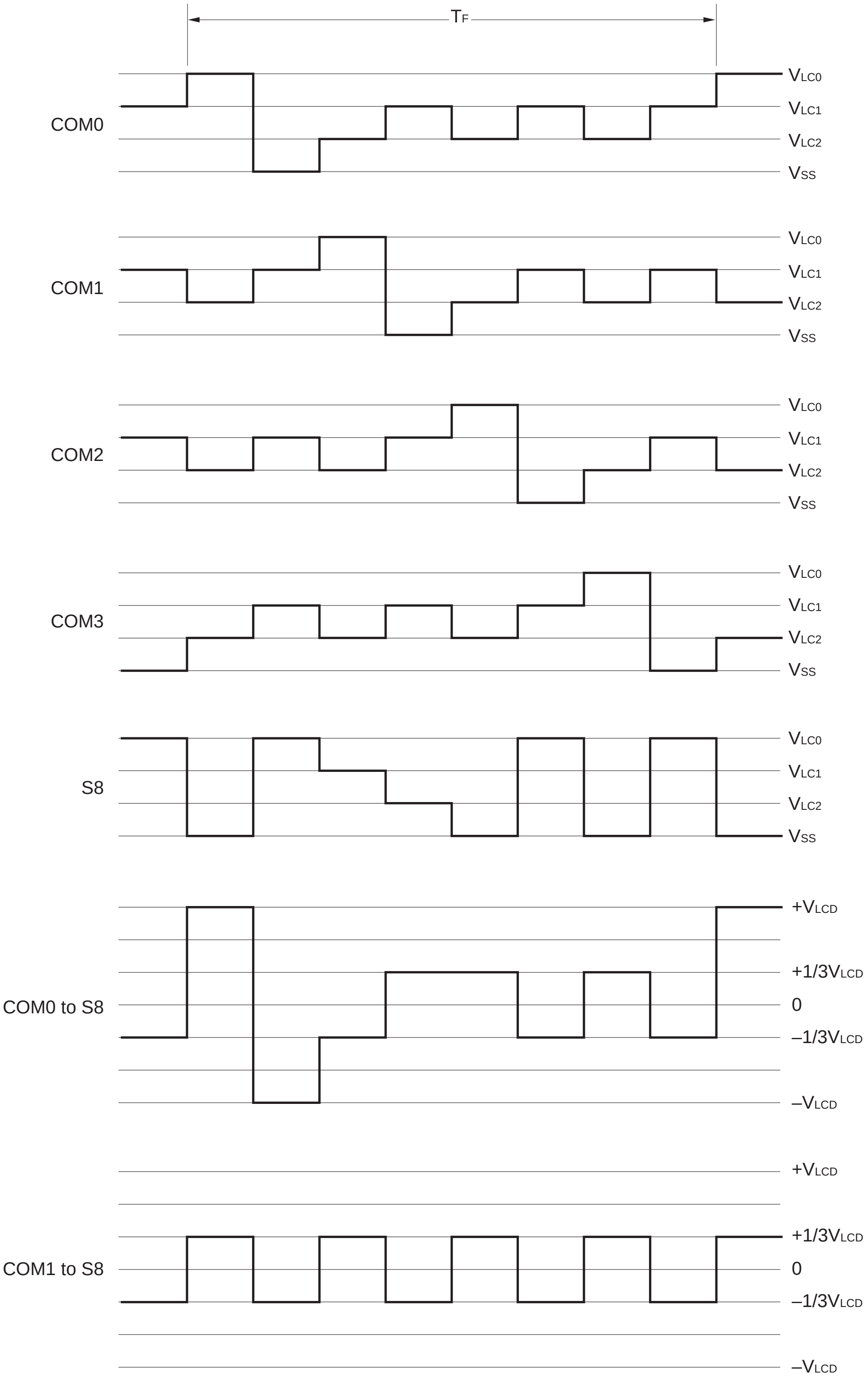


Figure 15-10: 4-Time-Division LCD Drive Waveform Examples (1/3 Bias Method)



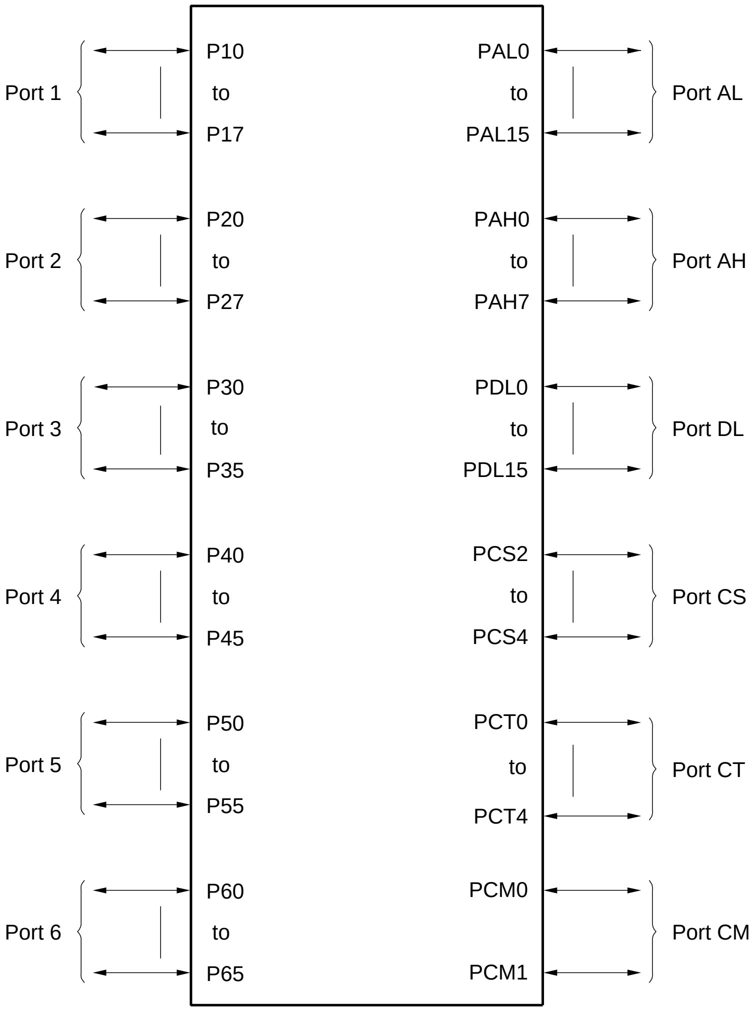
Chapter 16 Port Functions

16.1 Features

- Input/Output ports: 90
- Ports alternate as input/output pins of other peripheral functions
- Input or output can be specified in bit units

16.2 Port Configuration

The V850E/CA1 incorporates a total of 90 input/output ports, named ports P1 through P6, and PAL, PAH, PDL, PCS, PCT and PCM. The configuration is shown below.



(1) Functions of each port

The V850E/CA1 has the ports shown below.
Any port can operate in 8- or 1-bit units and can provide a variety of controls.
Moreover, besides its function as a port, each has functions as the input/output pins of on-chip peripheral I/O in control mode.
Refer to “(3) Port block diagrams” for a block diagram of the block type of each port.

Port Name	Pin Name	Port Function	Function In Control Mode	Block Type
Port 1	P10 to P17	8-bit input/output	Serial interface input/output (UART1, FCAN1 to FCAN3)	A
Port 2	P20 to P27	8-bit input/output	Serial interface input/output (CSI0, CSI1, UART0)	A
Port 3	P30 to P35	6-bit input/output	Real-time pulse unit (RPU) input/output External interrupt input	A
Port 4	P40 to P45	6-bit input/output	Real-time pulse unit (RPU) input/output External interrupt input	A
Port 5	P50 to P55	6-bit input/output	Real-time pulse unit (RPU) input/output External interrupt input	A
Port 6	P60 to P65	6-bit input/output	Serial interface input/output (UART2) External interrupt input External CAN clock	B
Port AL	PAL0 to PAH15	16-bit input/output	External address bus (A0 to A15)	C
Port AH	PAH0 to PAH7	8-bit input/output	External address bus (A16 to A23)	C
Port DL	PDL0 to PDL15	16-bit input/output	External address data bus (D0 to D15)	D
Port CS	PCS2 to PCS4	3-bit input/output	External bus interface control signal output	E
Port CT	PCT0 to PCT4	5-bit input/output	External bus interface control signal output	F
Port CM	PCM0 to PCM1	2-bit input/output	Wait insertion signal input Internal system clock output External bus interface control signal input/output	G, H

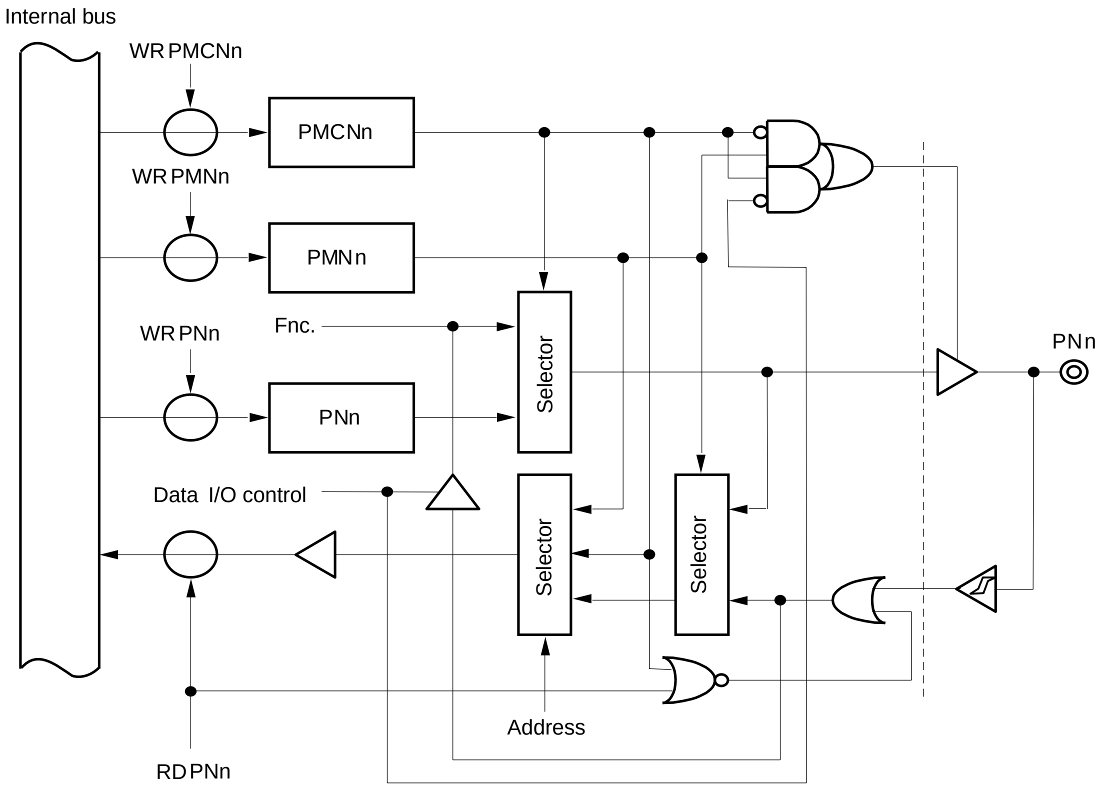
(2) Functions of each port pin on reset and registers that set port or control mode

Port Name	Pin Name	Pin Function after Reset	Mode-Setting Register
		Single-Chip Mode	
Port 1	P10/CRXD1	P10 (Input mode)	PMC1
	P11/CTXD1	P11 (Input mode)	
	P12/CRXD2	P12 (Input mode)	
	P13/CTXD2	P13 (Input mode)	
	P14/CRXD3	P14 (Input mode)	
	P15/CTXD3	P15 (Input mode)	
	P16/RXD1	P16 (Input mode)	
	P17/TXD1	P17 (Input mode)	
Port 2	P20/SI0	P20 (Input mode)	PMC2
	P21/SO0	P21 (Input mode)	
	P22/$\overline{\text{SCK0}}$	P22 (Input mode)	
	P23/SI1	P23 (Input mode)	
	P24/SO1	P24 (Input mode)	
	P25/$\overline{\text{SCK1}}$	P25 (Input mode)	
	P26/RXD0	P26 (Input mode)	
	P27/TXD0	P27 (Input mode)	
Port 3	P30/TIE0/INTPE00	P30 (Input mode)	PMC3
	P31/TOE10/INTPE10	P31 (Input mode)	
	P32/TOE20/INTPE20	P32 (Input mode)	
	P33/TOE30/INTPE30	P33 (Input mode)	
	P34/TOE40/INTPE40	P34 (Input mode)	
	P35/TCLRE0/INTPE50	P35 (Input mode)	
Port 4	P40/TIE1/INTPE01	P40 (Input mode)	PMC4
	P41/TOE11/INTPE11	P41 (Input mode)	
	P42/TOE21/INTPE21	P42 (Input mode)	
	P43/TOE31/INTPE31	P43 (Input mode)	
	P44/TOE41/INTPE41	P44 (Input mode)	
	P45/TCLRE1/INTPE51	P45 (Input mode)	
Port 5	P50/TIE2/INTPE02	P50 (Input mode)	PMC5
	P51/TOE12/INTPE12	P51 (Input mode)	
	P52/TOE22/INTPE22	P52 (Input mode)	
	P53/TOE32/INTPE32	P53 (Input mode)	
	P54/TOE42/INTPE42	P54 (Input mode)	
	P55/TCLRE2/INTPE52	P55 (Input mode)	

Port Name	Pin Name	Pin Function after Reset	Mode-Setting Register
		Single-Chip Mode	
Port 6	P60/CCLK/SEG34	P60 (Input mode)	PMC6
	P61/INT0/SEG35	P61 (Input mode)	
	P62/INT1/SEG36	P62 (Input mode)	
	P63/INT2/SEG37	P63 (Input mode)	
	P64/RXD2/SEG38	P64 (Input mode)	
	P65/TXD2/SEG39	P65 (Input mode)	
Port CM	PCM0/$\overline{\text{WAIT}}$/SEG32	PCM0 (Input mode)	PMCCM
	PCM1/CLKOUT/SEG33	PCM1 (Input mode)	
Port CT	PCT0/$\overline{\text{LWR}}$/SEG27	PCT0 (Input mode)	PMCCT
	PCT1/$\overline{\text{UWR}}$/SEG28	PCT1 (Input mode)	
	PCT2/SEG29	PCT2 (Input mode)	
	PCT3/SEG30	PCT3 (Input mode)	
	PCT4/$\overline{\text{RD}}$/SEG31	PCT4 (Input mode)	
Port CS	PCS2/$\overline{\text{CS2}}$/SEG24 to PCS4/$\overline{\text{CS4}}$/SEG26	PCS2 to PCS4 (Input mode)	PMCCS
Port DL	PDL0/D0 to PDL15/D15	PDL0 to PDL15 (Input mode)	PMCDL
Port AL	PAL0/A0/SEG23 to PAL15/A15/SEG8	PAL0 to PAL15 (Input mode)	PMCAL
Port AH	PAH0/A16/SEG0 to PAH7/A23/SEG7	PAH0 to PAH7 (Input mode)	PMCAH

