User's Manual

32

The revision list summarizes the locations of revisions and additions. Details should always be checked by referring to the relevant text.

H8SX/1720S Group

User's Manual: Hardware

Renesas 32-Bit CISC Microcomputer
H8SX Family / H8SX/1700 Series

H8SX/1727S R5F61727S
H8SX/1725S R5F61725S

Renesas Electronics
www.renesas.com

Rev.2.00 Jun 2012

Notice

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

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

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

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

(2012.4)

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

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

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

2. Processing at Power-on

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

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

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

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

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

4. Clock Signals

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

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

5. Differences between Products

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

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

How to Use This Manual

1. Objective and Target Users

   This manual was written to explain the hardware functions and electrical characteristics of this LSI to the target users, i.e. those who will be using this LSI in the design of application systems. Target users are expected to understand the fundamentals of electrical circuits, logic circuits, and microcomputers.

   This manual is organized in the following items: an overview of the product, descriptions of the CPU, system control functions, and peripheral functions, electrical characteristics of the device, and usage notes.

   > When designing an application system that includes this LSI, take all points to note into account. Points to note are given in their contexts and at the final part of each section, and in the section giving usage notes.

   > The list of revisions is a summary of major points of revision or addition for earlier versions. It does not cover all revised items. For details on the revised points, see the actual locations in the manual.

   The following documents have been prepared for the H8SX/1720S Group. Before using any of the documents, please visit our web site to verify that you have the most up-to-date available version of the document.

Document Type	Contents	Document Title	Document No.
Data Sheet	Overview of hardware and electrical characteristics	—	—
User's Manual: Hardware	Hardware specifications (pin assignments, memory maps, peripheral specifications, electrical characteristics, and timing charts) and descriptions of operation	H8SX/1720S Group User's Manual: Hardware	This user's manual
User's Manual: Software	Detailed descriptions of the CPU and instruction set	H8SX Family Software Manual	REJ09B0102
Application Note	Examples of applications and sample programs	The latest versions are available from our web site.	
Renesas Technical Update	Preliminary report on the specifications of a product, document, etc.		

2. Description of Numbers and Symbols

Aspects of the notations for register names, bit names, numbers, and symbolic names in this manual are explained below.

(1) Overall notation

In descriptions involving the names of bits and bit fields within this manual, the modules and registers to which the bits belong may be clarified by giving the names in the forms "module name"."register name"."bit name" or "register name"."bit name".

(2) Register notation

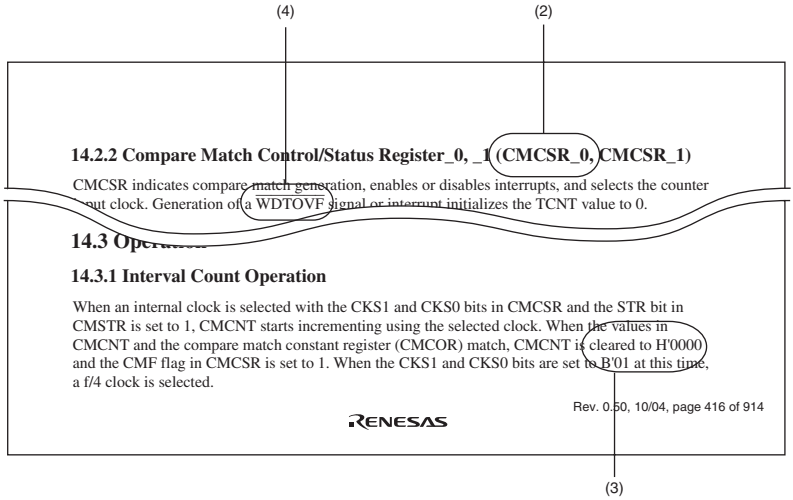
The style "register name"_"instance number" is used in cases where there is more than one instance of the same function or similar functions.
[Example] CMCSR_0: Indicates the CMCSR register for the compare-match timer of channel 0.

(3) Number notation

Binary numbers are given as B'nnnn (B' may be omitted if the number is obviously binary), hexadecimal numbers are given as H'nnnn or 0xnnnn, and decimal numbers are given as nnnn.
[Examples] Binary: B'11 or 11
Hexadecimal: H'EFA0 or 0xEFA0
Decimal: 1234

(4) Notation for active-low

An overbar on the name indicates that a signal or pin is active-low.
[Example] $\overline{\text{WDTOVF}}$

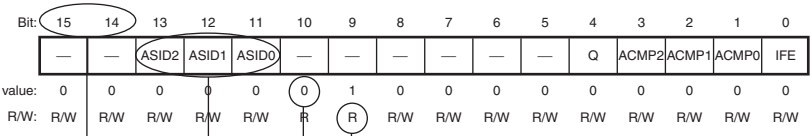


Note: The bit names and sentences in the above figure are examples and have nothing to do with the contents of this manual.

3. Description of Registers

Each register description includes a bit chart, illustrating the arrangement of bits, and a table of bits, describing the meanings of the bit settings. The standard format and notation for bit charts and tables are described below.

[Bit Chart]



[Table of Bits]

(1) (2) (3) (4) (5)

Bit	Bit Name	Initial Value	R/W	Description
15	–	0	R	Reserved
14	–	0	R	These bits are always read as 0.
13 to 11	ASID2 to ASID0	All 0	R/W	Address Identifier These bits enable or disable the pin function.
10	–	0	R	Reserved This bit is always read as 0.
9	–	1	R	Reserved This bit is always read as 1.
	–	0		

Note: The bit names and sentences in the above figure are examples, and have nothing to do with the contents of this manual.

(1) Bit
Indicates the bit number or numbers.
In the case of a 32-bit register, the bits are arranged in order from 31 to 0. In the case of a 16-bit register, the bits are arranged in order from 15 to 0.

(2) Bit name
Indicates the name of the bit or bit field.
When the number of bits has to be clearly indicated in the field, appropriate notation is included (e.g., ASID[3:0]).
A reserved bit is indicated by "–".
Certain kinds of bits, such as those of timer counters, are not assigned bit names. In such cases, the entry under Bit Name is blank.

(3) Initial value
Indicates the value of each bit immediately after a power-on reset, i.e., the initial value.
0: The initial value is 0
1: The initial value is 1
–: The initial value is undefined

(4) R/W
For each bit and bit field, this entry indicates whether the bit or field is readable or writable, or both writing to and reading from the bit or field are impossible.
The notation is as follows:

R/W: The bit or field is readable and writable.
R/(W): The bit or field is readable and writable.
However, writing is only performed to flag clearing.
R: The bit or field is readable.
"R" is indicated for all reserved bits. When writing to the register, write the value under Initial Value in the bit chart to reserved bits or fields.
W: The bit or field is writable.

(5) Description
Describes the function of the bit or field and specifies the values for writing.

4. Description of Abbreviations

The abbreviations used in this manual are listed below.

- Abbreviations specific to this product

Abbreviation	Description
BSC	Bus controller
CPG	Clock pulse generator
DTC	Data transfer controller
INTC	Interrupt controller
PPG	Programmable pulse generator
SCI	Serial communications interface
TMR	8-bit timer
TPU	16-bit timer pulse unit
RCAN	Controller Area Network
RSPI	Renesas Serial Peripheral Interface
WDT	Watchdog timer

- Abbreviations other than those listed above

Abbreviation	Description
ACIA	Asynchronous communications interface adapter
bps	Bits per second
CRC	Cyclic redundancy check
DMA	Direct memory access
DMAC	Direct memory access controller
GSM	Global System for Mobile Communications
Hi-Z	High impedance
IEBus	—
I/O	Input/output
IrDA	Infrared Data Association
LSB	Least significant bit
MSB	Most significant bit
NC	No connection
PLL	Phase-locked loop
PWM	Pulse width modulation
SFR	Special function register
SIM	Subscriber Identity Module
UART	Universal asynchronous receiver/transmitter
VCO	Voltage-controlled oscillator

Contents

Section 1 Overview

1.1 Features

- 32-bit high-speed H8SX V2 CPU
  Upward compatible with the H8/300 CPU, H8/300H CPU, H8S CPU, and H8SX V1 CPU at the object level
  Sixteen 16-bit general registers
  87 basic instructions
- Extensive peripheral functions
  DMA controller (DMAC)
  Data transfer controller (DTC)
  16-bit timer pulse unit (TPU)
  Programmable pulse generator (PPG)
  Watch dog timer (WDT)
  Serial communication interface (SCI) can be used in asynchronous and clocked synchronous mode
  Controller area network (RCAN-TL1)
  Renesas serial peripheral interface (RSPI)
  Hardware LIN (HWLIN)
  CRC operation circuit (CRC)
  10-bit A/D converter
  Clock pulse generator
- On-chip memory

Product Classification		Part Number	ROM (Flash Memory)	Data Flash	RAM	Operating Frequency
Flash memory version	H8SX/1727S	R5F61727S	512 Kbytes	32 Kbytes	40 Kbytes	8 MHz to 80 MHz
	H8SX/1725S	R5F61725S	256 Kbytes	16 Kbytes	24 Kbytes	

- General I/O port
  61 input/output port pins
  17 input-only port pins
- Supports power-down modes