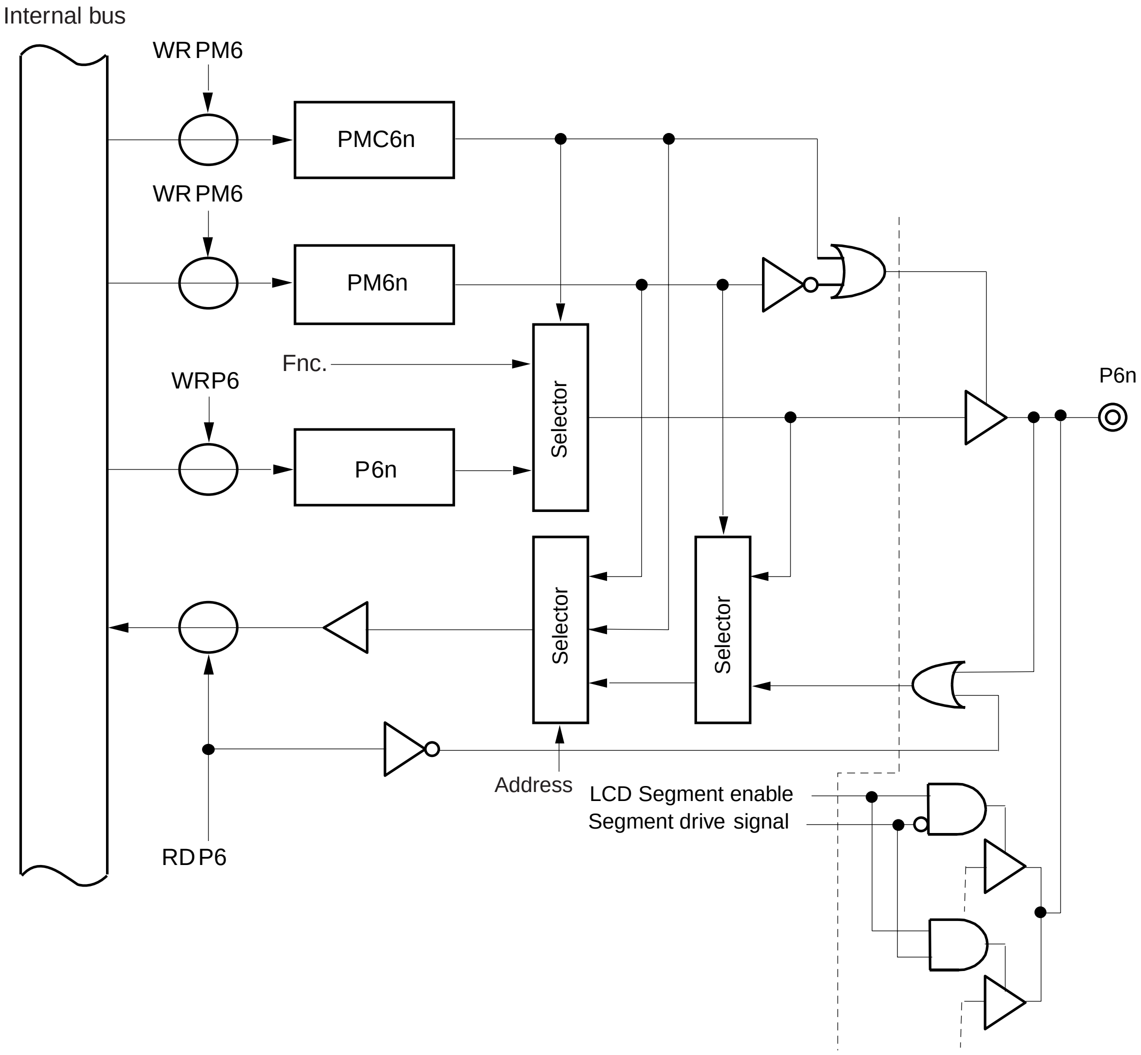
(3) Port block diagrams

Figure 16-1: Type A Block Diagram



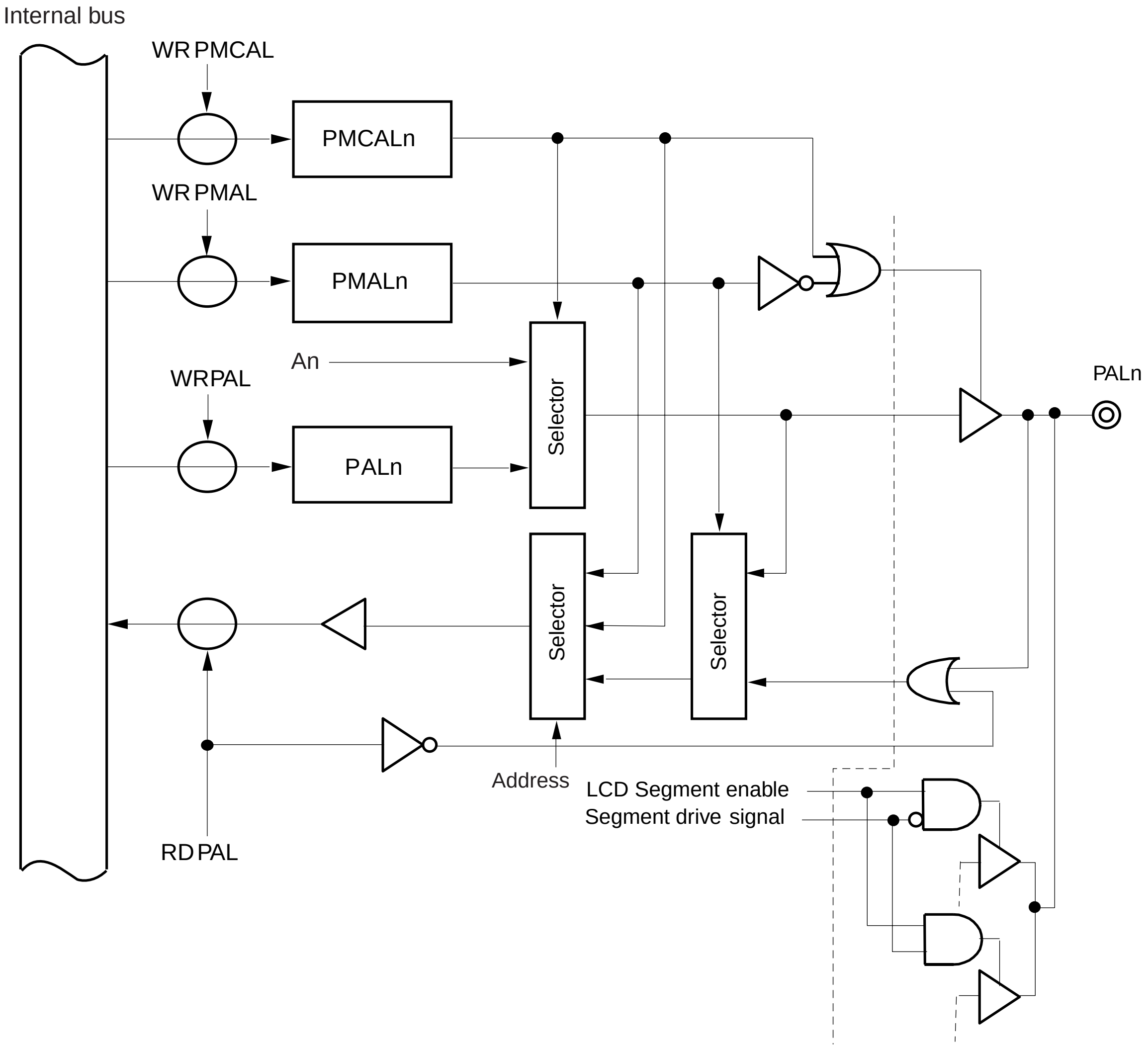
Remark: N = 1 to 5 : Port number
n = 0 to 7 : Port pin for port number 0 and 1
n = 0 to 5 : Port pin for port number 2 to 5.

Figure 16-2: Type B Block Diagram



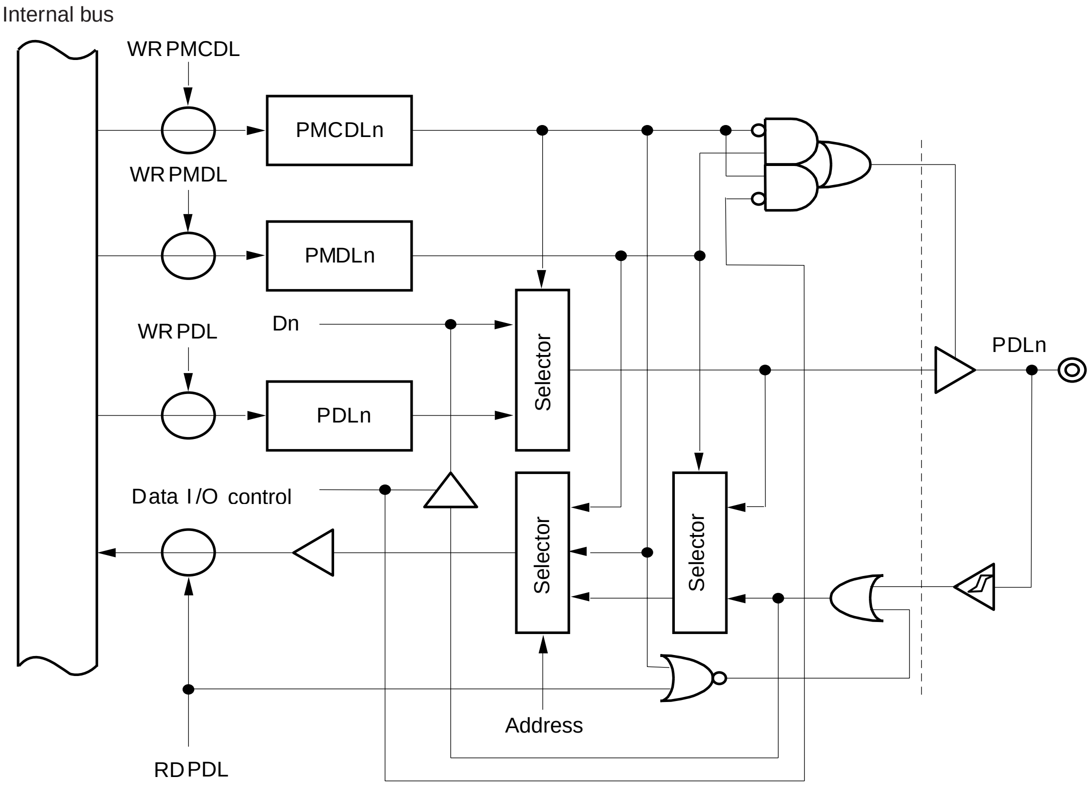
Remark: n = 0 to 5

Figure 16-3: Type C Block Diagram



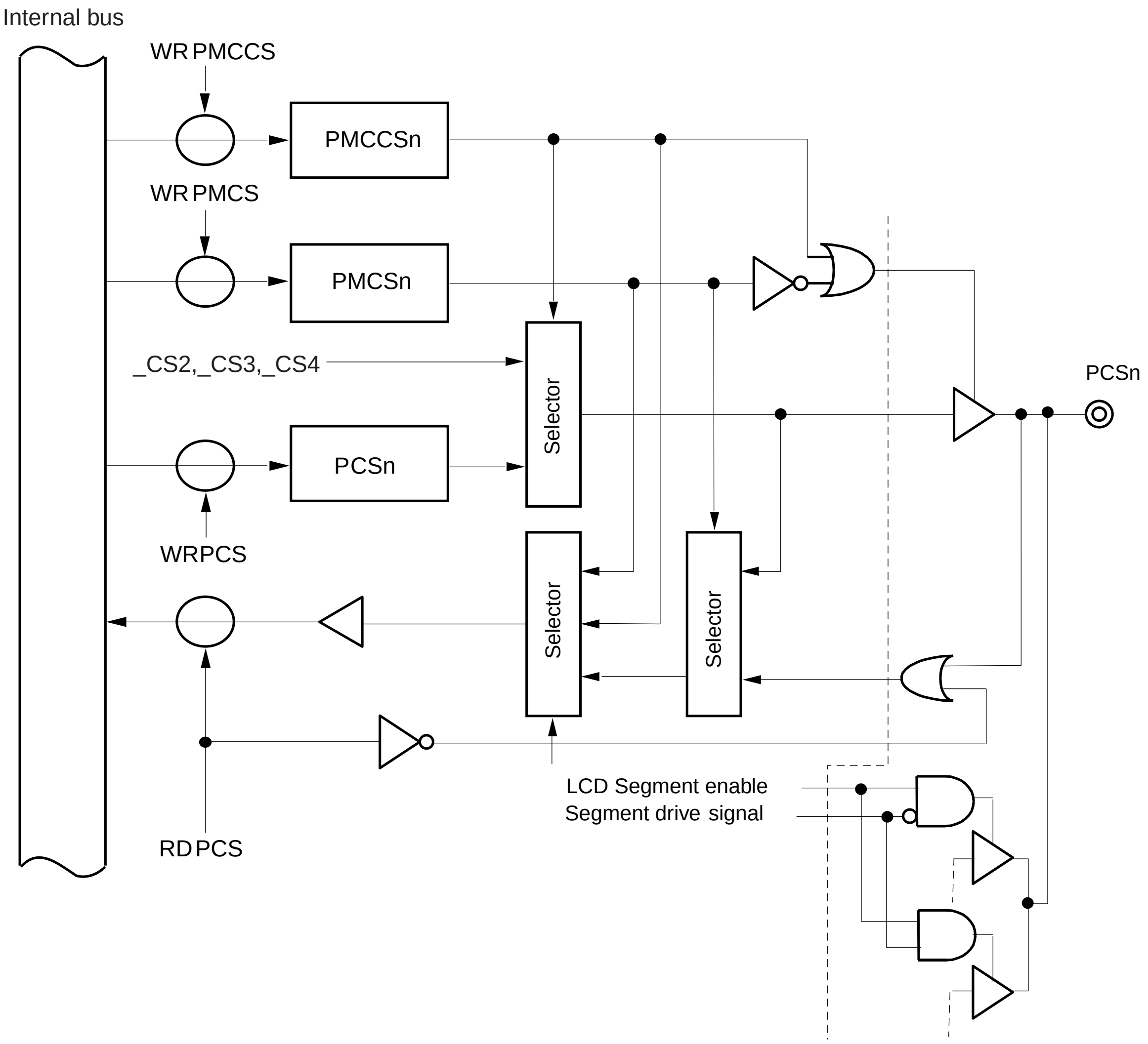
Remark: n = 0 to 15

Figure 16-4: Type D Block Diagram



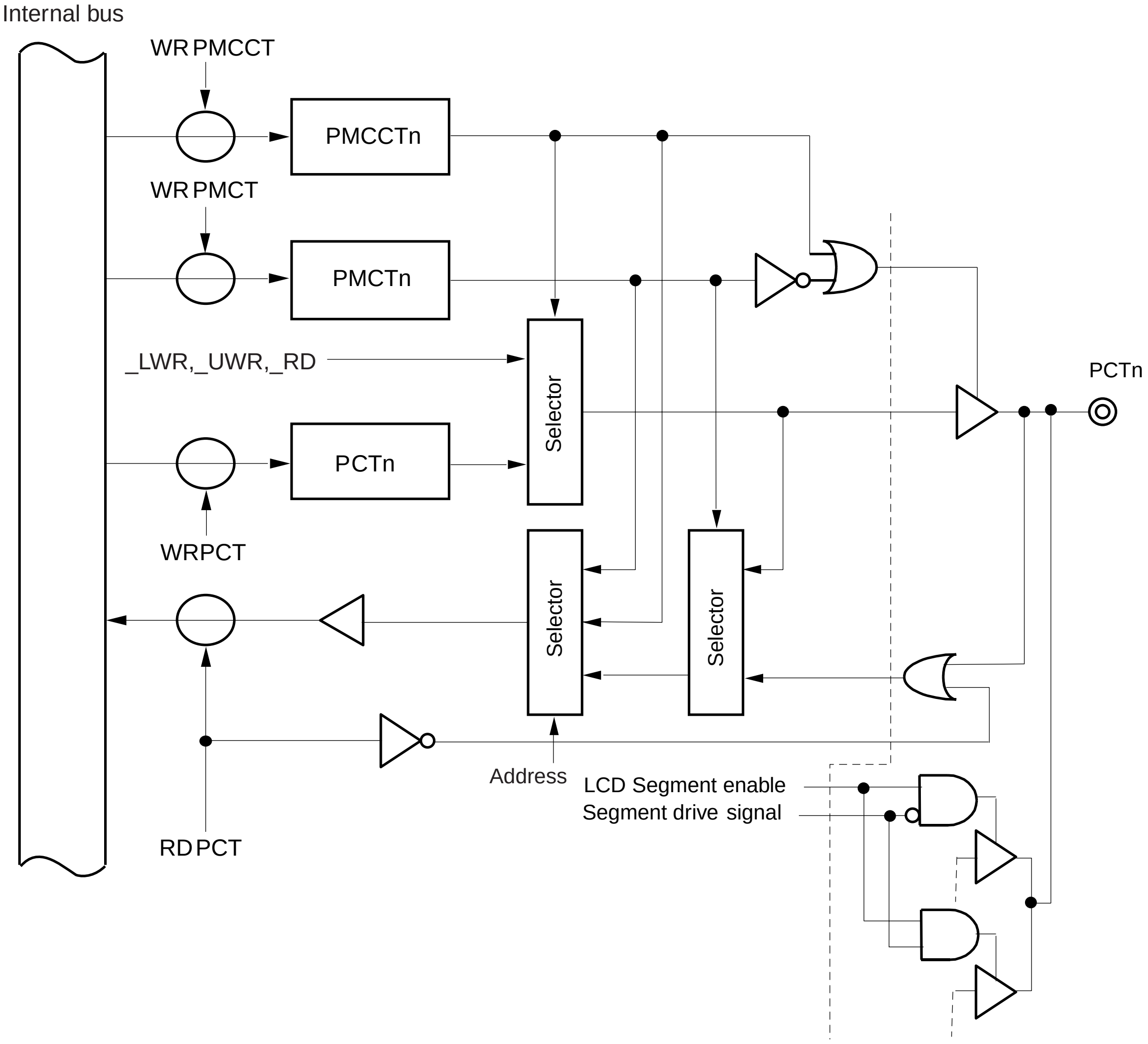
Remark: n = 0 to 15

Figure 16-5: Type E Block Diagram



Remark: n = 2, 3, 4

Figure 16-6: Type F Block Diagram



Remark: n = 0 to 4

Figure 16-7: Type G Block Diagram

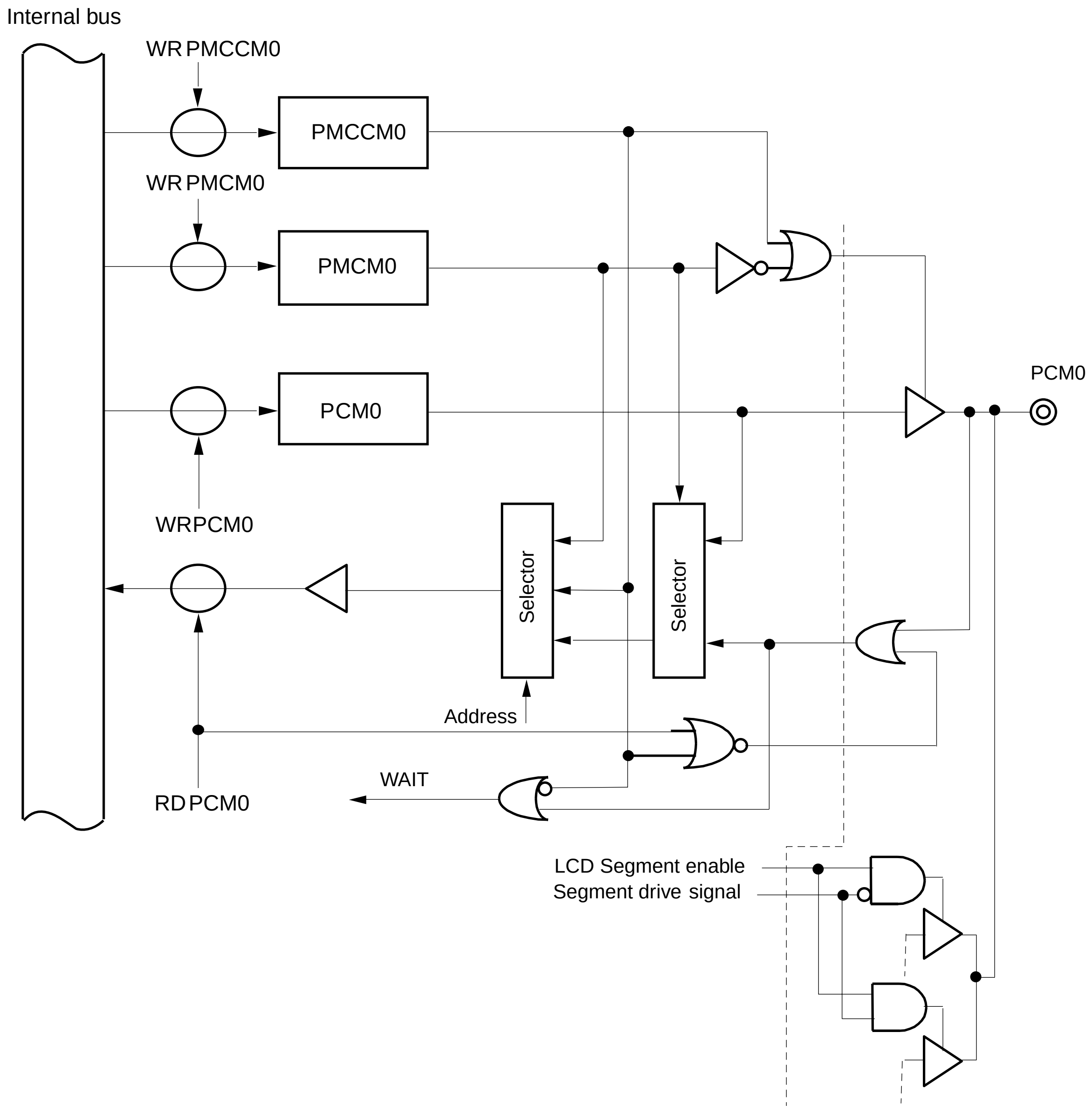
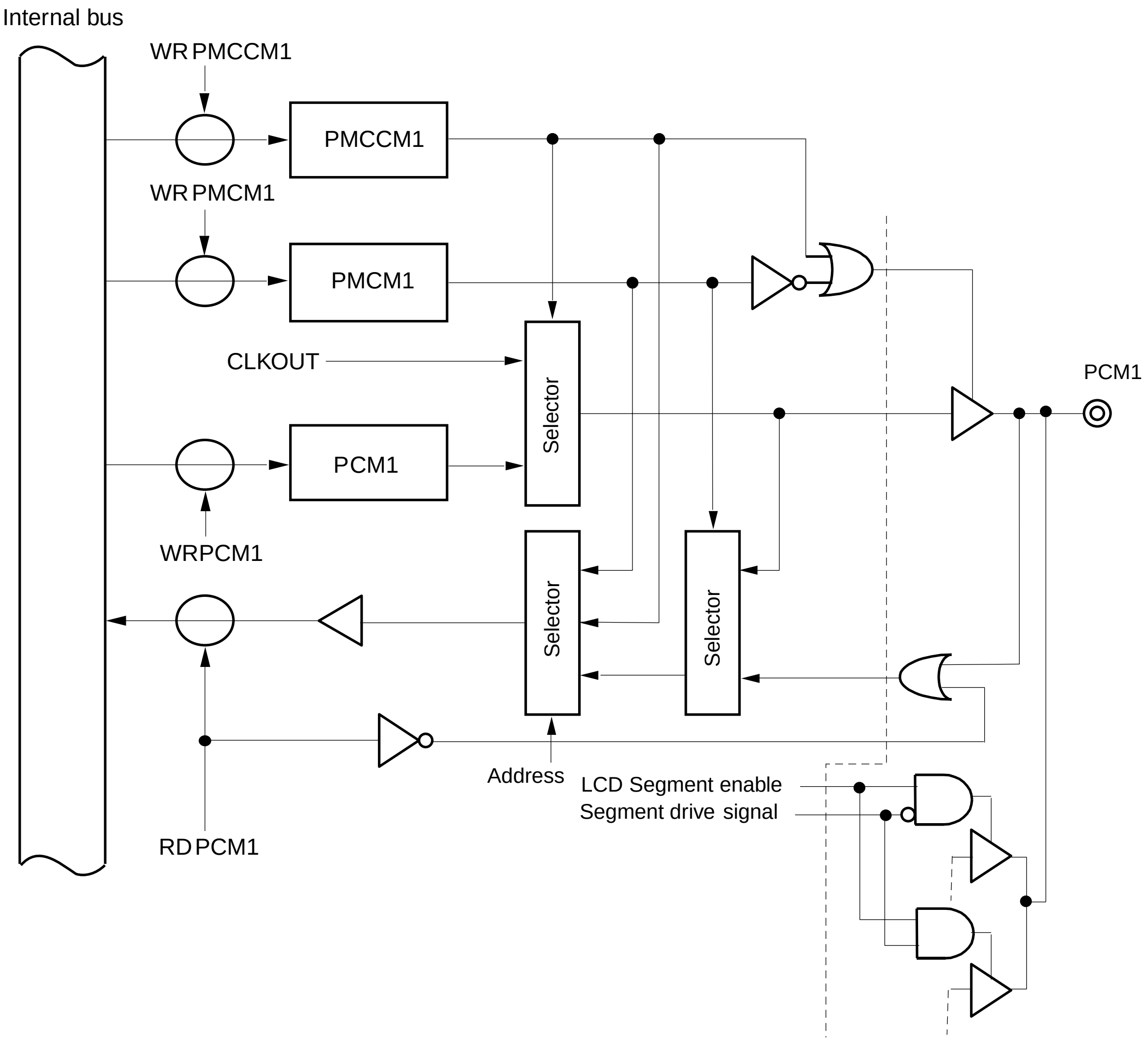


Figure 16-8: Type H Block Diagram



16.3 Pin Functions of Each Port

16.3.1 Port 1

Port 1 is a 8-bit input/output port in which input or output can be specified in 1-bit units.

Figure 16-9: Port 1 (P1)

	7	6	5	4	3	2	1	0	Address	At Reset
P1	P17	P16	P15	P14	P13	P12	P11	P10	FFFFF400H	00H

Bit position	Bit name	Function
7 to 0	P1n (n = 7 to 0)	Input/output port

Besides functioning as a port, in control mode, it also can operate as the serial interface (UART1, FCAN1, FCAN2, FCAN3) input/output.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port 1	P10	CRXD1	Serial interface (UART1, FCAN1, FCAN2, FCAN3) input/output.	A
	P11	CTXD1		
	P12	CRXD2		
	P13	CTXD2		
	P14	CRXD3		
	P15	CTXD3		
	P16	RXD1		
	P17	TXD1		

(2) Setting in input/output mode and control mode

Port 1 is set in input/output mode using the port 1 mode register (PM1). In control mode, it is set using the port 1 mode control register (PMC1).

(a) Port 1 mode register (PM1)

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

Figure 16-10: Port 1 Mode Register (PM1)

	7	6	5	4	3	2	1	0	Address	At Reset
PM1	PM17	PM16	PM15	PM14	PM13	PM12	PM11	PM10	FFFFF420H	FFH

Bit Position	Bit Name	Function
7 to 0	PM1n (n = 7 to 0)	Specifies input/output mode of P1n pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port 1 mode control register (PMC1)

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

Figure 16-11: Port 1 Mode Control Register (PMC1)

	7	6	5	4	3	2	1	0	Address	At Reset
PMC1	PMC17	PMC16	PMC15	PMC14	PMC13	PMC12	PMC11	PMC10	FFFFF44CH	00H

Bit Position	Bit Name	Function
7	PMC17	Specifies operation mode of P17 pin 0: Input/output port mode 1: TXD1 output mode
6	PMC16	Specifies operation mode of P16 pin 0: Input/output port mode 1: RXD1 input mode
5	PMC15	Specifies operation mode of P15 pin 0: Input/output port mode 1: CTXD3 output mode
4	PMC14	Specifies operation mode of P14 pin 0: Input/output port mode 1: CRXD3 input mode
3	PMC13	Specifies operation mode of P13 pin 0: Input/output port mode 1: CTXD2 output mode
2	PMC12	Specifies operation mode of P12 pin 0: Input/output port mode 1: CRXD2 input mode
1	PMC11	Specifies operation mode of P11 pin 0: Input/output port mode 1: CTXD1 output mode
0	PMC10	Specifies operation mode of P10 pin 0: Input/output port mode 1: CRXD1 input mode

16.3.2 Port 2

Port 2 is a 8-bit input/output port in which input or output can be specified in 1-bit units.

Figure 16-12: Port 2 (P2)

	7	6	5	4	3	2	1	0	Address	At Reset
P2	P27	P26	P25	P24	P23	P22	P21	P20	FFFFF402H	00H

Bit position	Bit name	Function
7 to 0	P2n (n = 7 to 0)	Input/output port

Besides functioning as a port, in control mode, it also can operate as the serial interface (CSI0, CSI1, UART0) input/output.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port 2	P20	SI0	Serial interface (CSI0, CSI1, UART0) input/output.	A
	P21	SO0		
	P22	$\overline{\text{SCK0}}$		
	P23	SI1		
	P24	SO1		
	P25	$\overline{\text{SCK1}}$		
	P26	RXD0		
	P27	TXD0		

(2) Setting in input/output mode and control mode

Port 2 is set in input/output mode using the port 2 mode register (PM2). In control mode, it is set using the port 2 mode control register (PMC2).

(a) Port 2 mode register (PM2)

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

Figure 16-13: Port 2 Mode Register (PM2)

	7	6	5	4	3	2	1	0	Address	At Reset
PM2	PM27	PM26	PM25	PM24	PM23	PM22	PM21	PM20	FFFFF422H	FFH

Bit Position	Bit Name	Function
7 to 0	PM2n (n = 7 to 0)	Specifies input/output mode of P2n pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port 2 mode control register (PMC2)

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

Figure 16-14: Port 2 Mode Control Register (PMC2)

	7	6	5	4	3	2	1	0	Address	At Reset
PMC2	PMC27	PMC26	PMC25	PMC24	PMC23	PMC22	PMC21	PMC20	FFFFF442H	00H

Bit Position	Bit Name	Function
7	PMC27	Specifies operation mode of P27 pin 0: Input/output port mode 1: TXD0 output mode
6	PMC26	Specifies operation mode of P26 pin 0: Input/output port mode 1: RXD0 input mode
5	PMC25	Specifies operation mode of P25 pin 0: Input/output port mode 1: $\overline{\text{SCK1}}$ input/output mode
4	PMC24	Specifies operation mode of P24 pin 0: Input/output port mode 1: SO1 output mode
3	PMC23	Specifies operation mode of P23 pin 0: Input/output port mode 1: SI1 input mode
2	PMC22	Specifies operation mode of P22 pin 0: Input/output port mode 1: $\overline{\text{SCK0}}$ input/output mode
1	PMC21	Specifies operation mode of P21 pin 0: Input/output port mode 1: SO0 output mode
0	PMC20	Specifies operation mode of P20 pin 0: Input/output port mode 1: SI0 input mode

16.3.3 Port 3

Port 3 is a 6-bit input/output port in which input or output can be specified in 1-bit units.

Figure 16-15: Port 3 (P3)

	7	6	5	4	3	2	1	0	Address	At Reset
P3	0	0	P35	P34	P33	P32	P31	P30	FFFFF404H	00H

Bit position	Bit name	Function
5 to 0	P3n (n = 5 to 0)	Input/output port

Besides functioning as a port, in control mode, it also can operate as the real-time pulse unit (RPU) input/output and external interrupt request input.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port 3	P30	TIE0/INTPE00	Real-time pulse unit (RPU) input or external interrupt request input	A
	P31	TOE10/INTPE10		
	P32	TOE20/INTPE20		
	P33	TOE30/INTPE30		
	P34	TOE40/INTPE40		
	P35	TCLRE0/INTPE50		

(2) Setting in input/output mode and control mode

Port 3 is set in input/output mode using the port 3 mode register (PM3). In control mode, it is set using the port 3 mode control register (PMC3).

(a) Port 3 mode register (PM3)
This register can be read or written in 8- or 1-bit units.

Figure 16-16: Port 3 Mode Register (PM3)

	7	6	5	4	3	2	1	0	Address	At Reset
PM3	1	1	PM35	PM34	PM33	PM32	PM31	PM30	FFFFF424H	FFH

Bit Position	Bit Name	Function
5 to 0	PM3n (n = 5 to 0)	Specifies input/output mode of P3n pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port 3 mode control register (PMC3)

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

Figure 16-17: Port 3 Mode Control Register (PMC3)

	7	6	5	4	3	2	1	0	Address	At Reset
PMC3	0	0	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30	FFFFF444H	00H

Bit Position	Bit Name	Function
5	PMC35	Specifies operation mode of P35 pin 0: Input/output port mode 1:TCLRE0 input mode or external interrupt request (INTP50) input mode
4	PMC34	Specifies operation mode of P34 pin 0: Input/output port mode 1: TOE40 input/output mode or external interrupt request (INTPE40) input mode
3	PMC33	Specifies operation mode of P33 pin 0: Input/output port mode 1: TOE30 input/output mode or external interrupt request (INTPE30) input mode
2	PMC32	Specifies operation mode of P32 pin 0: Input/output port mode 1: TOE20 input/output mode or external interrupt request (INTPE20) input mode
1	PMC31	Specifies operation mode of P31 pin 0: Input/output port mode 1: TOE10 input/output mode or external interrupt request (INTPE10) input mode
0	PMC30	Specifies operation mode of P30 pin 0: Input/output port mode 1: TIE0 input mode or external interrupt request (INTPE00) input mode

16.3.4 Port 4

Port 4 is a 6-bit input/output port in which input or output can be specified in 1-bit units.

Figure 16-18: Port 4 (P4)

	7	6	5	4	3	2	1	0	Address	At Reset
P4	0	0	P45	P44	P43	P42	P41	P40	FFFFF406H	00H

Bit position	Bit name	Function
5 to 0	P4n (n = 5 to 0)	Input/output port

Besides functioning as a port, in control mode, it also can operate as the real-time pulse unit (RPU) input/output and external interrupt request input.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port 4	P40	TIE1/INTPE01	Real-time pulse unit (RPU) input or external interrupt request input	A
	P41	TOE11/INTPE11		
	P42	TOE21/INTPE21		
	P43	TOE31/INTPE31		
	P44	TOE41/INTPE41		
	P45	TCLRE1/INTPE51		

(2) Setting in input/output mode and control mode

Port 4 is set in input/output mode using the port 4 mode register (PM4). In control mode, it is set using the port 4 mode control register (PMC4).

(a) Port 4 mode register (PM4)

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

Figure 16-19: Port 4 Mode Register (PM4)

	7	6	5	4	3	2	1	0	Address	At Reset
PM4	1	1	PM45	PM44	PM43	PM42	PM41	PM40	FFFFF426H	FFH

Bit Position	Bit Name	Function
5 to 0	PM4n (n = 5 to 0)	Specifies input/output mode of P4n pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port 4 mode control register (PMC4)

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

Figure 16-20: Port 4 Mode Control Register (PMC4)

	7	6	5	4	3	2	1	0	Address	At Reset
PMC4	0	0	PMC45	PMC44	PMC43	PMC42	PMC41	PMC40	FFFFF446H	00H

Bit Position	Bit Name	Function
5	PMC45	Specifies operation mode of P45 pin 0: Input/output port mode 1: TCLRE1 input mode or external interrupt request (INTP51) input mode
4	PMC44	Specifies operation mode of P44 pin 0: Input/output port mode 1: TOE41 input/output mode or external interrupt request (INTPE41) input mode
3	PMC43	Specifies operation mode of P43 pin 0: Input/output port mode 1: TOE31 input/output mode or external interrupt request (INTPE31) input mode
2	PMC42	Specifies operation mode of P42 pin 0: Input/output port mode 1: TOE21 input/output mode or external interrupt request (INTPE21) input mode
1	PMC41	Specifies operation mode of P41 pin 0: Input/output port mode 1: TOE11 input/output mode or external interrupt request (INTPE11) input mode
0	PMC40	Specifies operation mode of P40 pin 0: Input/output port mode 1: TIE1 input mode or external interrupt request (INTPE01) input mode

16.3.5 Port 5

Port 5 is a 6-bit input/output port in which input or output can be specified in 1-bit units.

Figure 16-21: Port 5 (P5)

	7	6	5	4	3	2	1	0	Address	At Reset
P5	0	0	P55	P54	P53	P52	P51	P50	FFFFF408H	00H

Bit position	Bit name	Function
5 to 0	P5n (n = 5 to 0)	Input/output port

Besides functioning as a port, in control mode, it also can operate as the real-time pulse unit (RPU) input/output and external interrupt request input.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port 5	P50	TIE2/INTPE02	Real-time pulse unit (RPU) input or external interrupt request input	A
	P51	TOE12/INTPE12		
	P52	TOE22/INTPE22		
	P53	TOE32/INTPE32		
	P54	TOE42/INTPE42		
	P55	TCLRE2/INTPE52		

(2) Setting in input/output mode and control mode

Port 5 is set in input/output mode using the port 5 mode register (PM5). In control mode, it is set using the port 5 mode control register (PMC5).

(a) Port 5 mode register (PM5)

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

Figure 16-22: Port 5 Mode Register (PM5)

	7	6	5	4	3	2	1	0	Address	At Reset
PM5	1	1	PM55	PM54	PM53	PM52	PM51	PM50	FFFFF428H	FFH

Bit Position	Bit Name	Function
5 to 0	PM5n (n = 5 to 0)	Specifies input/output mode of P5n pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port 5 mode control register (PMC5)

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

Figure 16-23: Port 5 Mode Control Register (PMC5)

	7	6	5	4	3	2	1	0	Address	At Reset
PMC5	0	0	PMC55	PMC54	PMC53	PMC52	PMC51	PMC50	FFFFF448H	00H

Bit Position	Bit Name	Function
5	PMC55	Specifies operation mode of P55 pin 0: Input/output port mode 1: TCLR2 input mode or external interrupt request (INTP52) input mode
4	PMC54	Specifies operation mode of P54 pin 0: Input/output port mode 1: TOE42 input/output mode or external interrupt request (INTPE42) input mode
3	PMC53	Specifies operation mode of P53 pin 0: Input/output port mode 1: TOE32 input/output mode or external interrupt request (INTPE32) input mode
2	PMC52	Specifies operation mode of P52 pin 0: Input/output port mode 1: TOE22 input/output mode or external interrupt request (INTPE22) input mode
1	PMC51	Specifies operation mode of P51 pin 0: Input/output port mode 1: TOE12 input/output mode or external interrupt request (INTPE12) input mode
0	PMC50	Specifies operation mode of P50 pin 0: Input/output port mode 1: TIE2 input mode or external interrupt request (INTPE02) input mode

16.3.6 Port 6

Port 6 is a 6-bit input/output port in which input or output can be specified in 1-bit units.