- Small package

Package	Code	Body Size	Pin Pitch
LQFP-100	PLQP0100KB-A	14.0 × 14.0 mm	0.50 mm

1.2 Block Diagram

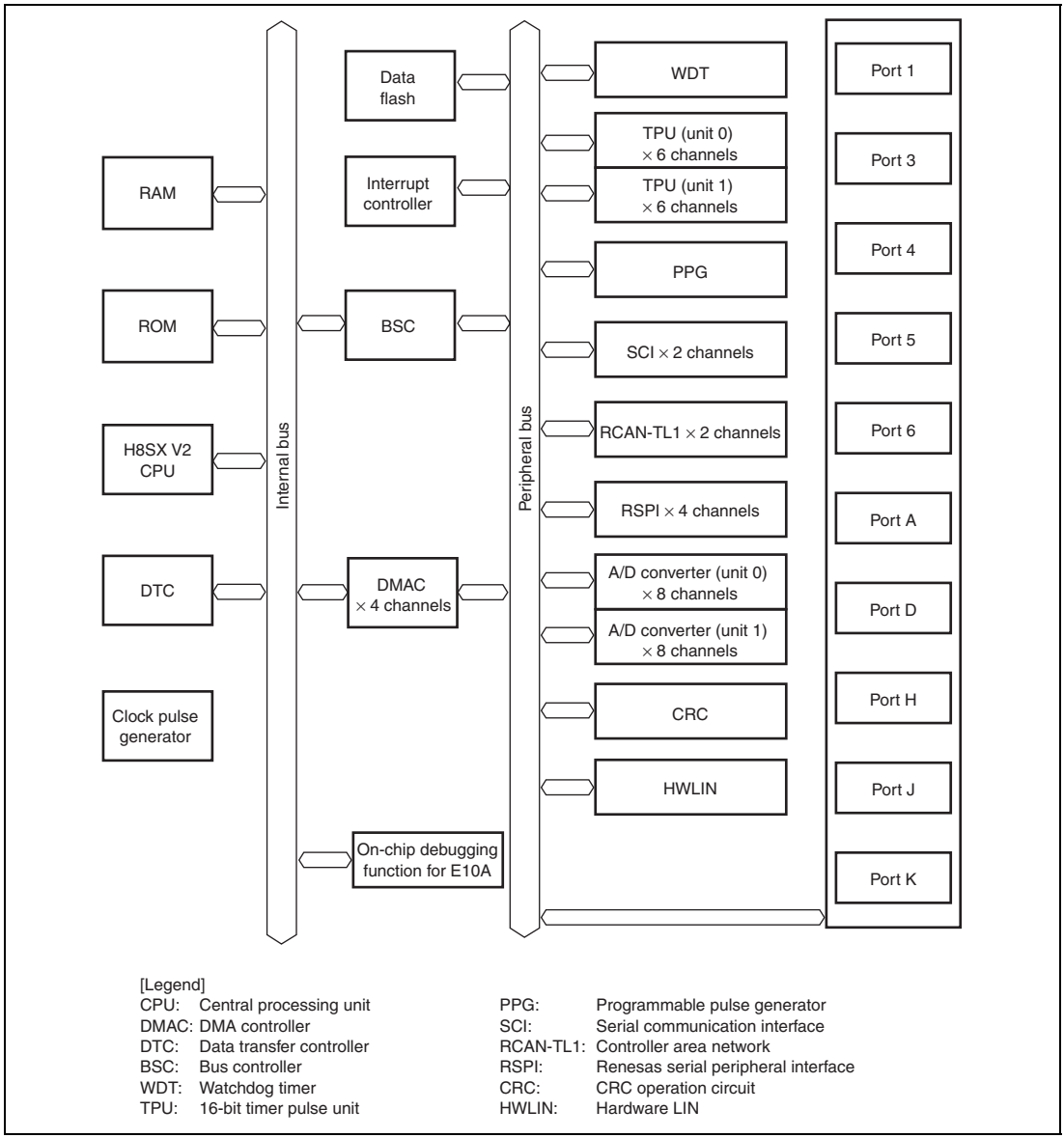
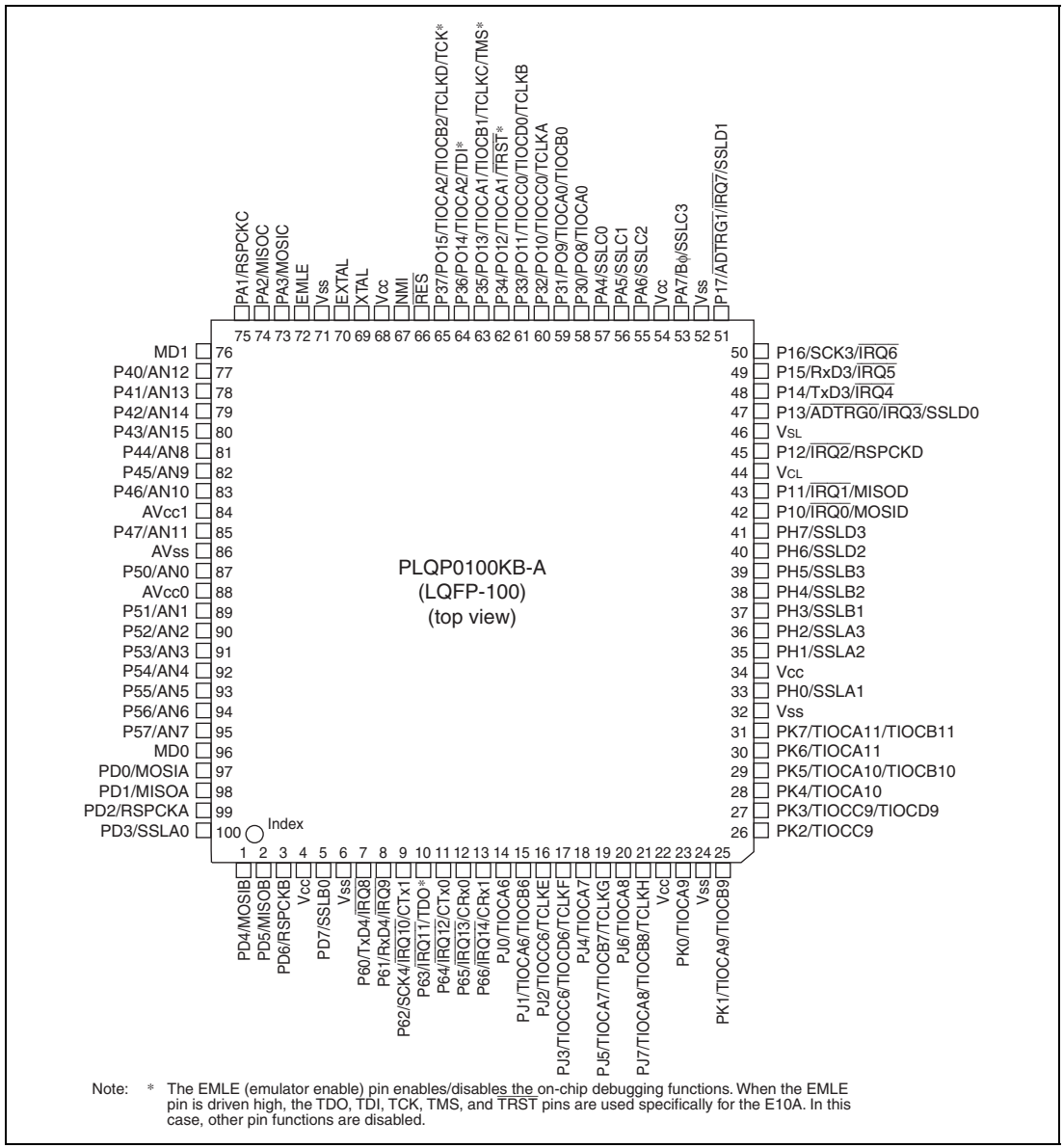


Figure 1.1 Block Diagram of H8SX/1725S and H8SX/1727S

1.3 Pin Description

1.3.1 Pin Assignments



Note: * The EMLE (emulator enable) pin enables/disables the on-chip debugging functions. When the EMLE pin is driven high, the TDO, TDI, TCK, TMS, and TRST pins are used specifically for the E10A. In this case, other pin functions are disabled.

Figure 1.2 Pin Assignments of H8SX/1725S and H8SX/1727S

1.3.2 Pin Configuration in Each Operating Mode

Table 1.1 Pin Configuration in Each Operating Mode

	Pin Name
Pin No.	**Mode 1, Mode 2, and Mode 3**
1	PD4/MOSIB
2	PD5/MISOB
3	PD6/RSPCKB
4	Vcc
5	PD7/SSLB0
6	Vss
7	P60/TxD4/$\overline{\text{IRQ8}}$
8	P61/RxD4/$\overline{\text{IRQ9}}$
9	P62/SCK4/$\overline{\text{IRQ10}}$/CTx1
10	P63/$\overline{\text{IRQ11}}$/TDO*
11	P64/$\overline{\text{IRQ12}}$/CTx0
12	P65/$\overline{\text{IRQ13}}$/CRx0
13	P66/$\overline{\text{IRQ14}}$/CRx1
14	PJ0/TIOCA6
15	PJ1/TIOCA6/TIOCB6
16	PJ2/TIOCC6/TCLKE
17	PJ3/TIOCC6/TIOCD6/TCLKF
18	PJ4/TIOCA7
19	PJ5/TIOCA7/TIOCB7/TCLKG
20	PJ6/TIOCA8
21	PJ7/TIOCA8/TIOCB8/TCLKH
22	Vcc
23	PK0/TIOCA9
24	Vss
25	PK1/TIOCA9/TIOCB9

Pin No.	Pin Name Mode 1, Mode 2, and Mode 3
26	PK2/TIOCC9
27	PK3/TIOCC9/TIOCD9
28	PK4/TIOCA10
29	PK5/TIOCA10/TIOCB10
30	PK6/TIOCA11
31	PK7/TIOCA11/TIOCB11
32	Vss
33	PH0/SSLA1
34	Vcc
35	PH1/SSLA2
36	PH2/SSLA3
37	PH3/SSLB1
38	PH4/SSLB2
39	PH5/SSLB3
40	PH6/SSLD2
41	PH7/SSLD3
42	P10/$\overline{\text{IRQ0}}$/MOSID
43	P11/$\overline{\text{IRQ1}}$/MISOD
44	V_{CL}
45	P12/$\overline{\text{IRQ2}}$/RSPCKD
46	V_{SL}
47	P13/$\overline{\text{ADTRG0}}$/$\overline{\text{IRQ3}}$/SSLD0
48	P14/TxD3/$\overline{\text{IRQ4}}$
49	P15/RxD3/$\overline{\text{IRQ5}}$
50	P16/SCK3/$\overline{\text{IRQ6}}$
51	P17/$\overline{\text{ADTRG1}}$/$\overline{\text{IRQ7}}$/SSLD1
52	Vss
53	PA7/Bϕ/SSLC3