Figure 16-24: Port 6 (P6)

	7	6	5	4	3	2	1	0	Address	At Reset
P6	0	0	P65	P64	P63	P62	P61	P60	FFFFF40AH	00H

Bit position	Bit name	Function
5 to 0	P6n (n = 5 to 0)	Input/output port

Besides functioning as a port, in control mode, it also can operate as segment signal output of LCD controller/driver, external CAN clock supply, serial interface (UART2) input/output and external interrupt request input.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port 6	P60	CCLK/SEG34	External CAN clock supply or segment signal output of LCD-controller	B
	P61	INT0/SEG35	External interrupt request pins or segment signal output of LCD-controller	
	P62	INT1/SEG36		
	P63	INT2/SEG37		
	P64	RXD2/SEG38	Serial interface (UART2) or segment signal output of LCD-controller	
	P65	TXD2/SEG39		

(2) Setting in input/output mode and control mode

Port 6 is set in input/output mode using the port 6 mode register (PM6). In control mode, it is set using the port 6 mode control register (PMC6).

(a) Port 6 mode register (PM6)

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

Figure 16-25: Port 6 Mode Register (PM6)

	7	6	5	4	3	2	1	0	Address	At Reset
PM6	1	1	PM65	PM64	PM63	PM62	PM61	PM60	FFFFF42AH	FFH

Bit Position	Bit Name	Function
5 to 0	PM6n (n = 5 to 0)	Specifies input/output mode of P6n pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port 6 mode control register (PMC6)

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

Figure 16-26: Port 6 Mode Control Register (PMC6)

	7	6	5	4	3	2	1	0	Address	At Reset
PMC6	0	0	PMC65	PMC64	PMC63	PMC62	PMC61	PMC60	FFFFF44AH	00H

Bit Position	Bit Name	Function
5	PMC65	Specifies operation mode of P65 pin 0: Input/output port mode or segment signal output of LCD-controller mode 1:TXD output mode
4	PMC64	Specifies operation mode of P64 pin 0: Input/output port mode or segment signal output of LCD-controller mode 1: RXD input mode
3	PMC63	Specifies operation mode of P63 pin 0: Input/output port mode or segment signal output of LCD-controller mode 1: External interrupt request (INT2) input mode
2	PMC62	Specifies operation mode of P62 pin 0: Input/output port mode or segment signal output of LCD-controller mode 1: External interrupt request (INT1) input mode
1	PMC61	Specifies operation mode of P61 pin 0: Input/output port mode or segment signal output of LCD-controller mode 1: External interrupt request (INT0) input mode
0	PMC60	Specifies operation mode of P60 pin 0: Input/output port mode or segment signal output of LCD-controller mode 1: External CAN clock supply input mode

Caution: Make sure that Port 6 is input mode and port mode before enabling LCD segment mode!

(To avoid high cross current and to ensure that LCD is working properly).

16.4 Port AL

Port AL is a 16-/8-bit input/output port that can be set the input or output mode in 1-bit units.

Figure 16-27: Port AL (PAL)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	At Reset
PAL	PAL15	PAL14	PAL13	PAL12	PAL11	PAL10	PAL9	PAL8	PAL7	PAL6	PAL5	PAL4	PAL3	PAL2	PAL1	PAL0	FFFFF000H	0000H

Bit position	Bit name	Function
15 to 0	PALn (n = 15 to 0)	Input/output port

In addition to their functions as port pins, in the control mode, the port AL pins operate as an address bus for when the memory is externally expanded. It also can operate as segment signal output of LCD controller/driver.

(1) Operation in control mode

Port		Alternate Function	Remark	Block Type
Port AL	PAL15 to PAL0	A15 to A0/ SEG23 to SEG8	Address bus when memory expanded or segment signal output of LCD-controller	C

(2) Input/output mode control mode setting

Port AL input/output mode setting is performed by means of the port AL mode register (PMAL), and control mode setting is performed by means of the port AL mode control register (PMCAL).

(a) Port AL mode register (PMAL)

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

Figure 16-28: Port AL Mode Register (PMAL)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	At Reset
PMAL	PMAL15	PMAL14	PMAL13	PMAL12	PMAL11	PMAL10	PMAL9	PMAL8	PMAL7	PMAL6	PMAL5	PMAL4	PMAL3	PMAL2	PMAL1	PMAL0	FFFFF020H	FFFFH

Bit Position	Bit Name	Function
15 to 0	PMALn (n = 15 to 0)	Specifies input/output mode of PMALn pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port AL mode control register (PMCAL)

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

Figure 16-29: Port AL Mode Control Register (PMCAL)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	At Reset
PMCAL	PMCAL15	PMCAL14	PMCAL13	PMCAL12	PMCAL11	PMCAL10	PMCAL9	PMCAL8	PMCAL7	PMCAL6	PMCAL5	PMCAL4	PMCAL3	PMCAL2	PMCAL1	PMCAL0	FFFFF040H	0000H

Bit Position	Bit Name	Function
15 to 0	PMCALn (n = 15 to 0)	Port Mode Control Specifies operation mode of PALn pin. 0: I/O port mode or segment signal output in LCD-controller mode 1: A15 to A0 output mode

16.5 Port AH

Port AH is a 16-bit input/output port for which input/output can be specified bitwise.

Figure 16-30: Port AH (PAH)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	At Reset
PAH	-	-	-	-	-	-	-	-	PAH7	PAH6	PAH5	PAH4	PAH3	PAH2	PAH1	PAH0	FFFFF002H	00H

Bit position	Bit name	Function
7 to 0	PAHn (n = 7 to 0)	Input/output port
15-8		Unknown Data

In addition to their functions as port pins, in the control mode, the port AH pins operate as an address bus for when the memory is externally expanded. It also can operate as segment signal output of LCD controller/driver.

(1) Operation in control mode

Port		Alternate Function	Remark	Block Type
Port AH	PAH7 to PAH0	A23 to A16/ SEG7 to SEG0	Address bus when memory expanded or segment signal output of LCD-controller	C

(2) Input/output mode control mode setting

Port AH input/output mode setting is performed by means of the port AH mode register (PMAH), and control mode setting is performed by means of the port AH mode control register (PMCAH).

(a) Port AH mode register (PMAH)

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

Figure 16-31: Port AH Mode Register (PMAH)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addres s	At Reset
PMAH	-	-	-	-	-	-	-	-	PMAH7	PMAH6	PMAH5	PMAH4	PMAH3	PMAH2	PMAH1	PMAH0	FFFFF022H	FFH

Bit Position	Bit Name	Function
7 to 0	PMAHn (n = 7 to 0)	Specifies input/output mode of PMAHn pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port Mode Control Register AH (PMCAH)

PMCAH can be read/written from/to in 16-bit units or bitwise.

Figure 16-32: Port AH Mode Control Register (PMCAH)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	At Reset
PMCAH	-	-	-	-	-	-	-	-	PMCAH7	PMCAH6	PMCAH5	PMCAH4	PMCAH3	PMCAH2	PMCAH1	PMCAH0	FFFFF042H	00H

Bit Position	Bit Name	Function
7 to 0	PMCAHn (n = 7 to 0)	Port Mode Control Specifies operation mode of PAHn pin. 0: I/O port mode or segment signal output in LCD-controller mode 1: A23 to A16 output mode

Caution: Make sure that Port AH is input mode and port mode before enabling LCD segment mode!

(To avoid high cross current and to ensure that LCD is working properly).

16.6 Port DL

Port DL is a 16-bit input/output port in which input or output can be specified in 1-bit units.
When using the higher 8 bits of PDL as PDLH and the lower 8 bits as PDLL, it can be used as an 8-bit input/output port that can specify input/output in 1-bit units.

Figure 16-33: Port DL (PDL)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	At Reset
PDL	PDL15	PDL14	PDL13	PDL12	PDL11	PDL10	PDL9	PDL8	PDL7	PDL6	PDL5	PDL4	PDL3	PDL2	PDL1	PDL0	FFFFF004H	0000H

Bit Position	Bit Name	Function
15 to 0	PDLn (n = 15 to 0)	Input/output port

Besides functioning as a port, in control mode, this can operate as a data bus when memory is expanded externally.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port DL	PDL15 to PDL0	D15 to D0	Memory expansion data bus	D

(2) Setting in input/output mode and control mode

Port DL is set in input/output mode using the port DL mode register (PMDL). In control mode, it is set using the port DL mode control register (PMCDL).

(a) Port DL mode register (PMDL)

The PMDL register can be read or written in 16-bit units.
When using the higher 8 bits of the PMDL register as the PMDLH register and the lower 8 bits as the PMDLL register, it can be read or written in 8- or 1-bit units.

Figure 16-34: Port DL Mode Register (PMDL)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	At Reset
PMDL	PMDL15	PMDL14	PMDL13	PMDL12	PMDL11	PMDL10	PMDL9	PMDL8	PMDL7	PMDL6	PMDL5	PMDL4	PMDL3	PMDL2	PMDL1	PMDL0	FFFFF024H	FFFFH

Bit Position	Bit Name	Function
15 to 0	PMDLn (n = 15 to 0)	Specifies input/output mode of PDLn pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port DL mode control register (PMCDL)

The PMCDL register can be read or written in 16-bit units.

When using the higher 8 bits of the PMCDL register as the PMCDLH register and the lower 8 bits as the PMCDLL register, it can be read or written in 8- or 1-bit units.

Figure 16-35: Port DL Mode Control Register (PMCDL)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	At Reset
PMCDL	PMCDL15	PMCDL14	PMCDL13	PMCDL12	PMCDL11	PMCDL10	PMCDL9	PMCDL8	PMCDL7	PMCDL6	PMCDL5	PMCDL4	PMCDL3	PMCDL2	PMCDL1	PMCDL0	FFFFF044H	0000H

Bit Position	Bit Name	Function
15 to 0	PMCDLn (n = 15 to 0)	Specifies operation mode of PDLn pin. 0: Input/output port mode 1: D15 to D0 input/output mode

16.7 Port CS

Port CS is a 3-bit input/output port in which input or output can be specified in 1-bit units.

Figure 16-36: Port CS (PCS)

	7	6	5	4	3	2	1	0	Address	At Reset
PCS	0	0	0	PCS4	PCS3	PCS2	0	0	FFFFF008H	00H

Bit Position	Bit Name	Function
7 to 0	PCSn (n = 4 to 2)	Input/output port

Besides functioning as a port, in control mode, this can operate as the chip select signal output when memory is expanded externally. It also can operate as segment signal output of LCD controller/driver.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port CS	PCS4 to PCS2	$\overline{\text{CS2}}$ to $\overline{\text{CS4}}$/ SEG24 to SEG26	Chip select signal output or segment signal output of LCD-controller	E

(2) Setting in input/output mode and control mode

Port CS is set in input/output mode using the port CS mode register (PMCS). In control mode, it is set using the port CS mode control register (PMCCS).

(a) Port CS mode register (PMCS)

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

Figure 16-37: Port CS Mode Register (PMCS)

	7	6	5	4	3	2	1	0	Address	At Reset
PMCS	1	1	1	PMCS4	PMCS3	PMCS2	1	1	FFFFF028H	FFH

Bit Position	Bit Name	Function
7 to 0	PMCSn (n = 4 to 2)	Specifies input/output mode of PCSn pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port CS mode control register (PMCCS)

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

Figure 16-38: Port CS Mode Control Register (PMCCS)

	7	6	5	4	3	2	1	0	Address	At Reset
PMCCS	0	0	0	PMCCS4	PMCCS3	PMCCS2	0	0	FFFFF048H	00H

Bit Position	Bit Name	Function
7 to 0	PMCCSn (n = 4 to 2)	Specifies operation mode of PCSn pin 0: Input/output port mode or segment signal output of LCD-controller 1: $\overline{\text{CS4}}$ to $\overline{\text{CS2}}$ output mode

Caution: Make sure that Port CS is input mode and port mode before enabling LCD segment mode!

(To avoid high cross current and to ensure that LCD is working properly).

16.8 Port CT

Port CT is an 8-bit input/output port in which input or output can be specified in 1-bit units.

Figure 16-39: Port CT (PCT)

	7	6	5	4	3	2	1	0	Address	At Reset
PCT	x	x	x	PCT4	PCT3	PCT2	PCT1	PCT0	FFFFF00AH	FFH

Bit Position	Bit Name	Function
7 to 0	PCTn (n = 4 to 0)	Input/output port

Besides functioning as a port, in control mode, this can operate as control signal outputs when memory is expanded externally. It also can operate as segment signal output of LCD controller/driver.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block type
Port CT	PCT0	$\overline{\text{LWR}}$/SEG27	Write strobe signal output or segment signal output of LCD-controller	F
	PCT1	$\overline{\text{UWR}}$/SEG28		
	PCT2	SEG29	Segment signal output of LCD-controller	
	PCT3	SEG30		
	PCT4	$\overline{\text{RD}}$/SEG31	Read strobe signal output or segment signal output of LCD-controller	

(2) Setting in input/output mode and control mode

Port CT is set in input/output mode using the port CT mode register (PMCT). In control mode, it is set using the port CT mode control register (PMCCT).

(a) Port CT mode register (PMCT)

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

Figure 16-40: Port CT Mode Register (PMCT)

	7	6	5	4	3	2	1	0	Address	At Reset
PMCT	1	1	1	PMCT4	PMCT3	PMCT2	PMCT1	PMCT0	FFFFF02AH	FFH

Bit Position	Bit Name	Function
7 to 0	PMCTn (n = 4 to 0)	Specifies input/output mode of PCTn pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port CT mode control register (PMCCT)

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

Figure 16-41: Port CT Mode Control Register (PMCCT)

	7	6	5	4	3	2	1	0	Address	At Reset
PMCCT	0	0	0	PMCCT4	PMCCT3	PMCCT2	PMCCT1	PMCCT0	FFFFF04AH	00H

Bit Position	Bit Name	Function
4	PMCCT4	Specifies operation mode of PCT4 pin 0: Input/output port mode or segment signal output of LCD-controller mode 1: $\overline{\text{RD}}$ output mode
1	PMCCT1	Specifies operation mode of PCT1 pin 0: Input/output port mode or segment signal output of LCD-controller mode 1: $\overline{\text{UWR}}$ output mode
0	PMCCT0	Specifies operation mode of PCT0 pin 0: Input/output port mode or segment signal output of LCD-controller mode 1: $\overline{\text{LWR}}$ output mode

Caution: Make sure that Port CT is input mode and port mode before enabling LCD segment mode!

16.9 Port CM

Port CM is a 2-bit input/output port in which input or output can be specified in 1-bit units.

Figure 16-42: Port CM (PCM)

	7	6	5	4	3	2	1	0	Address	At Reset
PCM	0	0	0	0	0	0	PCM1	PCM0	FFFFF00CH	00H

Bit Position	Bit Name	Function
1, 0	PCMn (n = 2 to 0)	Input/output port

Besides functioning as a port, in control mode, this can operate as the wait insertion signal input, internal system clock output, and bus hold control signal output. It also can operate as segment signal output of LCD controller/driver.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port CM	PCM0	$\overline{\text{WAIT}}$/SEG32	Wait insertion signal input or segment signal output of LCD-controller	G
	PCM1	CLKOUT/SEG33	Internal system clock output or segment signal output of LCD-controller	H

(2) Setting in input/output mode and control mode

Port CM is set in input/output mode using the port CM mode register (PMCM). In control mode, it is set using the port CM mode control register (PMCCM).

(a) Port CM mode register (PMCM)

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

Figure 16-43: Port CM Mode Register (PMCM)

	7	6	5	4	3	2	1	0	Address	At Reset
PMCM	1	1	1	1	1	1	PMCM1	PMCM0	FFFFF02CH	FFH

Bit Position	Bit Name	Function
1, 0	PMCMn (n = 2 to 0)	Specifies input/output mode of PCMn pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port CM mode control register (PMCCM)

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

Figure 16-44: Port CM Mode Control Register (PMCCM)

	7	6	5	4	3	2	1	0	Address	At Reset
PMCCM	0	0	0	0	0	0	PMCCM1	PMCCM0	FFFFF04CH	00H

Bit Position	Bit Name	Function
1	PMCCM1	Specifies operation mode of PCM1 pin 0: Input/output port mode or segment signal output of LCD-controller mode 1: CLKOUT output mode
0	PMCCM0	Specifies operation mode of PCM0 pin 0: Input/output port mode or segment signal output of LCD-controller mode 1: $\overline{\text{WAIT}}$ input mode

Caution: Make sure that Port CM is input mode and port mode before enabling LCD segment mode!

Chapter 17 RESET Function

When a low level is input to the $\overline{\text{RESET}}$ pin, there is a system reset and each hardware item of the V850E/CA1 / ATOMIC is initialized to its initial status.
When the $\overline{\text{RESET}}$ pin changes from low level to high level, reset status is released and the CPU starts program execution. The user has to initialize the contents of various registers as needed within the program.

17.1 Features

- Noise elimination of reset pin ($\overline{\text{RESET}}$) using analog delay (approximately 50 ns)

17.2 Pin Functions

During a system reset period, most pin output is high impedance (all pins except CLKOUT**Note**, $\overline{\text{RESET}}$, X2, V_{DD5x}, V_{SS5x}, V_{DD3x}, V_{SS3x}, CV_{DD}, CV_{SS}, AV_{DD}, AV_{REF}, and AV_{SS} pins).
Thus, if for example memory is extended externally, a pull-up (or pull-down) resistor must be attached to PAH, PAL, PDL, PCS, PCT, and PCM. If there are no resistors, the external memory that is connected may be destroyed when these pins become high impedance.
Similarly, perform pin processing so that on-chip peripheral I/O function signal output and output ports are not affected.

Note: The V850E/CA1 / ATOMIC has a total of 90 on-chip input/output ports (ports 1 to 6, PAL, PAH, PDL, PCS, PCT, PCM). The port configuration is shown below.

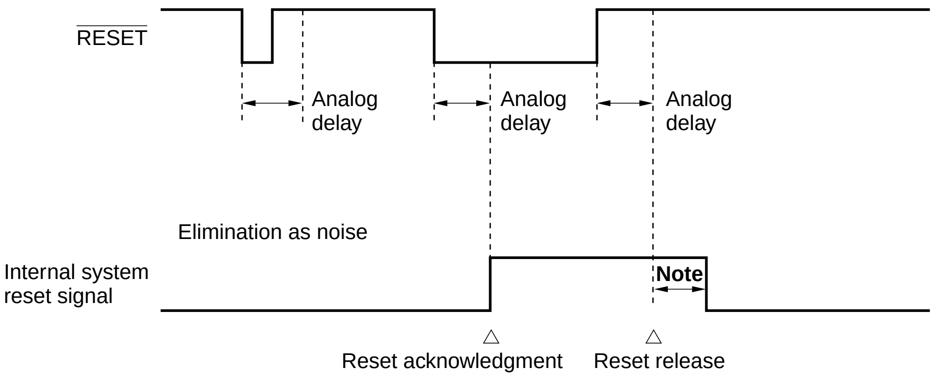
Table 17-1 shows the operation status of each pin during a reset period.