	Pin Name
Pin No.	**Mode 1, Mode 2, and Mode 3**
54	Vcc
55	PA6/SSLC2
56	PA5/SSLC1
57	PA4/SSLC0
58	P30/PO8/TIOCA0
59	P31/PO9/TIOCA0/TIOCB0
60	P32/PO10/TIOCC0/TCLKA
61	P33/PO11/TIOCC0/TIOCD0/TCLKB
62	P34/PO12/TIOCA1/$\overline{\text{TRST}}$*
63	P35/PO13/TIOCA1/TIOCB1/TCLKC/TMS*
64	P36/PO14/TIOCA2/TDI*
65	P37/PO15/TIOCA2/TIOCB2/TCLKD/TCK*
66	$\overline{\text{RES}}$
67	NMI
68	Vcc
69	XTAL
70	EXTAL
71	Vss
72	EMLE
73	PA3/MOSIC
74	PA2/MISOC
75	PA1/RSPCKC
76	MD1
77	P40/AN12
78	P41/AN13
79	P42/AN14
80	P43/AN15
81	P44/AN8

Pin No.	Pin Name Mode 1, Mode 2, and Mode 3
82	P45/AN9
83	P46/AN10
84	AVcc1
85	P47/AN11
86	AVss
87	P50/AN0
88	AVcc0
89	P51/AN1
90	P52/AN2
91	P53/AN3
92	P54/AN4
93	P55/AN5
94	P56/AN6
95	P57/AN7
96	MD0
97	PD0/MOSIA
98	PD1/MISOA
99	PD2/RSPCKA
100	PD3/SSLA0

Note: * The EMLE (emulator enable) pin enables/disables the on-chip debugging functions. When the EMLE pin is driven high, the TDO, TDI, TCK, TMS, and $\overline{\text{TRST}}$ pins are used specifically for the E10A. In this case, other pin functions are disabled.

1.3.3 Pin Functions

Table 1.2 Pin Functions

Classification	Abbreviation	Pin Number	I/O	Description
Power supply	V_{CC}	4, 22, 34, 54, 68	Input	Power supply pins. Connect to the system power supply.
	V_{CL}	44	Input	Connect to V_{SL} via a 0.47-uF capacitor (place the capacitor close to this pin).
	V_{SL}	46	Input	Ground pin dedicated to V_{CL}
	V_{SS}	6, 24, 32, 52, 71	Input	Ground pins. Connect to the system power supply (0 V).
Clock	XTAL	69	Input	Pins for a crystal resonator. External clock can be input to the EXTAL pin. For a connection example, see section 22, Clock Pulse Generator.
	EXTAL	70	Input	
	Bϕ	53	Output	Supplies the system clock to external devices.
Operating mode control	MD1	76	Input	Pins for setting the operating mode. The signal levels of these pins must not be changed during operation.
	MD0	96		
System control	$\overline{RES}$	66	Input	Reset signal input pin. This LSI enters the reset state when this signal goes low.
	EMLE	72	Input	Input pin for on-chip emulator enable signal. Normally the signal level should be fixed low.

Classification	Abbreviation	Pin Number	I/O	Description
Interrupts	NMI	67	Input	Non-maskable interrupt request signal. When this pin is not in use, this signal must be fixed high.
	$\overline{\text{IRQ14}}$	13	Input	Maskable interrupt request signal.
	$\overline{\text{IRQ13}}$	12		
	$\overline{\text{IRQ12}}$	11		
	$\overline{\text{IRQ11}}$	10		
	$\overline{\text{IRQ10}}$	9		
	$\overline{\text{IRQ9}}$	8		
	$\overline{\text{IRQ8}}$	7		
	$\overline{\text{IRQ7}}$	51		
	$\overline{\text{IRQ6}}$	50		
	$\overline{\text{IRQ5}}$	49		
	$\overline{\text{IRQ4}}$	48		
	$\overline{\text{IRQ3}}$	47		
	$\overline{\text{IRQ2}}$	45		
	$\overline{\text{IRQ1}}$	43		
	$\overline{\text{IRQ0}}$	42		
On-chip emulator	$\overline{\text{TRST}}$	62	Input	Interface pins for debugging by the on-chip emulator.
	TMS	63	Input	
	TDO	10	Output	
	TDI	64	Input	
	TCK	65	Input	

Classification	Abbreviation	Pin Number	I/O	Description
16-bit timer pulse unit (TPU) (unit 0)	TCLKA	60	Input	Input pins for the external clocks.
	TCLKB	61		
	TCLKC	63		
	TCLKD	65		
	TIOCA0	58, 59	I/O	Signals for TGRA_0 to TGRD_0. These are used for the input capture inputs/output compare outputs/PWM outputs.
	TIOCB0	59		
	TIOCC0	60, 61		
	TIOCD0	61		
	TIOCA1	62, 63	I/O	Signals for TGRA_1 and TGRB_1. These are used for the input capture inputs/output compare outputs/PWM outputs.
	TIOCB1	63		
	TIOCA2	64, 65	I/O	Signals for TGRA_2 and TGRB_2. These are used for the input capture inputs/output compare outputs/PWM outputs.
	TIOCB2	65		
16-bit timer pulse unit (TPU) (unit 1)	TCLKE	16	Input	Input pins for the external clocks.
	TCLKF	17		
	TCLKG	19		
	TCLKH	21		
	TIOCA6	14, 15	I/O	Signals for TGRA_6 toTGRD_6. These are used for the input capture inputs/output compare outputs/PWM outputs.
	TIOCB6	15		
	TIOCC6	16, 17		
	TIOCD6	17		
	TIOCA7	18, 19	I/O	Signals for TGRA_7 toTGRB_7. These are used for the input capture inputs/output compare outputs/PWM outputs.
	TIOCB7	19		
	TIOCA8	20, 21	I/O	Signals for TGRA_8 toTGRB_8. These are used for the input capture inputs/output compare outputs/PWM outputs.
	TIOCB8	21		

Classification	Abbreviation	Pin Number	I/O	Description
16-bit timer pulse unit (TPU) (unit 1)	TIOCA9	23, 25	I/O	Signals for TGRA_9 to TGRD_9. These are used for the input capture inputs/output compare outputs/PWM outputs.
	TIOCB9	25		
	TIOCC9	26, 27		
	TIOCD9	27		
	TIOCA10	28, 29	I/O	Signals for TGRA_10 to TGRB_10. These are used for the input capture inputs/output compare outputs/PWM outputs.
	TIOCB10	29		
	TIOCA11	30, 31	I/O	Signals for TGRA_11 to TGRB_11. These are used for the input capture inputs/output compare outputs/PWM outputs.
	TIOCB11	31		
Programmable pulse generator (PPG)	PO15	65	Output	Output pins for the pulse signals.
	PO14	64		
	PO13	63		
	PO12	62		
	PO11	61		
	PO10	60		
	PO9	59		
	PO8	58		
Serial communication interface (SCI)	TxD3	48	Output	Output pins for transmit data.
	TxD4	7		
	RxD3	49	Input	Input pins for receive data.
	RxD4	8		
	SCK3	50	I/O	Input/output pins for clock signals.
	SCK4	9		
Controller area network (RCAN-TL1)	CTx0	11	Output	Output pins for CAN bus transmission.
	CTx1	9		
	CRx0	12	Input	Input pins for CAN bus reception.
	CRx1	13		

Classification	Abbreviation	Pin Number	I/O	Description
Renesas serial peripheral interface (RSPI)	MOSID	42	I/O	Input/output pins for data.
	MOSIC	73		
	MOSIB	1		
	MOSIA	97		
	MISOD	43	I/O	Input/output pins for data.
	MISOC	74		
	MISOB	2		
	MISOA	98		
	RSPCKD	45	I/O	Input/output pins for clock.
	RSPCKC	75		
	RSPCKB	3		
	RSPCKA	99		
	SSLA3	36	I/O	Input/output pins for chip select.
	SSLA2	35		
	SSLA1	33		
	SSLA0	100		
	SSLB3	39	I/O	Input/output pins for chip select.
	SSLB2	38		
	SSLB1	37		
	SSLB0	5		
	SSLC3	53	I/O	Input/output pins for chip select.
	SSLC2	55		
	SSLC1	56		
	SSLC0	57		
	SSLD3	41	I/O	Input/output pins for chip select.
	SSLD2	40		
	SSLD1	51		
	SSLD0	47		

Classification	Abbreviation	Pin Number	I/O	Description
A/D converter	AN15	80	Input	Input pins for the analog signals for the A/D converter.
	AN14	79		
	AN13	78		
	AN12	77		
	AN11	85		
	AN10	83		
	AN9	82		
	AN8	81		
	AN7	95		
	AN6	94		
	AN5	93		
	AN4	92		
	AN3	91		
	AN2	90		
	AN1	89		
	AN0	87		
	$\overline{\text{ADTRG0}}$	47	Input	Input pins for the external trigger signal to start A/D conversion.
	$\overline{\text{ADTRG1}}$	51		
	$AV_{CC}0$	88	Input	Analog power supply and reference power supply pins for the A/D converter. When the A/D converter is not in use, connect to the system power supply.
	$AV_{CC}1$	84		
	AV_{SS}	86	Input	Ground pin for the A/D and D/A converters. Connect to the system power supply (0 V).

Classification	Abbreviation	Pin Number	I/O	Description
I/O port	P17	51	I/O	8-bit input/output pins.
	P16	50		
	P15	49		
	P14	48		
	P13	47		
	P12	45		
	P11	43		
	P10	42		
	P37	65	I/O	8-bit input/output pins.
	P36	64		
	P35	63		
	P34	62		
	P33	61		
	P32	60		
	P31	59		
	P30	58		
	P47	85	Input	8-bit input pins.
	P46	83		
	P45	82		
	P44	81		
	P43	80		
	P42	79		
	P41	78		
	P40	77		
	P57	95	Input	8-bit input pins.
	P56	94		
	P55	93		
	P54	92		
	P53	91		
	P52	90		
	P51	89		
	P50	87		

Classification	Abbreviation	Pin Number	I/O	Description
I/O port	P66	13	I/O	7-bit input/output pins.
	P65	12		
	P64	11		
	P63	10		
	P62	9		
	P61	8		
	P60	7		
	PA7	53	Input	1-bit input pin.
	PA6	55	I/O	6-bit input/output pins.
	PA5	56		
	PA4	57		
	PA3	73		
	PA2	74		
	PA1	75		
	PD7	5	I/O	8-bit input/output pins.
	PD6	3		
	PD5	2		
	PD4	1		
	PD3	100		
	PD2	99		
	PD1	98		
	PD0	97		
	PH7	41	I/O	8-bit input/output pins.
	PH6	40		
	PH5	39		
	PH4	38		
	PH3	37		
	PH2	36		
	PH1	35		
	PH0	33		

Classification	Abbreviation	Pin Number	I/O	Description
I/O port	PJ7	21	I/O	8-bit input/output pins.
	PJ6	20		
	PJ5	19		
	PJ4	18		
	PJ3	17		
	PJ2	16		
	PJ1	15		
	PJ0	14		
	PK7	31	I/O	8-bit input/output pins.
	PK6	30		
	PK5	29		
	PK4	28		
	PK3	27		
	PK2	26		
	PK1	25		
	PK0	23		

Section 2 CPU

The H8SX CPU is a high-speed CPU with an internal 32-bit architecture that is upward compatible with the H8/300, H8/300H, and H8S CPUs.