Table 17-1: Operation Status of Each Pin During Reset Period

Pin Name		Pin Status
A0 to A23, D0 to D15, $\overline{\text{CS2}}$ to $\overline{\text{CS4}}$, $\overline{\text{LWR}}$, $\overline{\text{UWR}}$, $\overline{\text{RD}}$, $\overline{\text{WAIT}}$		(Port mode)
CLKOUT		(Port mode)
Port pins	Ports 1 to 6	High impedance
	Ports PAL, PAH, PCM, PCS, PCT, PDL	High impedance

(1) Reset signal acknowledgment

Figure 17-1: Reset signal acknowledgment

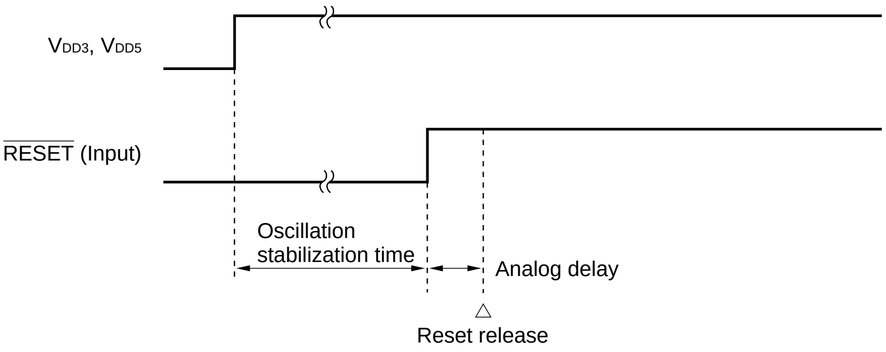


Note: The internal system reset signal continues in active status for a period of at least 4 system clocks after the timing of a reset release by the $\overline{\text{RESET}}$ signal.

(2) Reset at power-on

A reset operation at power-on (power supply application) must guarantee oscillation stabilization time from power-on until reset acknowledgment due to the low level width of the $\overline{\text{RESET}}$ signal.

Figure 17-2: Reset at power-on



17.3 Initialization

Initialize the contents of each register as needed within a program.
Table 17-2 shows the initial values of the CPU, internal RAM, and on-chip peripheral I/O’s after reset.

Table 17-2: Initial Values of CPU and Internal RAM After Reset

On-Chip Hardware		Register Name	Initial Value After Reset
CPU	Program registers	General-purpose register (r0)	00000000H
		General-purpose registers (r1 to r31)	Undefined
		Program counter (PC)	00000000H
	System registers	Status save registers during interrupt (EIPC, EIPSW)	Undefined
		Status save registers during NMI (FEPC, FEPSW)	Undefined
		Interrupt cause register (ECR)	00000000H
		Program status word (PSW)	00000020H
		Status save registers during CALLT execution (CTPC, CTPSW)	Undefined
		Status save registers during exception/debug trap (DBPC, DBPSW)	Undefined
		CALLT base pointer (CTBP)	Undefined
Internal RAM		-	Undefined

Caution: In the table above, “Undefined” means either undefined at the time of a power-on reset or undefined due to data destruction when $\overline{\text{RESET}}$ ↓ input and data write timing are synchronized. On a $\overline{\text{RESET}}$ ↓ other than this, data is maintained in its previous status.

[MEMO]

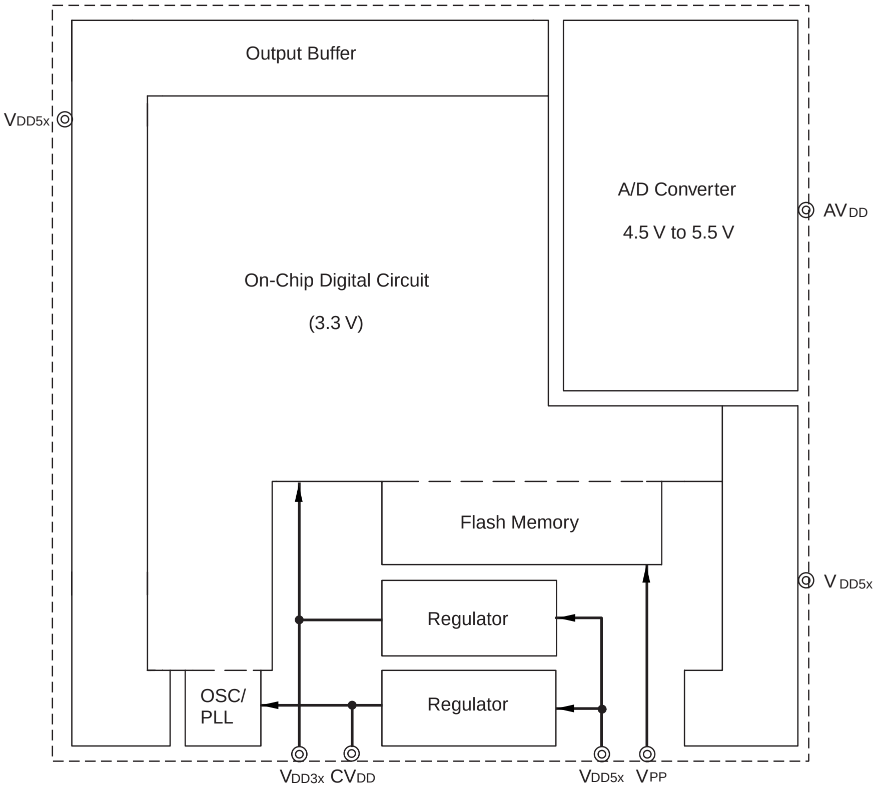
Chapter 18 Voltage Regulator

18.1 Outline

The V850E/CA1 incorporates two regulators to realize a 5-V single power supply, low power consumption, and to reduce noise.

These regulators supply a voltage obtained by stepping down V_{DD} power supply voltage to oscillation blocks and on chip logic circuits (excluding the A/D converter and output buffers). The regulators output voltage is set to 3.3 V.

Figure 18-1: Regulator



18.2 Operation

The V850E/CA1’s regulators operate in every mode (STOP, IDLE, WATCH, HALT). For stabilization of regulator outputs, it is recommended to connect capacitors to the V_{DD3X} pins and also to the CV_{DD} pin.

[MEMO]

Chapter 19 Internal Voltage Comparator

19.1 Features

- Input voltage comparison by comparator

- The comparator compares an internal reference voltage with an input voltage at the comparator input pin VCMPIN. The comparison result can be read in interrupt status flag.

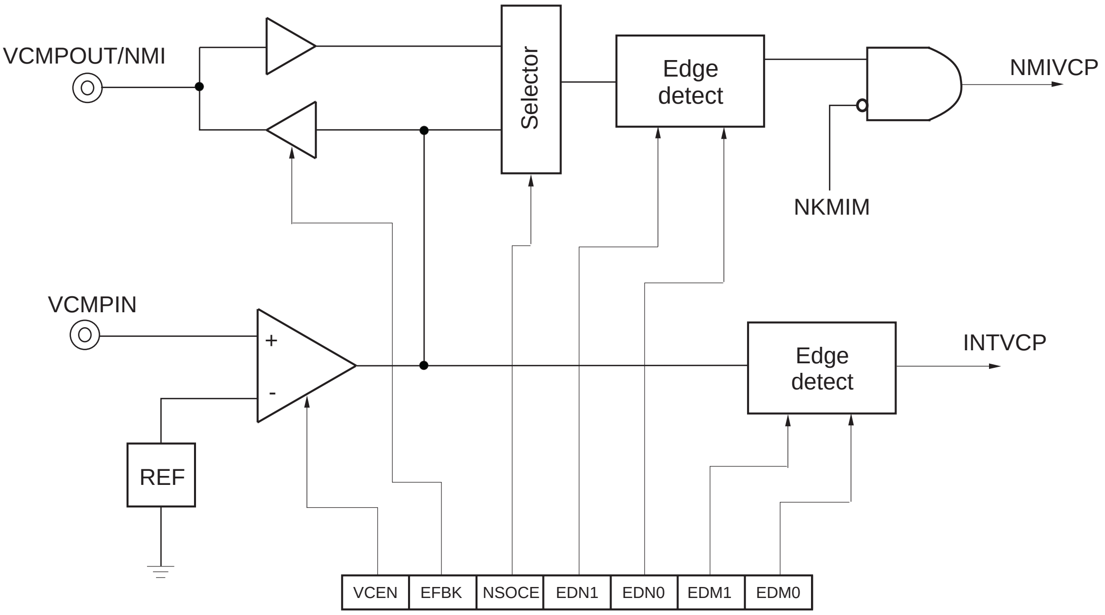
- Interrupt generation by comparator
  - The user can define the type of interrupt signal INT or NMI
  - Rising edge or falling edge can be selected
  - External NMI can be passed through from the comparator output pin VCMPOUT

- Comparator threshold and hysteresis are set by external resistors

- Easy detection of low voltage operation

The figure 19-1 shows a block diagram of the comparator.

Figure 19-1: Block Diagram of Voltage Comparator



19.2 Voltage Comparator Functions

The V850E/CA1 / ATOMIC device is designed to operate in a wide range of power supply from 3.5 V to 5.5 V. The conditions with respect to the operating voltages V_{DD}/AV_{DD} conforms to the following specification:

Table 19-1: Power Supply Voltage Operating Modes

Supply Voltage Range	Operating Mode
4.5 V – 5.5 V	Full operation
4.0 V – 4.5 V	Reduced accuracy of A/D converter, reduced quality of LCD segment display, low operation frequency (PLL off)
3.5 V – 4.0 V	Only watch or stop mode available

- In the voltage range from 4.5 V to 5.5 V the operation of CPU and all internal peripherals is guaranteed.

- In the voltage range from 4.0 V to 4.5 V a reduced accuracy of the internal peripheral A/D converter is assumed, low operation frequency.

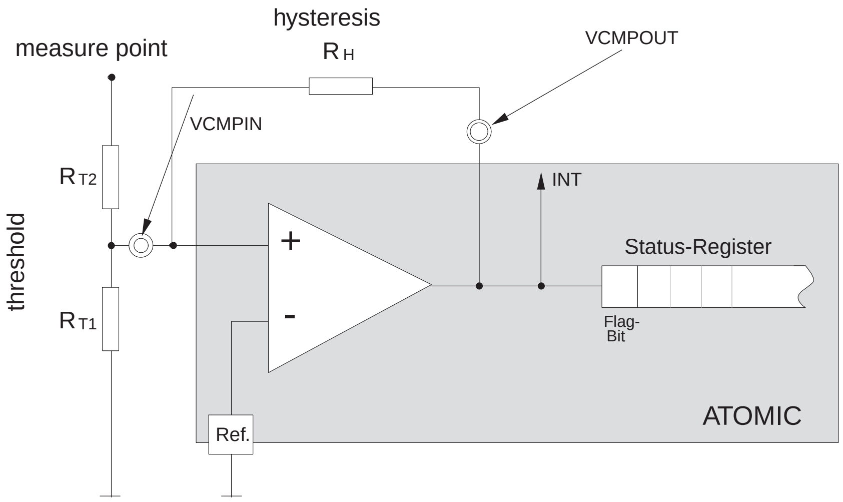
- In the voltage range from 3.5 V to 4.0 V only watch mode/stop mode operates. The A/D converter does not work at this range. The I/O-driver circuits are assumed to have reduced driving capabilities.

Caution: As the clock of the Operating System Timer is derived from the system clock, it might be necessary to adapt the operating system timer settings when entering and leaving the low frequency operating mode.

For easy detection of low voltage operation (V_{DD}/AV_{DD} < 4 V) a comparator circuit is integrated in the ATOMIC device. An interrupt is generated if the power supply drops below a threshold; this interrupt can be used by ISR to set microcontroller to low frequency operating mode. The **operating frequency needs to be switched to the lower frequency by the CPU**, otherwise a correct operation below 4.5 V can not be guaranteed.

The comparator threshold and hysteresis are set by three external resistors:

Figure 19-2: Internal Voltage Comparator



The threshold has to be set to a voltage above 4.5 V to guarantee the frequency change within the time the supply voltage drops below 4.5 V. The minimum threshold therefore depends on the maximal speed the supply voltage drops down and it depends on the execution time of the interrupt service routine.

19.2.1 Internal voltage comparator control register (VCMPM)

The VCMPN0 register is an 8-bit register, which defines the operation mode of internal voltage comparator. This register can be read/ written in 8- or 1-bit units.

	7	6	5	4	3	2	1	0	Address	At Reset
VCMPM	VCEN	NSOCE	EFBK	0	EDN1	EDN0	EDM1	EDM0	FFFFF860H	00H

<table>
<tr><th>Bit Name</th><th>Function</th></tr>
<tr><td>VCEN</td><td>Voltage Comparator Enable
Enables or disables internal voltage comparator.
0: Disabled
1: Enabled</td></tr>
<tr><td>NSOCE</td><td>NMI Source
Selects NMI source.
0: NMI from external pin input
1: NMI from internal voltage comparator
<b>Note:</b> Once set this bit to 0, reset signal only make this bit to 0.</td></tr>
<tr><td>EFBK</td><td>Enable Feedback
Enables or disables comparator feedback.
0: Disabled
1: Enabled</td></tr>
<tr><td>EDN1, EDN0</td><td>Edge detect control for NMI
<table>
<tr><th>EDN1</th><th>EDN0</th><th>Detect Mode</th></tr>
<tr><td>0</td><td>0</td><td>No detect</td></tr>
<tr><td>0</td><td>1</td><td>Positive edge</td></tr>
<tr><td>1</td><td>0</td><td>Negative edge</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
<tr><td>EDM1, EDM0</td><td>Edge detect control for INT
<table>
<tr><th>EDM1</th><th>EDM0</th><th>Toggle Mode</th></tr>
<tr><td>0</td><td>0</td><td>No detect</td></tr>
<tr><td>0</td><td>1</td><td>Positive edge</td></tr>
<tr><td>1</td><td>0</td><td>Negative edge</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
</table>

Chapter 20 Flash Memory (µPD70F3123)

The µPD70F3123 is the flash memory version of V850E/CA1 / ATOMIC, and is provided with a 256 KB flash memory. An instruction fetch from the flash memory takes one clock.
The flash memory can be programmed using a dedicated flash writer. Furthermore this product has a Selfprogramming mode, which allows to program the flash memory by control of the application without any dedicated writer.

The following can be considered as the development environment and the application using a flash memory:

- Software can be altered after the µPD70F3123 is solder mounted on the target system.
- Small scale production of various models is made easier by differentiating software.
- Data adjustment in starting mass production is made easier.
- Alter the software in the field using the Selfprogramming option.

20.1 Features

- 4-byte (1-word) access in 1 clock (instruction fetch access)
- Entire flash memory is divided into 2 areas, which can be erased separately
  - Area 0: 128 K
  - Area 1: 128 K
- Communication through serial interfaces (CSI0) from the dedicated flash writer
- Erase/write voltage: V_{PP} = 7.8 V
- On-board programming using flash writer
- Selfprogramming mode
- After erase flash memory becomes FFFFFFFFH

20.2 Writing by Flash writer

Writing can be performed either on-board or off-board by the dedicated flash writer.

(1) On-board programming

The contents of the flash memory is rewritten after µPD70F3123 is mounted on the target system. It has to be ensured that the signals required for programming are made available to the flash writer.

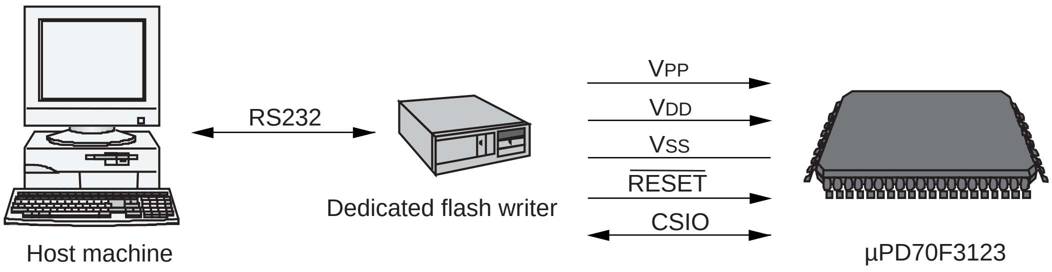
(2) Off-board programming

Writing to a flash memory is performed using a dedicated programming adapter (PA board), etc., before mounting the µPD70F3123 onto the target system.

20.3 Programming Environment

The following diagram shows the environment required for writing programs to the flash memory of the µPD70F3123.

Figure 20-1: Programming Environment in Conjunction with External Flash Writer



A host machine can be used to control the flash writer.
CSI0 is used as the interface between the flash writer and the µPD70F3123 to perform writing, erasing, etc. A programming adapter board is required for off-board writing.

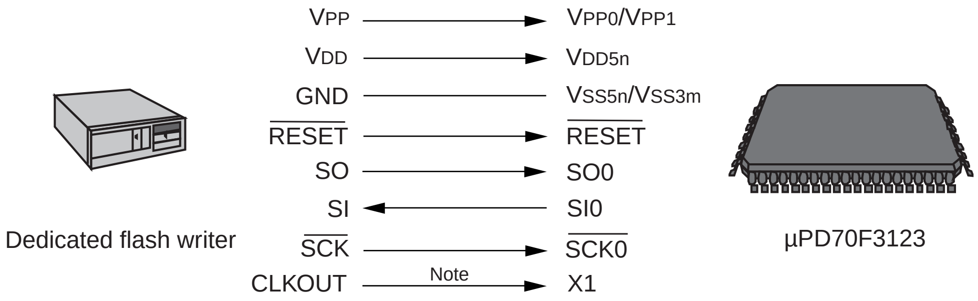
20.4 Communication System

The communication between the dedicated flash writer and the µPD70F3123 is performed by serial communication using CSI.