The H8SX CPU has sixteen 16-bit general registers, can handle a 4-Gbyte linear address space, and is ideal for a realtime control system.

2.1 Features

- Upward-compatible with H8/300, H8/300H, and H8S CPUs
  - Can execute H8/300, H8/300H, and H8S/2000 object programs
- Sixteen 16-bit general registers
  - Also usable as sixteen 8-bit registers or eight 32-bit registers
- 87 basic instructions
  - 8/16/32-bit arithmetic and logic instructions
  - Multiply and divide instructions
  - Bit field transfer instructions
  - Powerful bit-manipulation instructions
  - Bit condition branch instructions
  - Multiply-and-accumulate instruction
- Eleven addressing modes
  - Register direct [Rn]
  - Register indirect [@ERn]
  - Register indirect with displacement [@(d:2,ERn), @(d:16,ERn), or @(d:32,ERn)]
  - Index register indirect with displacement [@(d:16,RnL.B), @(d:32,RnL.B), @(d:16,Rn.W), @(d:32,Rn.W), @(d:16,ERn.L), or @(d:32,ERn.L)]
  - Register indirect with pre-/post-increment or pre-/post-decrement [@+ERn, @–ERn, @ERn+, or @ERn–]
  - Absolute address [@aa:8, @aa:16, @aa:24, or @aa:32]
  - Immediate [#xx:3, #xx:4, #xx:8, #xx:16, or #xx:32]
  - Program-counter relative [@(d:8,PC) or @(d:16,PC)]
  - Program-counter relative with index register [@(RnL.B,PC), @(Rn.W,PC), or @(ERn.L,PC)]
  - Memory indirect [@@aa:8]
  - Extended memory indirect [@@vec:7]

- Two base registers
  - Vector base register
  - Short address base register
- 4-Gbyte address space
  - Program: 4 Gbytes
  - Data: 4 Gbytes
- High-speed operation
  - All frequently-used instructions executed in one or two states
  - 8/16/32-bit register-register add/subtract: 1 state
  - 8 × 8-bit register-register multiply: 1 state
  - 16 ÷ 8-bit register-register divide: 10 states
  - 16 × 16-bit register-register multiply: 1 state
  - 32 ÷ 16-bit register-register divide: 18 states
  - 32 × 32-bit register-register multiply: 5 states
  - 32 ÷ 32-bit register-register divide: 18 states
- Four CPU operating modes
  - Normal mode
  - Middle mode
  - Advanced mode
  - Maximum mode
- Power-down modes
  - Transition is made by execution of SLEEP instruction
  - Choice of CPU operating clocks

Notes: 1 The CPUs of the H8SX/1720S Group can only operate in advanced mode; that is, normal mode, middle mode, and maximum mode are not available.
2 The H8SX/1720S Group supports the multiplier and divider.

2.2 CPU Operating Modes

The H8SX CPU has four operating modes: normal, middle, advanced and maximum modes. For details on mode settings, see section 3.1, Operating Mode Selection.

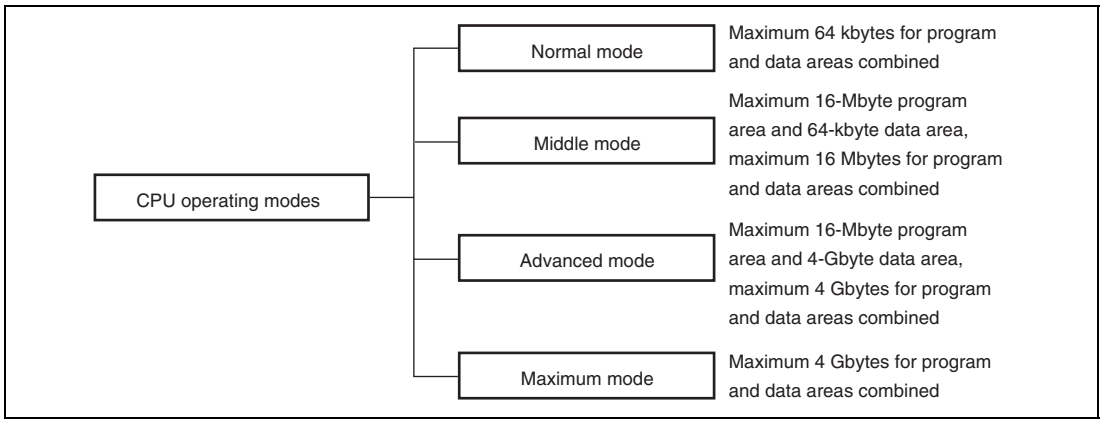


Figure 2.1 CPU Operating Modes

2.2.1 Normal Mode

The exception vector table and stack have the same structure as in the H8/300 CPU.

Note: Normal mode is not supported in this LSI.

- Address Space

  The maximum address space of 64 kbytes can be accessed.
- Extended Registers (En)

  The extended registers (E0 to E7) can be used as 16-bit registers, or as the upper 16-bit segments of 32-bit registers. When the extended register En is used as a 16-bit register it can contain any value, even when the corresponding general register Rn is used as an address register. (If the general register Rn is referenced in the register indirect addressing mode with pre-/post-increment or pre-/post-decrement and a carry or borrow occurs, however, the value in the corresponding extended register En will be affected.)
- Instruction Set

  All instructions and addressing modes can be used. Only the lower 16 bits of effective addresses (EA) are valid.

- Exception Vector Table and Memory Indirect Branch Addresses

  In normal mode, the top area starting at H'0000 is allocated to the exception vector table. One branch address is stored per 16 bits. The structure of the exception vector table is shown in figure 2.2.

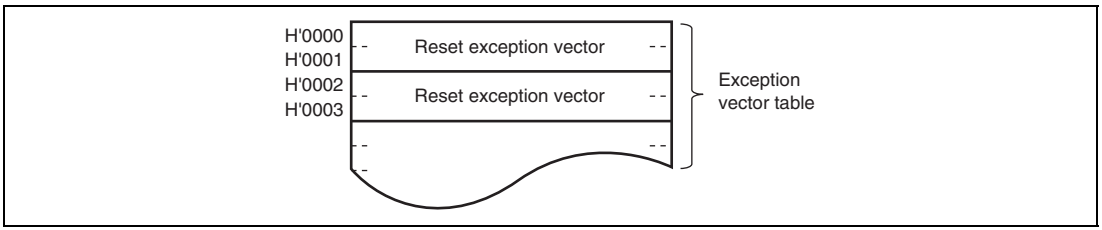


Figure 2.2 Exception Vector Table (Normal Mode)

The memory indirect (@@aa:8) and extended memory indirect (@@vec:7) addressing modes are used in the JMP and JSR instructions. An 8-bit absolute address included in the instruction code specifies a memory location. Execution branches to the contents of the memory location.

- Stack Structure

  The stack structure of PC at a subroutine branch and that of PC and CCR at an exception handling are shown in figure 2.3. The PC contents are saved or restored in 16-bit units.

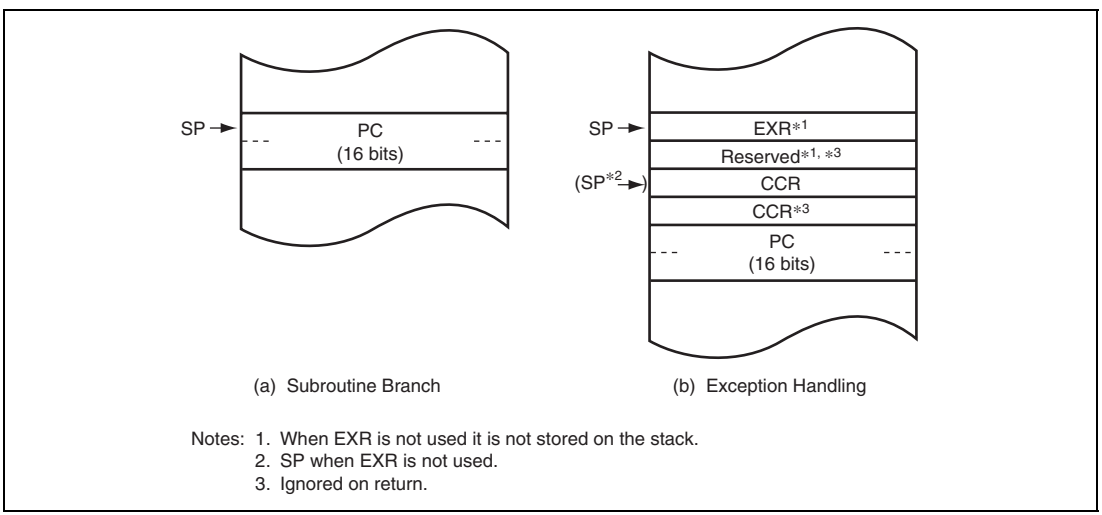


Figure 2.3 Stack Structure (Normal Mode)

Note: This mode is not supported in the H8SX/1720S Group.

2.2.2 Middle Mode

The program area in middle mode is extended to 16 Mbytes as compared with that in normal mode.