(1) CSI

Transfer rate: up to 1.0 Mbps (MSB first)

Figure 20-2: Flash Writer Communication via CSI0

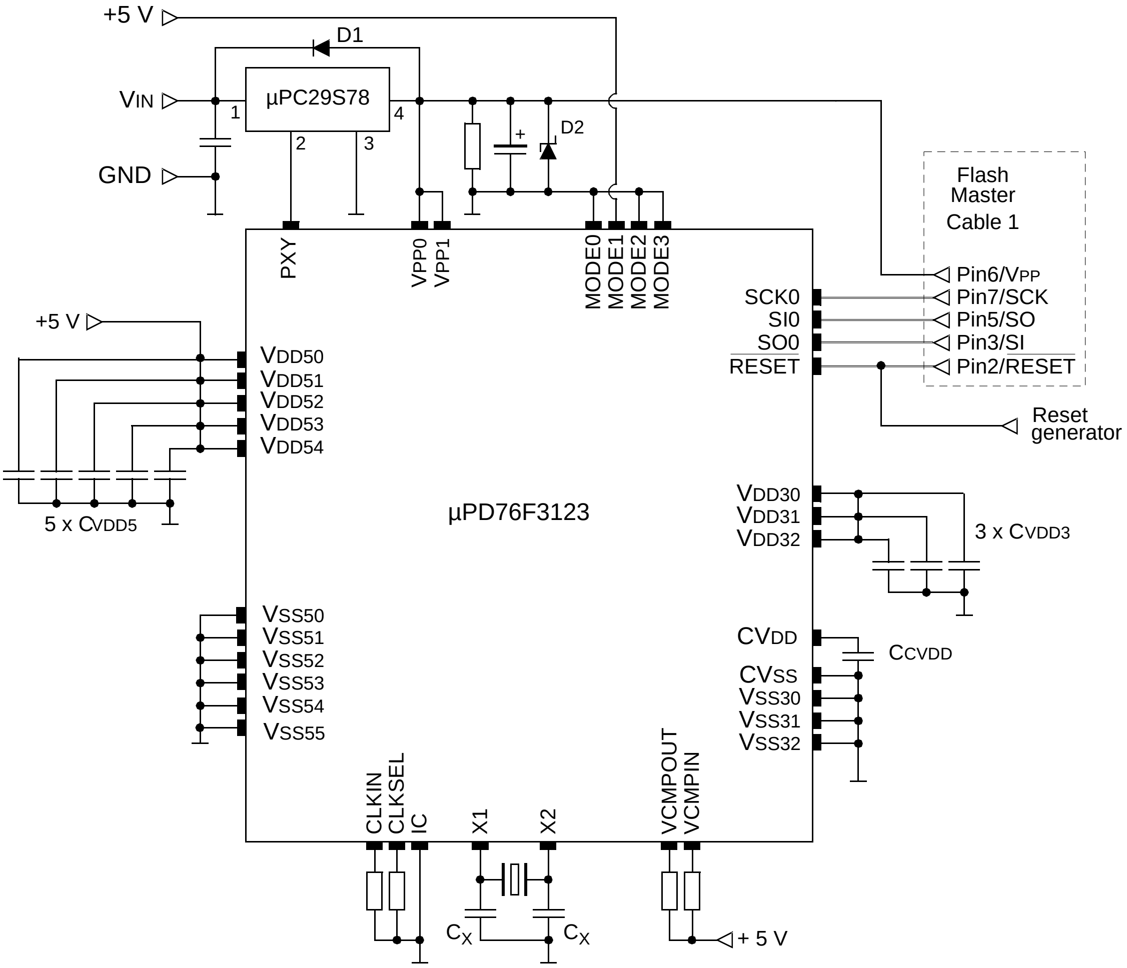


Note: The supply of operating clock from the flash writer to the µPD70F3123 is not mandatory. Like in normal operating mode, the µPD70F3123 may also operate in flash memory programming mode with a target system clock, e.g. a crystal or resonator connected to X1, X2 pins. In such case do not connect the CLKOUT signal from the flash writer.

20.5 Flash Programming Circuitry

The following schematic shows the minimal circuitry, that is needed for the µPD70F3123. The circuitry incorporates a low-dropout voltage regulator (µPC29S78) as well as flash writer support. If the device is not used for Selfprogramming the V_{PP0}/V_{PP1} pins have to be connected via a pull down resistor to ground and the voltage regulator (µPC29S78) can be removed.

Figure 20-3: Application Example for Flash Selfprogramming



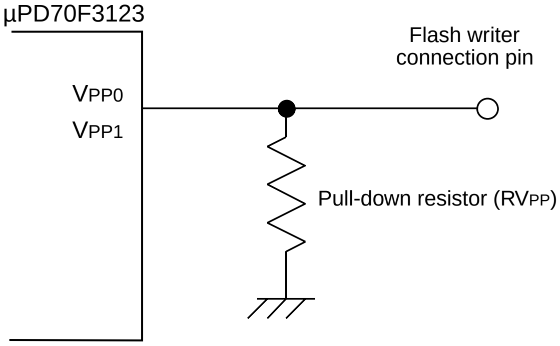
20.6 Pin Handling

When performing on-board writing, all required signals on the target system have to be made accessible to the dedicated flash writer. Also, it has to be ensured that the modes are set correctly and the V_{PP0}/V_{PP1} signal, which is required to enter the programming mode can be controlled by the flash writer. In flash memory programming mode, all pins not required for the flash memory programming, remain in the same status as immediately after reset.

20.6.1 V_{PP0}/V_{PP1} pins

In the normal operation mode, 0 V is input to both V_{PP0} and V_{PP1} pins. In the flash memory programming mode, 7.8 V writing voltage is supplied to both V_{PP0} and V_{PP1} pins. The following figure shows an example of the connection of V_{PP0}/V_{PP1} pins.

Figure 20-4: Pin Handling of V_{PP0}/V_{PP1} pins



20.6.2 Serial interface pins

The following shows the pins used by the serial interface.

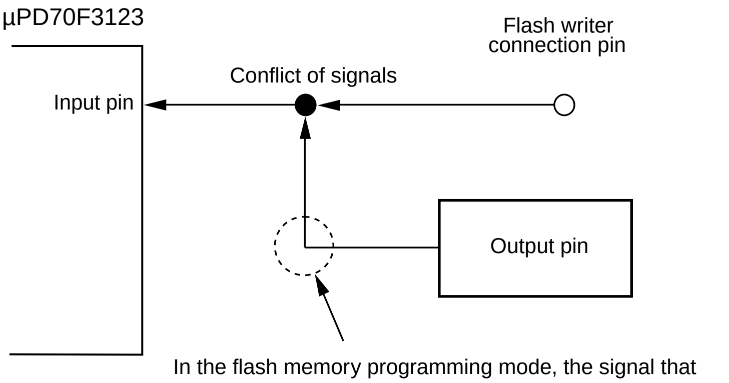
Serial Interface	Pins Used
CSI0	SO0,SI0,$\overline{\text{SCK0}}$

When connecting a dedicated flash writer to a serial interface pin, which is connected to other devices on-board, care should be taken to avoid the conflict of signals and the malfunction of other devices, etc.

(1) Conflict of signals

When connecting a dedicated flash writer (output) to a serial interface pin (input) which is connected to another device (output), conflict of signals may happen. To avoid the conflict of signals, isolate the connection to the other device or set the other device to the high-impedance status.

Figure 20-5: Conflict between Flash Writer and Other Output Pin

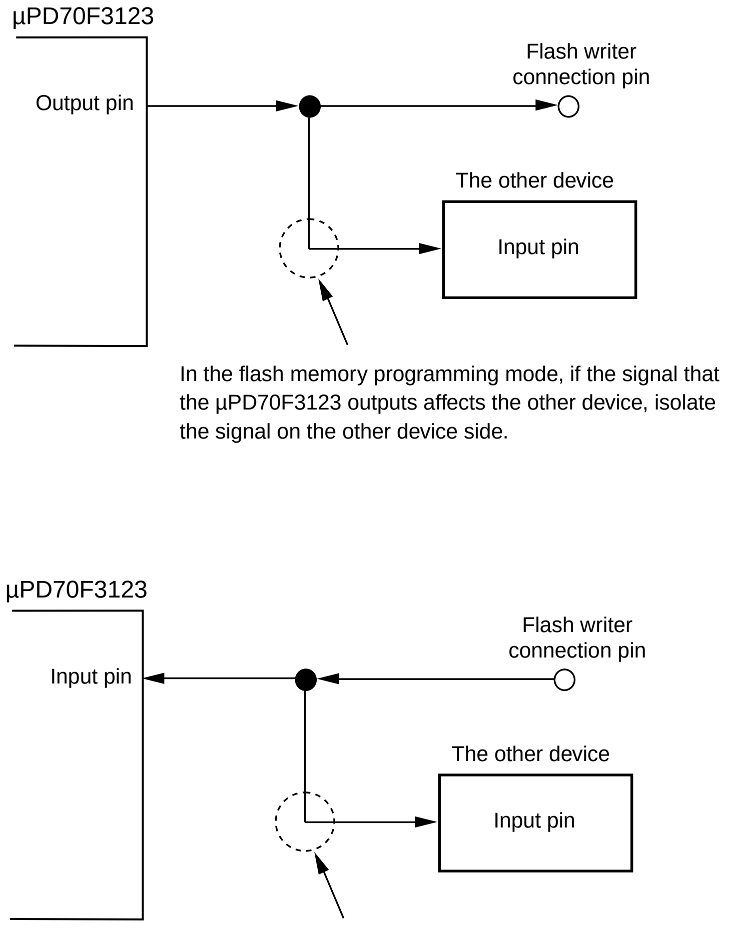


In the flash memory programming mode, the signal that the dedicated flash writer sends out conflicts with signals the other device outputs. Therefore, isolate the signals on the other device side.

(2) Malfunction of the other device

When connecting a dedicated flash writer (output or input) to a serial interface pin (input or output) connected to another device (input), the signal output to the other device may cause the device to malfunction. To avoid this, isolate the connection to the other device or make the setting so that the input signal to the other device is ignored.

Figure 20-6: Malfunction of Other Input Pins



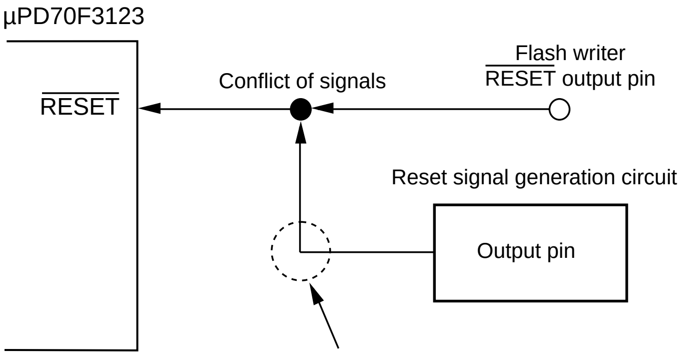
In the flash memory programming mode, if the signal that the µPD70F3123 outputs affects the other device, isolate the signal on the other device side.

In the flash memory programming mode, if the signal that the dedicated flash writer outputs affects the other device, isolate the signal on the other device side.

20.6.3 $\overline{\text{RESET}}$ pin

When connecting the reset signals of the dedicated flash writer to the $\overline{\text{RESET}}$ pin which is connected to the reset signal generation circuit on-board, conflict of signals may happen. To avoid the conflict of signals, isolate the connection to the reset signal generation circuit.
When reset signal is input from the user system during the flash memory programming mode, programming operation will not be performed correctly. Therefore, do not input signals other than the reset signals from the dedicated flash writer.

Figure 20-7: Conflict between Flash Writer Reset Line and Reset Signal Generation Circuit


µPD70F3123
$\overline{\text{RESET}}$
Conflict of signals
Flash writer
$\overline{\text{RESET}}$ output pin
Reset signal generation circuit
Output pin

In the flash memory programming mode, the signal that the reset signal generation circuit outputs conflicts with the signal that the flash writer outputs. Therefore, isolate the signals on the reset signal generation circuit side.

20.6.4 NMI pin

Do not change the input signal to the NMI pin during the flash memory programming mode. If the NMI pin is changed during the flash memory programming mode, the programming may not be performed correctly.

20.6.5 MODE pin

To switch to the flash memory programming mode, change MODE0 through MODE2 accordingly with jumpers, etc., apply writing voltage to V_{PP0}/V_{PP1} pins, and release the reset.

20.6.6 Port pins

When the flash memory programming mode is set, all the port pins except the pins which communicate with the dedicated flash writer become high-impedance status. The treatment of these port pins is not necessary.

20.6.7 Other signal pins

Connect X1, X2, CKSEL, CLOCKIN, IC, VCMPIN, VCMPOUT, and AV_{REF} to the same status as that in the normal operation mode.

20.6.8 Power supply

Provide the same power supply (V_{DD50} to V_{DD54}, V_{SS5} to V_{SS54}, V_{DD30} to V_{DD33}, V_{SS30} to V_{SS33}, AV_{DD}, AV_{SS}, CV_{DD}, CV_{SS}) as that in normal operation mode.

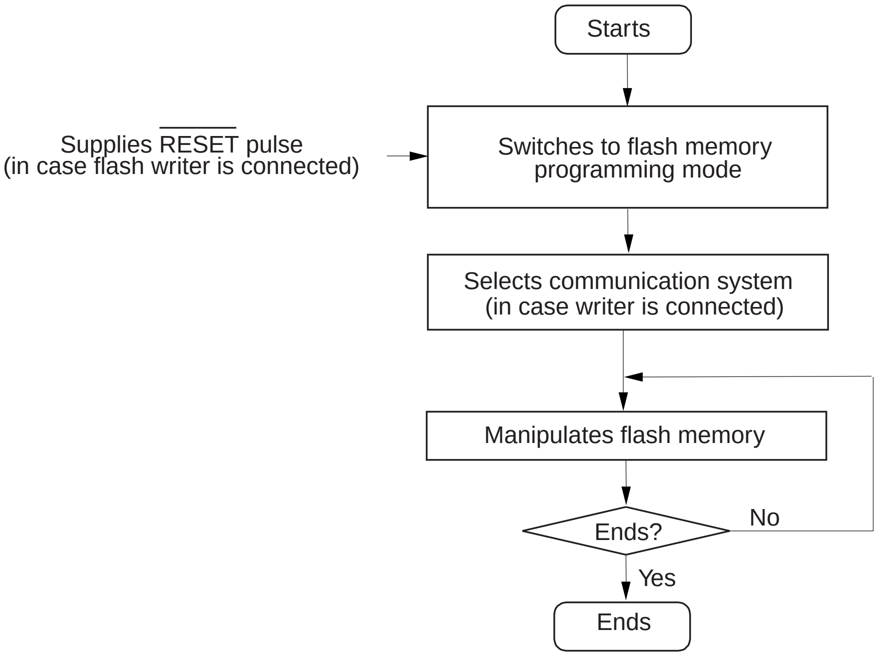
20.7 Programming Method

20.7.1 Flash memory control

To manipulate the flash memory the µPD70F3123 has to operate in a special flash memory programming mode. This mode can be entered either by applying the programming voltage of 7.8 V to the V_{PP0}/ V_{PP1} before the reset is release or by entering the Selfprogramming mode.

The following figure shows the procedure for manipulating the flash memory.

Figure 20-8: Flow Chart of Flash Memory Manipulation


Starts
Supplies RESET pulse
(in case flash writer is connected)
Switches to flash memory programming mode
Selects communication system
(in case writer is connected)
Manipulates flash memory
Ends?
No
Yes
Ends

20.7.2 Selection of communication mode

In the µPD70F3123 as well as for other V850 family devices, a communication system is selected by inputting pulses (16 pulses max.) to V_{PP0} pin after switching to the flash memory programming mode. The V_{PP0} pulses are generated by the dedicated flash writer.
The following table shows the relation between the number of pulses and the communication systems.

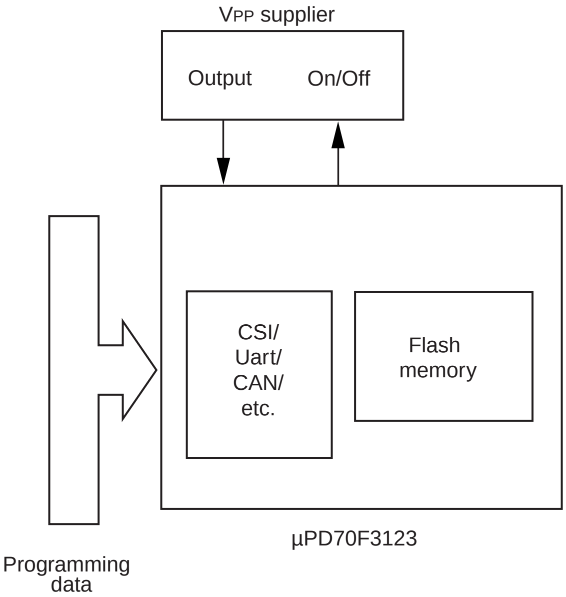
Table 20-1: List of Communication Systems

VPP pulse	Communication System	Remarks
0	CSI0	µPD70F3123 performs slave operation, MSB first
Others	(reserved)	Setting prohibited

20.8 Selfprogramming Mode

The flash Selfprogramming feature allows user to reprogram the flash contents by a user application program, without the necessity of an external flash writer.
This feature allows an update of the application with only on-board resources and a user defined communication interface.

Figure 20-9: Configuration in Selfprogramming Mode



In order to operate flash Selfprogramming, flash Selfprogramming libraries are prepared for user.

Following operations to the flash memory are supported by libraries.

- Initialize
- Blank Check
- Erase
- Write
- Verify
- Blank check
- Vpp Voltage Check
- Create Signature
- Check Signature
- Swap Area
- Check Area

For further details please refer to the following documents:

- Application Note - Selfprogramming 32-/16-bit Single chip microcontroller Selfprogramming Library.

20.9 Secure Selfprogramming

20.9.1 General description

A flash memory area can only be erased as a whole. If parts of the lower flash area have to be updated, the complete flash memory has to be erased. This bears the risk that a problem during Selfprogramming, particularly a power failure, leaves the device without any valid program for start-up.
To overcome those limitations Atomic features a method which is called “secure Selfprogramming”. By using “secure Selfprogramming”, it is always ensured that a valid boot program is available in the flash memory. This is achieved by enabling the user to select which of the two flash areas is mapped at address 0 and therefore accessed after reset, thus ensuring that the boot program located in this area is executed.