- Address Space

  The maximum address space of 16 Mbytes can be accessed as a total of the program and data areas. For individual areas, up to 16 Mbytes of the program area or up to 64 kbytes of the data area can be allocated.
- Extended Registers (En)

  The extended registers (E0 to E7) can be used as 16-bit registers, or as the upper 16-bit segments of 32-bit registers. When the extended register En is used as a 16-bit register (in other than the JMP and JSR instructions), it can contain any value even when the corresponding general register Rn is used as an address register. (If the general register Rn is referenced in the register indirect addressing mode with pre-/post-increment or pre-/post-decrement and a carry or borrow occurs, however, the value in the corresponding extended register En will be affected.)
- Instruction Set

  All instructions and addressing modes can be used. Only the lower 16 bits of effective addresses (EA) are valid and the upper eight bits are sign-extended.
- Exception Vector Table and Memory Indirect Branch Addresses

  In middle mode, the top area starting at H'000000 is allocated to the exception vector table. One branch address is stored per 32 bits. The upper eight bits are ignored and the lower 24 bits are stored. The structure of the exception vector table is shown in figure 2.4.

  The memory indirect (@@aa:8) and extended memory indirect (@@vec:7) addressing modes are used in the JMP and JSR instructions. An 8-bit absolute address included in the instruction code specifies a memory location. Execution branches to the contents of the memory location.

  In middle mode, an operand is a 32-bit (longword) operand, providing a 32-bit branch address. The upper eight bits are reserved and assumed to be H'00.
- Stack Structure

  The stack structure of PC at a subroutine branch and that of PC and CCR at an exception handling are shown in figure 2.5. The PC contents are saved or restored in 24-bit units.

Note: This mode is not supported in the H8SX/1720S Group.

2.2.3 Advanced Mode

The data area is extended to 4 Gbytes as compared with that in middle mode.

- Address Space

  The maximum address space of 4 Gbytes can be linearly accessed. For individual areas, up to 16 Mbytes of the program area and up to 4 Gbytes of the data area can be allocated.
- Extended Registers (En)

  The extended registers (E0 to E7) can be used as 16-bit registers, or as the upper 16-bit segments of 32-bit registers or address registers.
- Instruction Set

  All instructions and addressing modes can be used.
- Exception Vector Table and Memory Indirect Branch Addresses

  In advanced mode, the top area starting at H'00000000 is allocated to the exception vector table. One branch address is stored per 32 bits. The upper eight bits are ignored and the lower 24 bits are stored. The structure of the exception vector table is shown in figure 2.4.

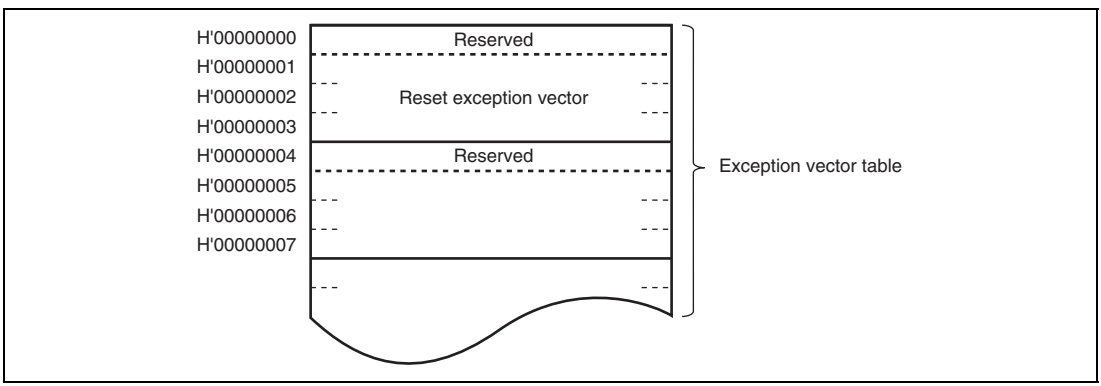


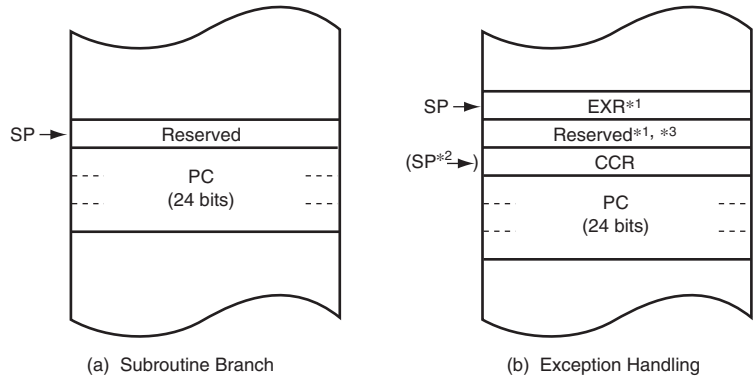
Figure 2.4 Exception Vector Table (Middle and Advanced Modes)

The memory indirect (@@aa:8) and extended memory indirect (@@vec:7) addressing modes are used in the JMP and JSR instructions. An 8-bit absolute address included in the instruction code specifies a memory location. Execution branches to the contents of the memory location.

In advanced mode, an operand is a 32-bit (longword) operand, providing a 32-bit branch address. The upper eight bits are reserved and assumed to be H'00.

- Stack Structure

  The stack structure of PC at a subroutine branch and that of PC and CCR at an exception handling are shown in figure 2.5. The PC contents are saved or restored in 24-bit units.



Notes: 1. When EXR is not used it is not stored on the stack.
2. SP when EXR is not used.
3. Ignored on return.

Figure 2.5 Stack Structure (Middle and Advanced Modes)

2.2.4 Maximum Mode

The program area is extended to 4 Gbytes as compared with that in advanced mode.

- Address Space

  The maximum address space of 4 Gbytes can be linearly accessed.
- Extended Registers (En)

  The extended registers (E0 to E7) can be used as 16-bit registers or as the upper 16-bit segments of 32-bit registers or address registers.
- Instruction Set

  All instructions and addressing modes can be used.
- Exception Vector Table and Memory Indirect Branch Addresses

  In maximum mode, the top area starting at H'00000000 is allocated to the exception vector table. One branch address is stored per 32 bits. The structure of the exception vector table is shown in figure 2.6.

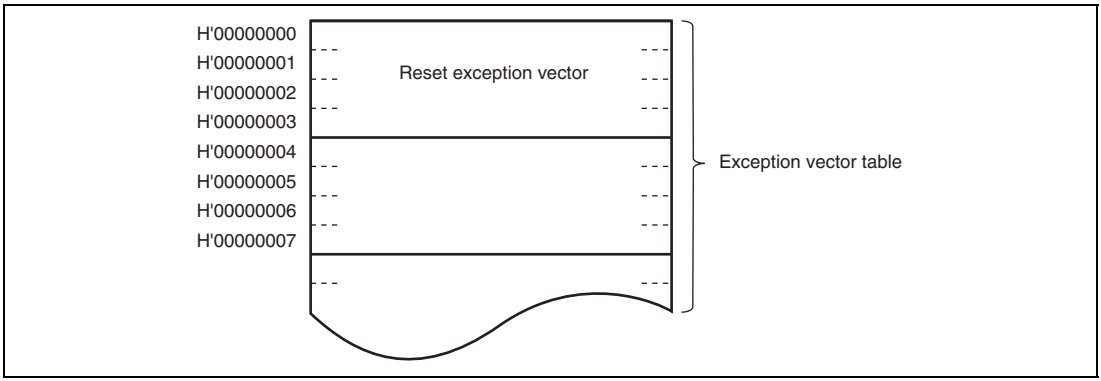


Figure 2.6 Exception Vector Table (Maximum Modes)

The memory indirect (@@aa:8) and extended memory indirect (@@vec:7) addressing modes are used in the JMP and JSR instructions. An 8-bit absolute address included in the instruction code specifies a memory location. Execution branches to the contents of the memory location.

In maximum mode, an operand is a 32-bit (longword) operand, providing a 32-bit branch address.

- Stack Structure

  The stack structure of PC at a subroutine branch and that of PC and CCR at an exception handling are shown in figure 2.7. The PC contents are saved or restored in 32-bit units. The EXR contents are saved or restored regardless of whether or not EXR is in use.

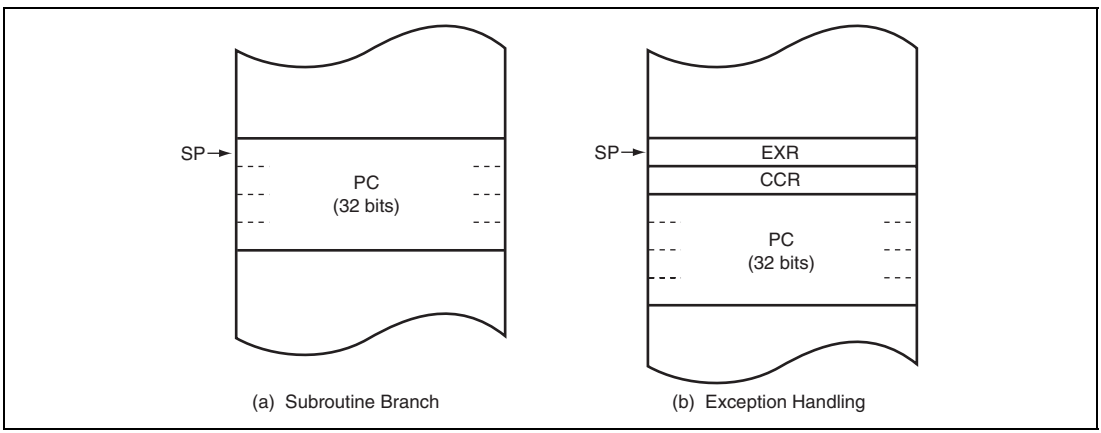


Figure 2.7 Stack Structure (Maximum Mode)

Note: This mode is not supported in the H8SX/1720S Group.

2.3 Instruction Fetch

The H8SX CPU has two modes for instruction fetch: 16-bit and 32-bit modes. It is recommended that the mode be set according to the bus width of the memory in which a program is stored. The instruction-fetch mode setting does not affect operation other than instruction fetch such as data accesses.

Note: The instruction fetch in the H8SX/1720S Group is 32-bit mode.

2.4 Address Space

Figure 2.8 shows a memory map of the H8SX CPU. The address space differs depending on the CPU operating mode.

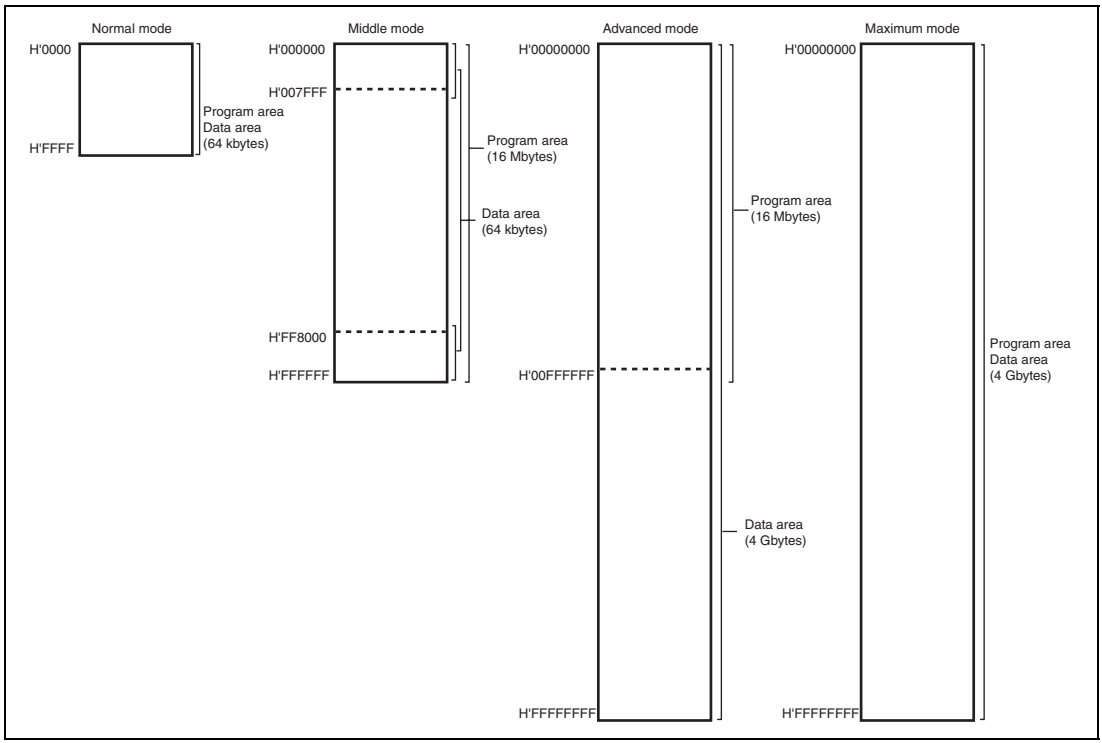


Figure 2.8 Memory Map

2.5 Registers

The H8SX CPU has the internal registers shown in figure 2.9. There are two types of registers: general registers and control registers. The control registers are the 32-bit program counter (PC), 8-bit extended control register (EXR), 8-bit condition-code register (CCR), 32-bit vector base register (VBR), 32-bit short address base register (SBR), and 64-bit multiply-accumulate register (MAC).

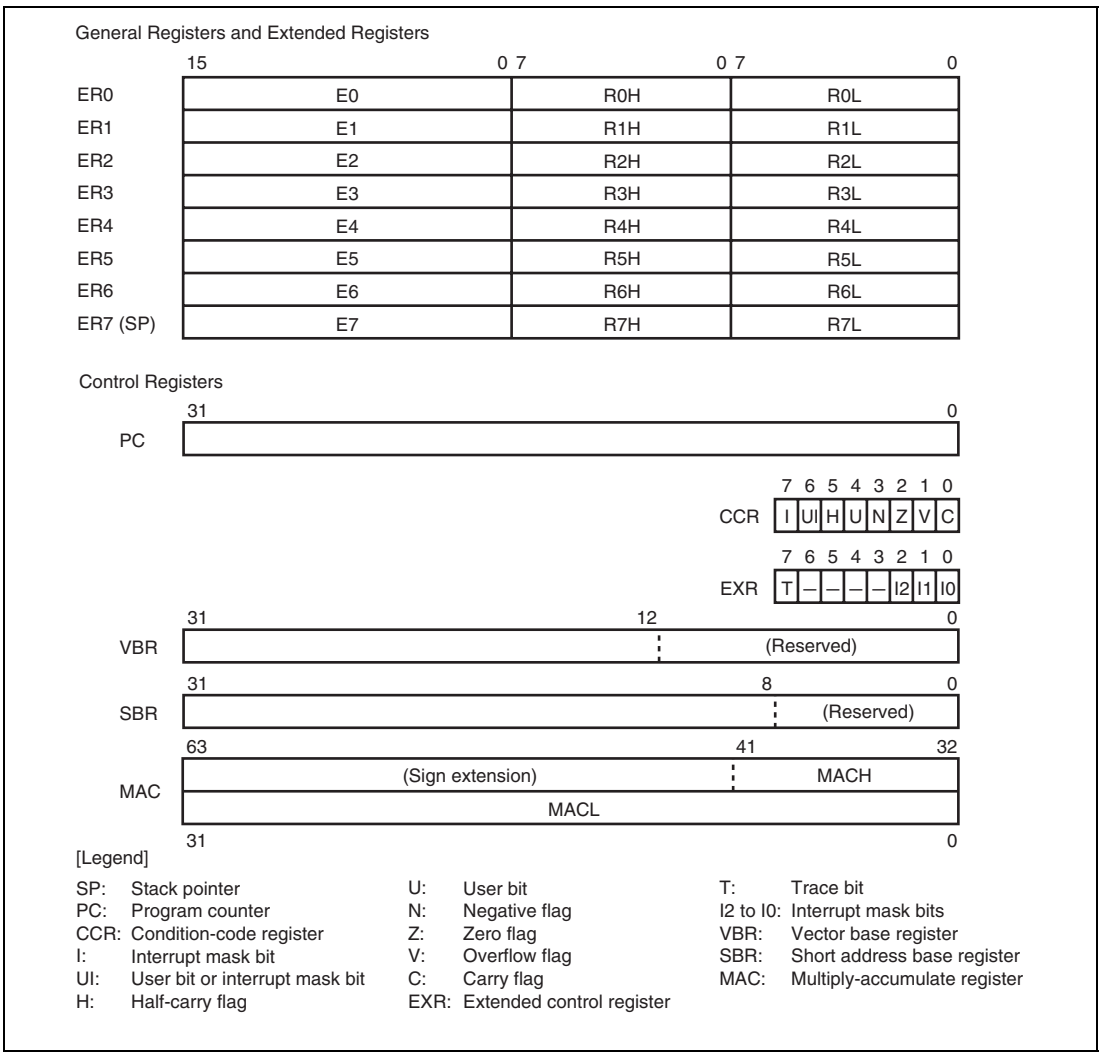


Figure 2.9 CPU Registers

2.5.1 General Registers

The H8SX CPU has eight 32-bit general registers. These general registers are all functionally alike and can be used as both address registers and data registers. When a general register is used as a data register, it can be accessed as a 32-bit, 16-bit, or 8-bit register. Figure 2.10 illustrates the usage of the general registers.

When the general registers are used as 32-bit registers or address registers, they are designated by the letters ER (ER0 to ER7).

When the general registers are used as 16-bit registers, the ER registers are divided into 16-bit general registers designated by the letters E (E0 to E7) and R (R0 to R7). These registers are functionally equivalent, providing a maximum sixteen 16-bit registers. The E registers (E0 to E7) are also referred to as extended registers.

When the general registers are used as 8-bit registers, the R registers are divided into 8-bit general registers designated by the letters RH (R0H to R7H) and RL (R0L to R7L). These registers are functionally equivalent, providing a maximum sixteen 8-bit registers.

The general registers ER (ER0 to ER7), R (R0 to R7), and RL (R0L to R7L) are also used as index registers. The size in the operand field determines which register is selected.

The usage of each register can be selected independently.

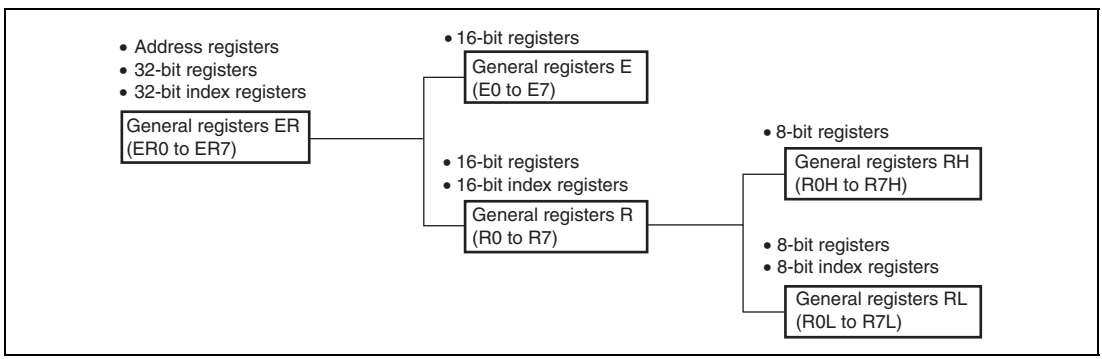


Figure 2.10 Usage of General Registers

General register ER7 has the function of stack pointer (SP) in addition to its general-register function, and is used implicitly in exception handling and subroutine branches. Figure 2.11 shows the stack.

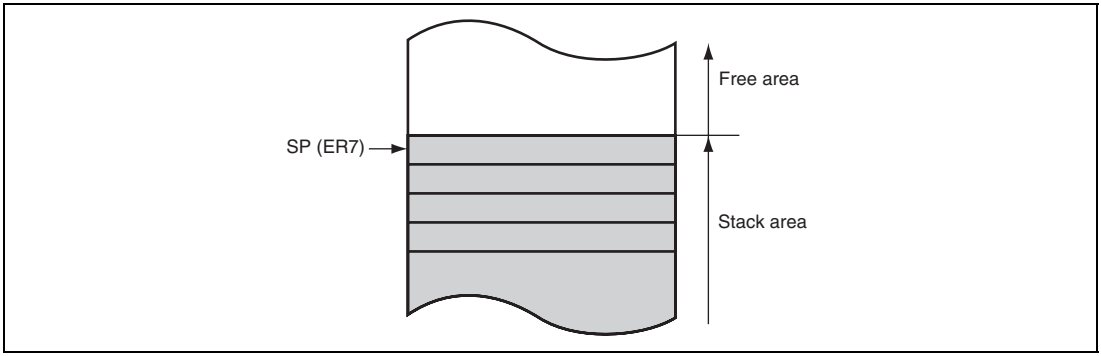


Figure 2.11 Stack

2.5.2 Program Counter (PC)

PC is a 32-bit counter that indicates the address of the next instruction the CPU will execute. The length of all CPU instructions is two bytes (one word), so the least significant bit is ignored.