This selection is done by creating a signature at address 1000H or 21000H, depending on which area should become the one located at address 0. Directly after reset, the device determines which area contains a valid signature and maps this area to address 0.

20.9.2 Signature Structure

The library provides a function to create a signature in either one of the two areas. It is located within the user address space of the flash memory. The signature structure was chosen in a way to ensure that no user data can be mistakenly interpreted as a signature. This is achieved by a different usage of internal structures of the flash memory.

20.9.3 Secure Selfprogramming flow

A reprogramming of the flash memory starts with an erase of the upper area. After a successful erase, the boot program has to be written to the upper area. The boot program can be a copy, but can also be modified. Afterwards, a signature is created in the upper area, indicating that this area contains a valid boot program. The signature which is found in the lower area is killed by writing a 00000000H to this address, and the areas are swapped, ensuring that the copied boot program is now located at address 0. This is followed by an erase of the area, which became the upper one still containing the old boot program and an invalid signature. After completion of the erase operation, the flash memory contains now only the boot program, so that the new application program can be written.

The flow looks as follows:

- Erase upper area
- Copy boot program from lower area to upper area
- Create signature in upper area
- Kill signature in lower area
- Swap area so that the lower area becomes the upper and vice versa
- Erase the upper area (which was formerly the lower)
- Write new application program to the flash memory

Figure 20-10: Secure Selfprogramming Flow (1/2)

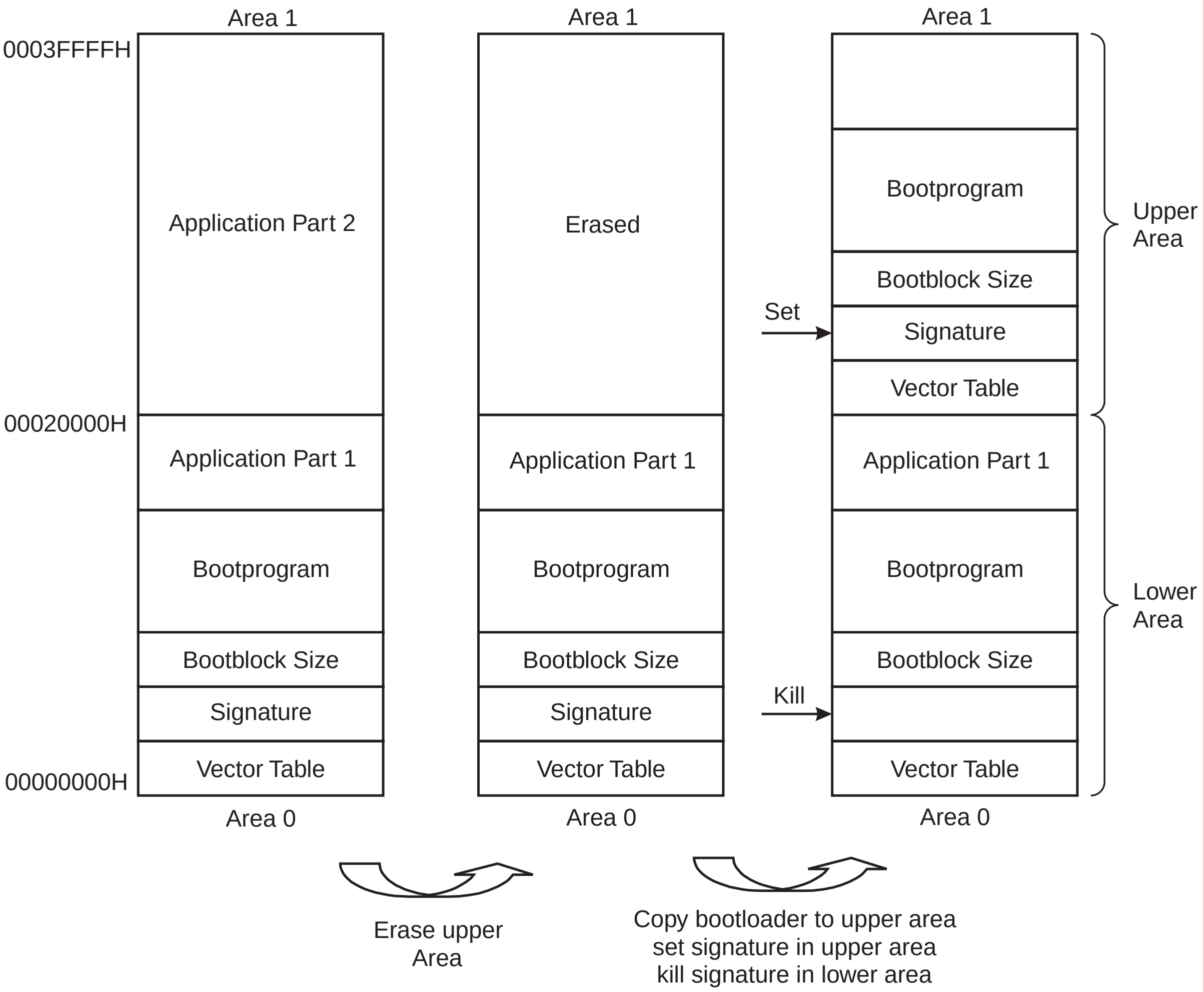
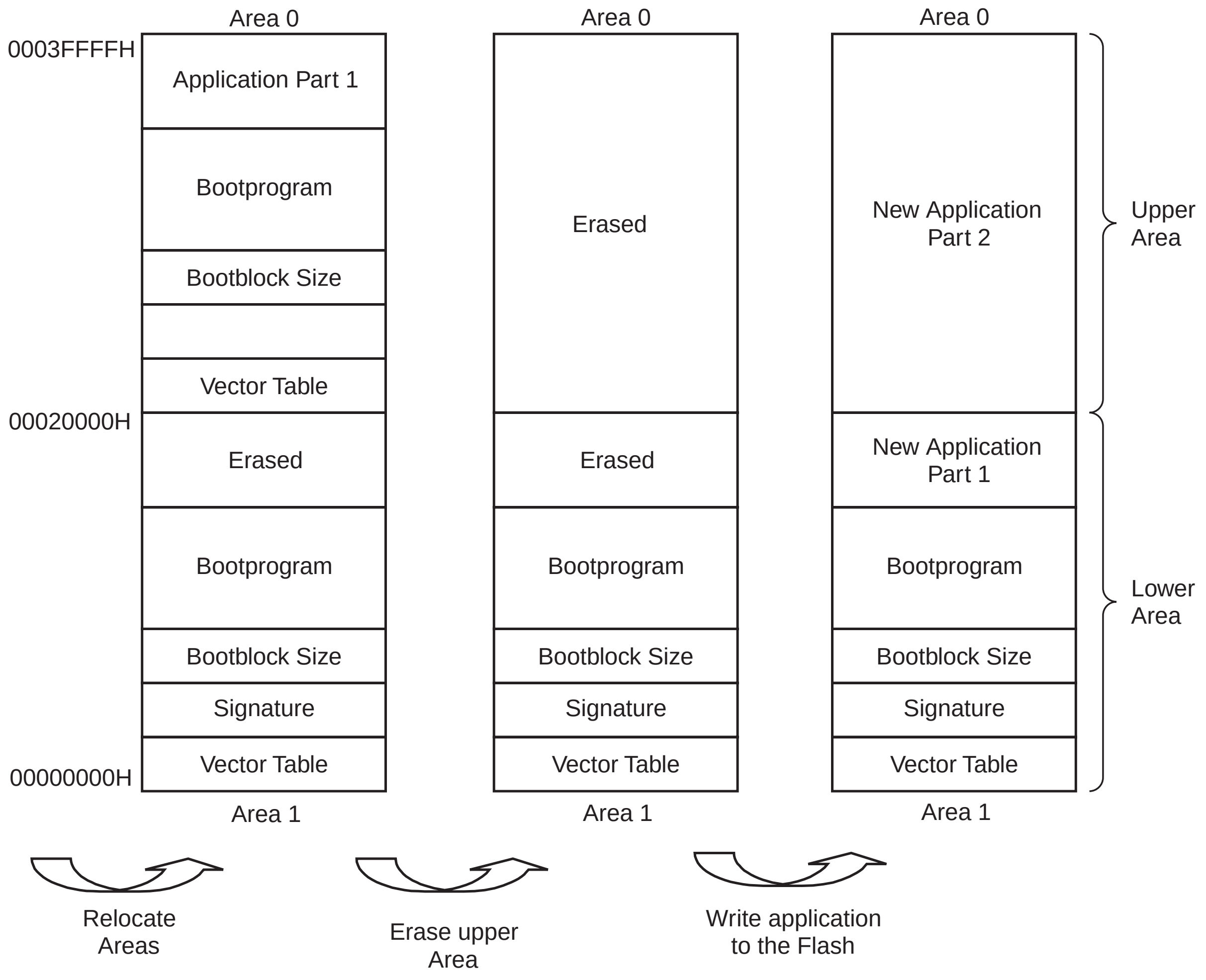


Figure 20-10: Secure Selfprogramming Flow (2/2)



20.9.4 Advantages of Secure Selfprogramming

- A boot program is always available, thus ensuring that the device can always be reprogrammed.
- The size of the boot block is not fixed. All sizes up to 128 KB are supported.
- It is possible to update the boot program using the secure mechanisms.

[MEMO]

Appendix A Instruction Set List

A.1 Convention

(a) Register symbols used to describe operands

Register Symbol	Explanation
reg1	General registers: Used as source registers
reg2	General registers: Used mainly as destination registers. Also used as source register in some instructions.
reg3	General registers: Used mainly to store the remainders of division results and the higher order 3 bits of multiplication results.
bit#3	33-bit data for specifying the bit number
immX	X bit immediate data
dispX	X bit displacement data
regID	System register number
vector	5-bit data that specifies the trap vector (00H to 1FH)
cccc	4-bit data that shows the conditions code
sp	Stack pointer (SP)
ep	Element pointer (r30)
listX	X item register list

(b) Register symbols used to describe opcodes

Register Symbol	Explanation
R	1-bit data of a code that specifies reg1 or regID
r	1-bit data of the code that specifies reg2
w	1-bit data of the code that specifies reg3
d	1-bit displacement data
I	1-bit immediate data (indicates the higher bits of immediate data)
i	1-bit immediate data
cccc	4-bit data that shows the condition codes
CCCC	4-bit data that shows the condition codes of Bcond instruction
bbb	3-bit data for specifying the bit number
L	1-bit data that specifies a program register in the register list
S	1-bit data that specifies a system register in the register list

(c) Register symbols used in operation

Register Symbol	Explanation
¨	Input for
GR [ ]	General register
SR [ ]	System register
zero-extend (n)	Expand n with zeros until word length.
sign-extend (n)	Expand n with signs until word length.
load-memory (a, b)	Read size b data from address a.
store-memory (a, b, c)	Write data b into address a in size c.
load-memory-bit (a, b)	Read bit b of address a.
store-memory-bit (a, b, c)	Write c to bit b of address a.
saturated (n)	Execute saturated processing of n (n is a 2’s complement). If, as a result of calculations, n ≥ 7FFFFFFFH, let it be 7FFFFFFFH. n ≤ 80000000H, let it be 80000000H.
result	Reflects the results in a flag.
Byte	Byte (8 bits)
Half-word	Half word (16 bits)
Word	Word (32 bits)
+	Addition
–	Subtraction
ll	Bit concatenation
×	Multiplication
÷	Division
%	Remainder from division results
AND	Logical product
OR	Logical sum
XOR	Exclusive OR
NOT	Logical negation
logically shift left by	Logical shift left
logically shift right by	Logical shift right
arithmetically shift right by	Arithmetic shift right

(d) Register symbols used in an execution clock

Register Symbol	Explanation
I	If executing another instruction immediately after executing the first instruction (issue).
r	If repeating execution of the same instruction immediately after executing the first instruction (repeat).
l	If using the results of instruction execution in the instruction immediately after the execution (latency).

(e) Register symbols used in flag operations

Identifier	Explanation
(Blank)	No change
0	Clear to 0
X	Set or cleared in accordance with the results.
R	Previously saved values are restored.

(f) Condition codes

Condition Name (cond)	Condition Code (cccc)	Condition Formula	Explanation
V	0 0 0 0	OV = 1	Overflow
NV	1 0 0 0	OV = 0	No overflow
C/L	0 0 0 1	CY = 1	Carry Lower (Less than)
NC/NL	1 0 0 1	CY = 0	No carry Not lower (Greater than or equal)
Z/E	0 0 1 0	Z = 1	Zero Equal
NZ/NE	1 0 1 0	Z = 0	Not zero Not equal
NH	0 0 1 1	(CY or Z) = 1	Not higher (Less than or equal)
H	1 0 1 1	(CY or Z) = 0	Higher (Greater than)
N	0 1 0 0	S = 1	Negative
P	1 1 0 0	S = 0	Positive
T	0 1 0 1	—	Always (Unconditional)
SA	1 1 0 1	SAT = 1	Saturated
LT	0 1 1 0	(S xor OV) = 1	Less than signed
GE	1 1 1 0	(S xor OV) = 0	Greater than or equal signed
LE	0 1 1 1	((S xor OV) or Z) = 1	Less than or equal signed
GT	1 1 1 1	((S xor OV) or Z) = 0	Greater than signed

A.2 Instruction Set (In Alphabetical Order)

(1/4)

Mnemonic	Operand	Opcode	Operation		Execution Clock			Flags				
					i	r	l	CY	OV	S	Z	SAT
ADD	reg1,reg2	`rrrrr001110RRRRR`	GR[reg2]←GR[reg2] + GR[reg1]		1	1	1	×	×	×	×	
	imm5,reg2	`rrrrr010010iiiii`	GR[reg2]←GR[reg2] + sign-extend(imm5)		1	1	1	×	×	×	×	
ADDI	imm16,reg1,reg2	`rrrrr110000RRRRR` `iiiiiiiiiiiiiiii`	GR[reg2]←GR[reg1] + sign-extend(imm16)		1	1	1	×	×	×	×	
AND	reg1,reg2	`rrrrr001010RRRRR`	GR[reg2]←GR[reg2] AND GR[reg1]		1	1	1		0	×	×	
ANDI	imm16,reg1,reg2	`rrrrr110110RRRRR` `iiiiiiiiiiiiiiii`	GR[reg2]←GR[reg1] AND zero-extend(imm16)		1	1	1		0	0	×	×
Bcond	disp9	`ddddd1011dddcccc` **Note 1**	if conditions are satisfied then PC←PC+sign-extend(disp9)	When conditions are satisfied	2 **Note 2**	2 **Note 2**	2 **Note 2**					
				When conditions are not satisfied	1	1	1					
BSH	reg2,reg3	`rrrrr11111100000` `wwwww01101000010`	GR[reg3]←GR[reg2] (23 : 16) ll GR[reg2] (31 : 24) ll GR[reg2] (7 : 0) ll GR[reg2] (15 : 8)		1	1	1	×	0	×	×	
BSW	reg2,reg3	`rrrrr11111100000` `wwwww01101000000`	GR[reg3]←GR[reg2] (7 : 0) ll GR[reg2] (15 : 8) ll GR[reg2] (23 : 16) ll GR[reg2] (31 : 24)		1	1	1	×	0	×	×	
CALLT	imm6	`0000001000iiiiii`	CTPC←PC + 2(return PC) CTPSW←PSW adr←CTBP+zero-extend(imm6 logically shift left by 1) PC←CTBP+zero-extend(Load-memory(adr,Half-word))		4	4	4					
CLR1	bit#3, disp16[reg1]	`10bbb111110RRRRR` `dddddddddddddddd`	adr←GR[reg1] + sign-extend(disp16) Z flag←Not(Load-memory-bit(adr,bit#3)) Store-memory-bit(adr,bit#3,0)		3 **Note 3**	3 **Note 3**	3 **Note 3**				×	
	reg2,[reg1]	`rrrrr111111RRRRR` `0000000011100100`	adr←GR[reg1] Z flag←Not(Load-memory-bit(adr,reg2)) Store-memory-bit(adr,reg2,0)		3 **Note 3**	3 **Note 3**	3 **Note 3**				×	
CMOV	cccc,imm5,reg2, reg3	`rrrrr111111iiiii` `wwwww011000cccc0`	if conditions are satisfied then GR[reg3]←sign-extended(imm5) else GR[reg3]←GR[reg2]		1	1	1					
	cccc,reg1,reg2, reg3	`rrrrr111111RRRRR` `wwwww011001cccc0`	if conditions are satisfied then GR[reg3]←GR[reg1] else GR[reg3]←GR[reg2]		1	1	1					
CMP	reg1,reg2	`rrrrr001111RRRRR`	result←GR[reg2] – GR[reg1]		1	1	1	×	×	×	×	
	imm5,reg2	`rrrrr010011iiiii`	result←GR[reg2] – sign-extend(imm5)		1	1	1	×	×	×	×	
CTRET		`0000011111100000` `0000000101000100`	PC←CTPC PSW←CTPSW		3	3	3	R	R	R	R	R
DBRET		`0000011111100000` `0000000101000110`	PC←DBPC PSW←DBPSW		3	3	3	R	R	R	R	R
DBTRAP		`1111100001000000`	DBPC←PC + 2 (returned PC) DBPSW←PSW PSW.NP←1 PSW.EP←1 PSW.ID←1 PC←00000060H		3	3	3					
DI		`0000011111100000` `0000000101100000`	PSW.ID←1		1	1	1					
DISPOSE	imm5,list12	`0000011001iiiiiL` `LLLLLLLLLLL00000`	sp←sp + zero-extend(imm5 logically shift left by 2) GR[reg in list12]←Load-memory(sp,Word) sp←sp + 4 repeat 2 steps above until all regs in list12 are loaded		N+1 **Note 4**	N+1 **Note 4**	N+1 **Note 4**					
	imm5,list12,[reg1]	`0000011001iiiiiL` `LLLLLLLLLLLRRRRR` **Note 5**	sp←sp + zero-extend(imm5 logically shift left by 2) R[reg in list12]←Load-memory(sp,Word) sp←sp + 4 repeat 2 steps above until all regs in list12 are loaded PC←GR[reg1]		N+3 **Note 4**	N+3 **Note 4**	N+3 **Note 4**					
DIV	reg1,reg2,reg3	`rrrrr111111RRRRR` `wwwww01011000000`	GR[reg2]←GR[reg2] ÷ GR[reg1] GR[reg3]←GR[reg2] % GR[reg1]		35	35	35					
DIVH	reg1,reg2	`rrrrr000010RRRRR`	GR[reg2]←GR[reg2] ÷ GR[reg1] **Note 6**		35	35	35		×	×	×	
	reg1,reg2,reg3	`rrrrr111111RRRRR` `wwwww01010000000`	GR[reg2]←GR[reg2] ÷ GR[reg1] **Note 6** GR[reg3]←GR[reg2] % GR[reg1]		35	35	35		×	×	×	
DIVHU	reg1,reg2,reg3	`rrrrr111111RRRRR` `wwwww01010000010`	GR[reg2]←GR[reg2] ÷ GR[reg1] **Note 6** GR[reg3]←GR[reg2] % GR[reg1]		34	34	34		×	×	×	
DIVU	reg1,reg2,reg3	`rrrrr111111RRRRR` `wwwww01011000010`	GR[reg2]←GR[reg2] ÷ GR[reg1] GR[reg3]←GR[reg2] % GR[reg1]		34	34	34		×	×	×	
EI		`1000011111100000` `0000000101100000`	PSW.ID←0		1	1	1					