2.5.3 Condition-Code Register (CCR)

CCR is an 8-bit register that contains internal CPU status information, including an interrupt mask (I) and user (UI, U) bits and half-carry (H), negative (N), zero (Z), overflow (V), and carry (C) flags.

Operations can be performed on the CCR bits by the LDC, STC, ANDC, ORC, and XORC instructions. The N, Z, V, and C flags are used as branch conditions for conditional branch (Bcc) instructions.

Bit	7	6	5	4	3	2	1	0
Bit Name	I	UI	H	U	N	Z	V	C
Initial Value	1	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	I	1	R/W	Interrupt Mask Bit Masks interrupts when set to 1. This bit is set to 1 at the start of an exception handling.
6	UI	Undefined	R/W	User Bit/Interrupt Mask Bit Can be written to and read from by software using the LDC, STC, ANDC, ORC, and XORC instructions. This bit can also be used as an interrupt mask bit.
5	H	Undefined	R/W	Half-Carry Flag When the ADD.B, ADDX.B, SUB.B, SUBX.B, CMP.B, or NEG.B instruction is executed, this flag is set to 1 if there is a carry or borrow at bit 3, and cleared to 0 otherwise. When the ADD.W, SUB.W, CMP.W, or NEG.W instruction is executed, this flag is set to 1 if there is a carry or borrow at bit 11, and cleared to 0 otherwise. When the ADD.L, SUB.L, CMP.L, or NEG.L instruction is executed, this flag is set to 1 if there is a carry or borrow at bit 27, and cleared to 0 otherwise.
4	U	Undefined	R/W	User Bit Can be written to and read from by software using the LDC, STC, ANDC, ORC, and XORC instructions.
3	N	Undefined	R/W	Negative Flag Stores the value of the most significant bit (regarded as sign bit) of data.

Bit	Bit Name	Initial Value	R/W	Description
2	Z	Undefined	R/W	Zero Flag Set to 1 to indicate zero data, and cleared to 0 to indicate non-zero data.
1	V	Undefined	R/W	Overflow Flag Set to 1 when an arithmetic overflow occurs, and cleared to 0 otherwise.
0	C	Undefined	R/W	Carry Flag Set to 1 when a carry occurs, and cleared to 0 otherwise. A carry has the following types: • Carry from the result of addition • Borrow from the result of subtraction • Carry from the result of shift or rotation The carry flag is also used as a bit accumulator by bit manipulation instructions.

2.5.4 Extended Control Register (EXR)

EXR is an 8-bit register that contains the trace bit (T) and three interrupt mask bits (I2 to I0).

Operations can be performed on the EXR bits by the LDC, STC, ANDC, ORC, and XORC instructions.

For details, see section 4, Exception Handling.

Bit	7	6	5	4	3	2	1	0
Bit Name	T	–	–	–	–	I2	I1	I0
Initial Value	0	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	T	0	R/W	Trace Bit When this bit is set to 1, a trace exception is generated each time an instruction is executed. When this bit is cleared to 0, instructions are executed in sequence.

Bit	Bit Name	Initial Value	R/W	Description
6 to 3	—	All 1	R/W	Reserved These bits are always read as 1.
2	I2	1	R/W	Interrupt Mask Bits
1	I1	1	R/W	These bits designate the interrupt mask level (0 to 7).
0	I0	1	R/W	

2.5.5 Vector Base Register (VBR)

VBR is a 32-bit register in which the upper 20 bits are valid. The lower 12 bits of this register are read as 0s. This register is a base address of the vector area for exception handlings other than a reset and a CPU address error (extended memory indirect is also out of the target). The initial value is H'00000000. The VBR contents are changed with the LDC and STC instructions.

2.5.6 Short Address Base Register (SBR)

SBR is a 32-bit register in which the upper 24 bits are valid. The lower eight bits are read as 0s. In 8-bit absolute address addressing mode (@aa:8), this register is used as the upper address. The initial value is H'FFFFFF00. The SBR contents are changed with the LDC and STC instructions.

2.5.7 Multiply-Accumulate Register (MAC)

MAC is a 64-bit register that stores the results of multiply-and-accumulate operations. It consists of two 32-bit registers denoted MACH and MACL. The lower 10 bits of MACH are valid; the upper bits are sign extended. The MAC contents are changed with the MAC, CLRMAC, LDMAC, and STMAC instructions.

2.5.8 Initial Values of CPU Registers

Reset exception handling loads the start address from the vector table into the PC, clears the T bit in EXR to 0, and sets the I bits in CCR and EXR to 1. The general registers, MAC, and the other bits in CCR are not initialized. In particular, the initial value of the stack pointer (ER7) is undefined. The SP should therefore be initialized using an MOV.L instruction executed immediately after a reset.

2.6 Data Formats

The H8SX CPU can process 1-bit, 4-bit BCD, 8-bit (byte), 16-bit (word), and 32-bit (longword) data.

Bit-manipulation instructions operate on 1-bit data by accessing bit n (n = 0, 1, 2, …, 7) of byte operand data. The DAA and DAS decimal-adjust instructions treat byte data as two digits of 4-bit BCD data.

2.6.1 General Register Data Formats

Figure 2.12 shows the data formats in general registers.

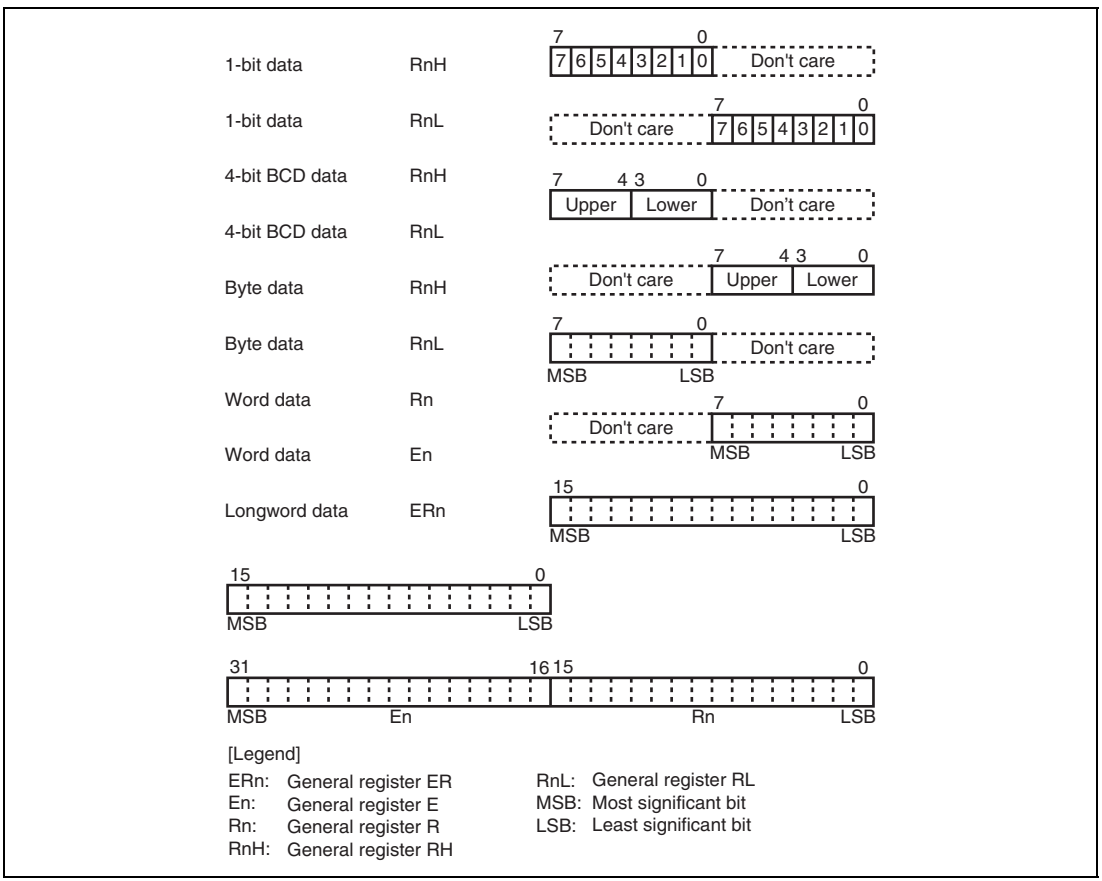


Figure 2.12 General Register Data Formats

2.6.2 Memory Data Formats

Figure 2.13 shows the data formats in memory.

The H8SX CPU can access word data and longword data which are stored at any addresses in memory. When word data begins at an odd address or longword data begins at an address other than a multiple of 4, a bus cycle is divided into two or more accesses. For example, when longword data begins at an odd address, the bus cycle is divided into byte, word, and byte accesses. In this case, these accesses are assumed to be individual bus cycles.

However, instructions to be fetched, word and longword data to be accessed during execution of the stack manipulation, branch table manipulation, block transfer instructions, and MAC instruction should be located to even addresses.

When SP (ER7) is used as an address register to access the stack, the operand size should be word size or longword size.