(2/4)

Mnemonic	Operand	Opcode	Operation		Execution Clock			Flags				
					i	r	l	CY	OV	S	Z	SAT
HALT		0000011111100000 0000000100100000	Stop		1	1	1					
HSW	reg2,reg3	rrrrr11111100000 wwwww01101000100	GR[reg3]←GR[reg2](15 : 0) ll GR[reg2] (31 : 16)		1	1	1	×	0	×	×	
JARL	disp22,reg2	rrrrr11110dddddd ddddddddddddddd0 **Note 7**	GR[reg2]←PC + 4 PC←PC + sign-extend(disp22)		2	2	2					
JMP	[reg1]	00000000011RRRRR	PC←GR[reg1]		3	3	3					
JR	disp22	0000011110dddddd ddddddddddddddd0 **Note 7**	PC←PC + sign-extend(disp22)		2	2	2					
LD.B	disp16[reg1],reg2	rrrrr111000RRRRR dddddddddddddddd	adr←GR[reg1] + sign-extend(disp16) GR[reg2]←sign-extend(Load-memory(adr,Byte))		1	1	**Note 11**					
LD.BU	disp16[reg1],reg2	rrrrr11110bRRRRR ddddddddddddddd1 **Notes 8, 10**	adr←GR[reg1] + sign-extend(disp16) GR[reg2]←zero-extend(Load-memory(adr,Byte))		1	1	**Note 11**					
LD.H	disp16[reg1],reg2	rrrrr111001RRRRR ddddddddddddddd0 **Note 8**	adr←GR[reg1] + sign-extend(disp16) GR[reg2]←sign-extend(Load-memory(adr,Half-word))		1	1	**Note 11**					
LDSR	reg2,regID	rrrrr111111RRRRR 0000000000100000 **Note 12**	SR[regID]←GR[reg2]	Other than regID = PSW	1	1	1					
				regID = PSW	1	1	1	×	×	×	×	×
LD.HU	disp16[reg1],reg2	rrrrr111111RRRRR ddddddddddddddd1 **Note 8**	adr←GR[reg1]+sign-exend(disp16) GR[reg2]←zero-extend(Load-memory(adr,half-word)		1	1	**Note 11**					
LD.W	disp16[reg1],reg2	rrrrr111001RRRRR ddddddddddddddd1 **Note 8**	adr←GR[reg1] + sign-exend(disp16) GR[reg2]←Load-memory(adr,Word)		1	1	**Note 9**					
MOV	reg1,reg2	rrrrr000000RRRRR	GR[reg2]←GR[reg1]		1	1	1					
	imm5,reg2	rrrrr010000iiiii	GR[reg2]←sign-extend(imm5)		1	1	1					
	imm32,reg1	00000110001RRRRR iiiiiiiiiiiiiiii iiiiiiiiiiiiiiii	GR[reg1]←imm32		2	2	2					
MOVEA	imm16,reg1,reg2	rrrrr110001RRRRR iiiiiiiiiiiiiiii	GR[reg2]←GR[reg1] + sign-extend(imm16)		1	1	1					
MOVHI	imm16,reg1,reg2	rrrrr110010RRRRR iiiiiiiiiiiiiiii	GR[reg2]←GR[reg1] + (imm16 ll 0^{16})		1	1	1					
MUL	reg1,reg2,reg3	rrrrr111111RRRRR wwwww01000100000	GR[reg3] ll GR[reg2]←GR[reg2] × GR[reg1]		1	2 **Note 14**	2					
	imm9,reg2,reg3	rrrrr111111iiiii wwwww01001IIII00	GR[reg3] ll GR[reg2]←GR[reg2] × sign-extend(imm9) **Note 13**		1	2 **Note 14**	2					
MULH	reg1,reg2	rrrrr000111RRRRR	GR[reg2]←GR[reg2]**Note 6** × GR[reg1]**Note 6**		1	1	2					
	imm5,reg2	rrrrr010111iiiii	GR[reg2]←GR[reg2]**Note 6** × sign-extend(imm5)		1	1	2					
MULHI	imm16,reg1,reg2	rrrrr110111RRRRR iiiiiiiiiiiiiiii	GR[reg2]←GR[reg1]**Note 6** × imm16		1	1	2					
MULU	reg1,reg2,reg3	rrrrr111111RRRRR wwwww01000100010	GR[reg3] ll GR[reg2]←GR[reg2]xGR[reg1]		1	2 **Note 14**	2					
	imm9,reg2,reg3	rrrrr111111iiiii wwwww01001IIII10	GR[reg3] ll GR[reg2]←GR[reg2]xzero-extend(imm9) **Note 13**		1	2 **Note 14**	2					
NOP		0000000000000000	Pass at least one clock cycle doing nothing.		1	1	1					
NOT	reg1,reg2	rrrrr000001RRRRR	GR[reg2]←NOT(GR[reg1])		1	1	1		0	×	×	
NOT1	bit#3,disp16[reg1]	01bbb111110RRRRR dddddddddddddddd	adr←GR[reg1]+sign-extend(disp16) Z flag←Not(Load-memory-bit(adr,bit#3)) Store-memory-bit(adr,bit#3,Z flag)		3 **Note 3**	3 **Note 3**	3 **Note 3**				×	
	reg2,[reg1]	rrrrr111111RRRRR 0000000011100010	adr←GR[reg1] Z flag←Not(Load-memory-bit(adr,reg2)) Store-memory-bit(adr,reg2,Z flag)		3 **Note 3**	3 **Note 3**	3 **Note 3**				×	
OR	reg1,reg2	rrrrr001000RRRRR	GR[reg2]←GR[reg2]OR GR[reg1]		1	1	1		0	×	×	
ORI	imm16,reg1,reg2	rrrrr110100RRRRR iiiiiiiiiiiiiiii	GR[reg2]←GR[reg1]OR zero-extend(imm16)		1	1	1		0	×	×	

(3/4)

Mnemonic	Operand	Opcode	Operation	Execution Clock			Flags				
				i	r	l	CY	OV	S	Z	SAT
PREPARE	list12,imm5	0000011110iiiiiL LLLLLLLLLLL00001	Store-memory(sp – 4,GR[reg in list12],Word) sp←sp – 4 repeat 1 step above until all regs in list12 are stored sp←sp – zero-extend(imm5)	n+1 Note 4	n+1 Note 4	n+1 Note 4					
	list12,imm5, sp/imm Note 15	0000011110iiiiiL LLLLLLLLLLLff011 imm16/imm32 Note 16	Store-memory(sp – 4,GR[reg in list12],Word) sp←sp – 4 repeat 1 step above until all regs in list12 are stored sp←sp – zero-extend(imm5) ep←sp/imm	n+2 Note 4 Note 17	n+2 Note 4 Note 17	n+2 Note 4 Note 17					
RETI		0000011111100000 0000000101000000	if PSW.EP=1 then PC ←EIPC PSW ←EIPSW else if PSW.NP=1 then PC ←FEPC PSW ←FEPSW else PC←EIPC PSW ←EIPSW	3	3	3	R	R	R	R	R
SAR	reg1,reg2	rrrrr111111RRRRR 0000000010100000	GR[reg2]←GR[reg2] arithmetically shift right by GR[reg1]	1	1	1	×	0	×	×	
	imm5,reg2	rrrrr010101iiiii	GR[reg2]←GR[reg2] arithmetically shift right by zero-extend (imm5)	1	1	1	×	0	×	×	
SASF	cccc,reg2	rrrrr1111110cccc 0000001000000000	if conditions are satisfied then GR[reg2]←(GR[reg2]Logically shift left by 1) OR 00000001H else GR[reg2]←(GR[reg2]Logically shift left by 1) OR 00000000H	1	1	1					
SATADD	reg1,reg2	rrrrr000110RRRRR	GR[reg2]←saturated(GR[reg2]+GR[reg1])	1	1	1	×	×	×	×	×
	imm5,reg2	rrrrr010001iiiii	GR[reg2]←saturated(GR[reg2]+sign-extend(imm5)	1	1	1	×	×	×	×	×
SATSUB	reg1,reg2	rrrrr000101RRRRR	GR[reg2]←saturated(GR[reg2]–GR[reg1])	1	1	1	×	×	×	×	×
SATSUBI	imm16,reg1,reg2	rrrrr110011RRRRR iiiiiiiiiiiiiiii	GR[reg2]←saturated(GR[reg1]–sign-extend(imm16)	1	1	1	×	×	×	×	×
SATSUBR	reg1,reg2	rrrrr000100RRRRR	GR[reg2]←saturated(GR[reg1]–GR[reg2])	1	1	1	×	×	×	×	×
SETF	cccc,reg2	rrrrr1111110cccc 0000000000000000	If conditions are satisfied then GR[reg2]←00000001H else GR[reg2]←00000000H	1	1	1					
SET1	bit#3,disp16[reg1]	00bbb111110RRRRR dddddddddddddddd	adr←GR[reg1] + sign-extend(disp16) Z flag←Not (Load-memory-bit(adr,bit#3)) Store-memory-bit(adr,bit#3,1)	3 Note 3	3 Note 3	3 Note 3				×	
	reg2,[reg1]	rrrrr111111RRRRR 0000000011100000	adr←GR[reg1] Z flag←Not(Load-memory-bit(adr,reg2)) Store-memory-bit(adr,reg2,1)	3 Note 3	3 Note 3	3 Note 3				×	
SHL	reg1,reg2	rrrrr111111RRRRR 0000000011000000	GR[reg2]←GR[reg2] logically shift left by GR[reg1]	1	1	1	×	0	×	×	
	imm5,reg2	rrrrr010110iiiii	GR[reg2]←GR[reg2] logically shift left by zero-extend(imm5)	1	1	1	×	0	×	×	
SHR	reg1,reg2	rrrrr111111RRRRR 0000000010000000	GR[reg2]←GR[reg2] logically shift right by GR[reg1]	1	1	1	×	0	×	×	
	imm5,reg2	rrrrr010100iiiii	GR[reg2]←GR[reg2] logically shift right by zero-extend(imm5)	1	1	1	×	0	×	×	
SLD.B	disp7[ep],reg2	rrrrr0110ddddddd	adr←ep + zero-extend(disp7) GR[reg2]←sign-extend(Load-memory(adr,Byte))	1	1	Note 9					
SLD.BU	disp4[ep],reg2	rrrrr0000110dddd Note 18	adr←ep + zero-extend(disp4) GR[reg2]←zero-extend(Load-memory(adr,Byte))	1	1	Note 9					
SLD.H	disp8[ep],reg2	rrrrr1000ddddddd Note 19	adr←ep + zero-extend(disp8) GR[reg2]←sign-extend(Load-memory(adr,Half-word))	1	1	Note 9					
SLD.HU	disp5[ep],reg2	rrrrr0000111dddd Notes 18, 20	adr←ep+zero-extend(disp5) GR[reg2]←zero-extend(Load-memory(adr,Half-word))	1	1	Note 9					
SLD.W	disp8[ep],reg2	rrrrr1010dddddd0 Note 21	adr←ep + zero-extend(disp8) GR[reg2]←Load-memory(adr,Word)	1	1	Note 9					
SST.B	reg2,disp7[ep]	rrrrr0111ddddddd	adr←ep + zero-extend(disp7) Store-memory(adr,GR[reg2],Byte)	1	1	1					
SST.H	reg2,disp8[ep]	rrrrr1001ddddddd Note 19	adr←ep + zero-extend(disp8) Store-memory(adr,GR[reg2],Half-word)	1	1	1					
SST.W	reg2,disp8[ep]	rrrrr1010dddddd1 Note 21	adr←ep + zero-extend(disp8) Store-memory(adr,GR[reg2],Word)	1	1	1					
ST.B	reg2,disp16[reg1]	rrrrr111010RRRRR dddddddddddddddd	adr←GR[reg1] + sign-extend(disp16) Store-memory(adr,GR[reg2],Byte)	1	1	1					
ST.H	reg2,disp16[reg1]	rrrrr111011RRRRR ddddddddddddddd0 Note 8	adr←GR[reg1] + sign-extend(disp16) Store-memory (adr,GR[reg2], Half-word)	1	1	1					

(4/4)

Mnemonic	Operand	Opcode	Operation	Execution Clock			Flags				
				i	r	l	CY	OV	S	Z	SAT

ST.W	reg2,disp16[reg1]	`rrrrr111011RRRRR`
		`ddddddddddddddd1`
		Note 8
			adr←GR[reg1] + sign-extend(disp16)
			Store-memory (adr,GR[reg2], Word)
				1	1	1					

STSR	regID,reg2	`rrrrr111111RRRRR`
		`0000000001000000`
			GR[reg2]←SR[regID]	1	1	1					

SUB	reg1,reg2	`rrrrr001101RRRRR`	GR[reg2]←GR[reg2]–GR[reg1]	1	1	1	×	×	×	×	
SUBR	reg1,reg2	`rrrrr001100RRRRR`	GR[reg2]←GR[reg1]–GR[reg2]	1	1	1	×	×	×	×	
SWITCH	reg1	`00000000010RRRRR`	adr←(PC+2) + (GR [reg1] logically shift left by 1) PC←(PC+2) + (sign-extend (Load-memory (adr,Half-word))) logically shift left by 1	5	5	5					
SXB	reg1	`00000000101RRRRR`	GR[reg1]←sign-extend (GR[reg1] (7 : 0))	1	1	1					
SXH	reg1	`00000000111RRRRR`	GR[reg1]←sign-extend (GR[reg1] (15 : 0))	1	1	1					

TRAP	vector	`00000111111iiiii`
		`0000000100000000`
			EIPC←PC+4 (Return PC)
			EIPSW←PSW
			ECR.EICC←Interrupt Code
			PSW.EP←1
			PSW.ID←1
			PC←00000040H (when vector is 00H to 0FH)
			00000050H (when vector is 10H to 1FH)
				3	3	3					

TST	reg1,reg2	`rrrrr001011RRRRR`	result←GR[reg2] AND GR[reg1]	1	1	1		0	×	×	

TST1	bit#3,disp16[reg1]	`11bbb111110RRRRR`
		`dddddddddddddddd`
			adr←GR[reg1] + sign-extend(disp16)
			Z flag←Not (Load-memory-bit (adr,bit#3))
				3
				Note 3
					3
					Note 3
						3
						Note 3
										×	

	reg2, [reg1]	`rrrrr111111RRRRR`
		`0000000011100110`
			adr←GR[reg1]
			Z flag←Not (Load-memory-bit (adr,reg2))
				3
				Note 3
					3
					Note 3
						3
						Note 3
										×	

XOR	reg1,reg2	`rrrrr001001RRRRR`	GR[reg2]←GR[reg2] XOR GR[reg1]	1	1	1		0	×	×	
XORI	imm16,reg1,reg2	`rrrrr110101RRRRR` `iiiiiiiiiiiiiiii`	GR[reg2]←GR[reg1] XOR zero-extend (imm16)	1	1	1		0	×	×	
ZXB	reg1	`00000000100RRRRR`	GR[reg1]←zero-extend (GR[reg1] (7 : 0))	1	1	1					
ZXH	reg1	`00000000110RRRRR`	GR[reg1]←zero-extend (GR[reg1] (15 : 0))	1	1	1					

Notes:

1. `dddddddd`: Higher 8 bits of disp9.

2. 3 clocks if the final instruction includes PSW write access.

3. If there is no wait state (3 + the number of read access wait states).

4. n is the total number of list X load registers. (According to the number of wait states. Also, if there are no wait states, n is the number of list X registers.)

5. RRRRR: other than 00000.

6. The lower half word data only are valid.

7. `ddddddddddddddddddddd`: The higher 21 bits of disp22.

8. `ddddddddddddddd`: The higher 15 bits of disp16.

9. According to the number of wait states (1 if there are no wait states).

10. b: bit 0 of disp16.

11. According to the number of wait states (2 if there are no wait states).

12. In this instruction, for convenience of mnemonic description, the source register is made reg2, but the reg1 field is used in the opcode. Therefore, the meaning of register specification in the mnemonic description and in the opcode differs from other instructions.
`rrrrr`= regID specification
RRRRR= reg2 specification

13. `iiiii`: Lower 5 bits of imm9.
`IIII`: Lower 4 bits of imm9.

14. In the case of reg2 = reg3 (the lower 32 bits of the results are not written in the register) or reg3 = r0 (the higher 32 bits of the results are not written in the register), shortened by 1 clock.

15. sp/imm: specified by bits 19 and 20 of the sub-opcode.

16. `ff = 00`: Load sp in ep.
`10`: Load sign expanded 16-bit immediate data (bits 47 to 32) in ep.
`11`: Load 32-bit immediate data (bits 63 to 32) in ep.
17. If imm = imm32, n + 3 clocks.
18. `rrrrr` : Other than 00000.
19. `ddddddd`: Higher 7 bits of disp8.
20. `dddd`: Higher 4 bits of disp5.
21. `dddddd`: Higher 6 bits of disp8.

Appendix B Index

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6465-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

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

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-2719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Semiconductor Technical Hotline
Fax: 044-548-7900

I would like to report the following error/make the following suggestion:

Document title: ____________________

Document number: ____________________ Page number: __________

____

____

____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	□	□	□	□
Technical Accuracy	□	□	□	□
Organization	□	□	□	□