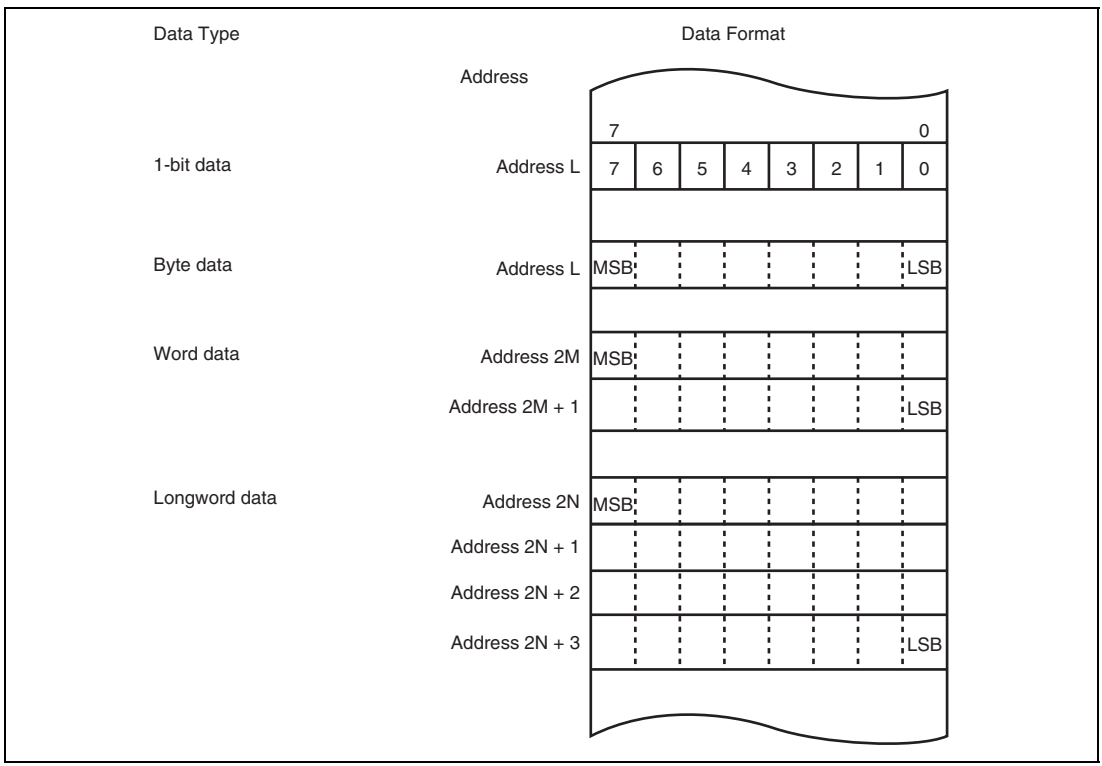


Figure 2.13 Memory Data Formats

2.7 Instruction Set

The H8SX CPU has 87 types of instructions. The instructions are classified by function as shown in table 2.1. The arithmetic operation, logic operation, shift, and bit manipulation instructions are called operation instruction in this manual.

Table 2.1 Instruction Classification

Function	Instructions	Size	Types
Data transfer	MOV	B/W/L	6
	MOVFPE*[6], MOVTPE*[6]	B	
	POP, PUSH*[1]	W/L	
	LDM, STM	L	
	MOVA	B/W*[2]	
Block transfer	EEPMOV	B	3
	MOVMD	B/W/L	
	MOVSD	B	
Arithmetic operations	ADD, ADDX, SUB, SUBX, CMP, NEG, INC, DEC	B/W/L	27
	DAA, DAS	B	
	ADDS, SUBS	L	
	MULXU, DIVXU, MULXS, DIVXS	B/W	
	MULU, DIVU, MULS, DIVS	W/L	
	MULU/U, MULS/U	L	
	EXTU, EXTS	W/L	
	TAS	B	
	MAC	—	
	LDMAC, STMAC	—	
	CLRMAC	—	
Logic operations	AND, OR, XOR, NOT	B/W/L	4
Shift	SHLL, SHLR, SHAL, SHAR, ROTL, ROTR, ROTXL, ROTXR	B/W/L	8
Bit manipulation	BSET, BCLR, BNOT, BTST, BAND, BIAND, BOR, BIOR, BXOR, BIXOR, BLD, BILD, BST, BIST	B	20
	BSET/EQ, BSET/NE, BCLR/EQ, BCLR/NE, BSTZ, BISTZ	B	
	BFLD, BFST	B	

Function	Instructions	Size	Types
Branch	BRA/BS, BRA/BC, BSR/BS, BSR/BC	B*3	9
	Bcc*4, JMP, BSR, JSR, RTS	—	
	RTS/L	L*5	
	BRA/S	—	
System control	TRAPA, RTE, SLEEP, NOP	—	10
	RTE/L	L*5	
	LDC, STC, ANDC, ORC, XORC	B/W/L	
		Total	87

[Legend]

B: Byte size

W: Word size

L: Longword size

Notes: 1. POP.W Rn and PUSH.W Rn are identical to MOV.W @SP+, Rn and MOV.W Rn, @−SP.
POP.L ERn and PUSH.L ERn are identical to MOV.L @SP+, ERn and MOV.L ERn, @−SP.

2. Size of data to be added with a displacement
3. Size of data to specify a branch condition
4. Bcc is the generic designation of a conditional branch instruction.
5. Size of general register to be restored
6. Not available in this LSI.

2.7.1 Instructions and Addressing Modes

Table 2.2 indicates the combinations of instructions and addressing modes that the H8SX CPU can use.

Table 2.2 Combinations of Instructions and Addressing Modes (1)

			Addressing Mode								
Classification	**Instruction**	**Size**	**#xx**	**Rn**	**@ERn**	**@(d,ERn)**	**@(d, RnL.B/ Rn.W/ ERn.L)**	**@–ERn/ @ERn+/ @ERn–/ @+ERn**	**@aa:8**	**@aa:16/ @aa:32**	**—**
Data transfer	MOV	B/W/L	S	SD	SD	SD	SD	SD		SD	
		B		S/D					S/D		
	MOVFPE, MOVTPE*12	B		S/D						S/D*1	
	POP, PUSH	W/L		S/D				S/D*2			
	LDM, STM	L		S/D				S/D*2			
	MOVA*4	B/W		S	S	S	S	S		S	
Block transfer	EEPMOV	B									SD*3
	MOVMD	B/W/L									SD*3
	MOVSD	B									SD*3
Arithmetic operations	ADD, CMP	B	S	D	D	D	D	D	D	D	
		B		S	D	D	D	D	D	D	
		B		D	S	S	S	S	S	S	
		B			SD	SD	SD	SD		SD	
		W/L	S	SD	SD	SD	SD	SD		SD	
	SUB	B	S		D	D	D	D	D	D	
		B		S	D	D	D	D	D	D	
		B		D	S	S	S	S	S	S	
		B			SD	SD	SD	SD		SD	
		W/L	S	SD	SD	SD	SD	SD		SD	
	ADDX, SUBX	B/W/L	S	SD							
		B/W/L	S		SD						
		B/W/L	S					SD*5			
	INC, DEC	B/W/L		D							
	ADDS, SUBS	L		D							
	DAA, DAS	B		D							
	MULXU, DIVXU	B/W	S:4	SD							
	MULU, DIVU	W/L	S:4	SD							

			Addressing Mode								
Classification	**Instruction**	**Size**	**#xx**	**Rn**	**@ERn**	**@(d,ERn)**	**@(d, RnL.B/ Rn.W/ ERn.L)**	**@–ERn/ @ERn+/ @ERn–/ @+ERn**	**@aa:8**	**@aa:16/ @aa:32**	**—**
Arithmetic operations	MULXS, DIVXS	B/W	S:4	SD							
	MULS, DIVS	W/L	S:4	SD							
	NEG	B		D	D	D	D	D	D	D	
		W/L		D	D	D	D	D		D	
	EXTU, EXTS	W/L		D	D	D	D	D		D	
	TAS	B			D						
	MAC	—									
	CLRMAC	—									O
	LDMAC	—		S							
	STMAC	—		D							
Logic operations	AND, OR, XOR	B		S	D	D	D	D	D	D	
		B		D	S	S	S	S	S	S	
		B			SD	SD	SD	SD		SD	
		W/L	S	SD	SD	SD	SD	SD		SD	
	NOT	B		D	D	D	D	D	D	D	
		W/L		D	D	D	D	D		D	
Shift	SHLL, SHLR	B		D	D	D	D	D	D	D	
		W/L*[6]		D	D	D	D	D		D	
		B/W/L*[7]		D							
	SHAL, SHAR, ROTL, ROTR, ROTXL, ROTXR	B		D	D	D	D	D	D	D	
		W/L		D	D	D	D	D		D	
Bit manipulation	BSET, BCLR, BNOT, BTST, BSET/cc, BCLR/cc	B		D	D				D	D	
	BAND, BIAND, BOR, BIOR, BXOR, BIXOR, BLD, BILD, BST, BIST, BSTZ, BISTZ	B		D	D				D	D	

			Addressing Mode								
Classification	Instruction	Size	#xx	Rn	@ERn	@(d,ERn)	@(d, RnL.B/ Rn.W/ ERn.L)	@–ERn/ @ERn+/ @ERn–/ @+ERn	@aa:8	@aa:16/ @aa:32	—
Bit manipulation	BFLD	B		D	S				S	S	
	BFST	B		S	D				D	D	
Branch	BRA/BS, BRA/BC*[8]	B			S				S	S	
	BSR/BS, BSR/BC*[8]	B			S				S	S	
System control	LDC (CCR, EXR)	B/W*[9]	S	S	S	S		S*[10]		S	
	LDC (VBR, SBR)	L		S							
	STC (CCR, EXR)	B/W*[9]		D	D	D		D*[11]		D	
	STC (VBR, SBR)	L		D							
	ANDC, ORC, XORC	B	S								
	SLEEP	—									O
	NOP	—									O

[Legend]

d: d:16 or d:32

S: Can be specified as a source operand.

D: Can be specified as a destination operand.

SD: Can be specified as either a source or destination operand or both.

S/D: Can be specified as either a source or destination operand.

S:4: 4-bit immediate data can be specified as a source operand.

Notes:
1. Only @aa:16 is available.
2. @ERn+ as a source operand and @–ERn as a destination operand
3. Specified by ER5 as a source address and ER6 as a destination address for data transfer.
4. Size of data to be added with a displacement
5. Only @ERn– is available
6. When the number of bits to be shifted is 1, 2, 4, 8, or 16
7. When the number of bits to be shifted is specified by 5-bit immediate data or a general register
8. Size of data to specify a branch condition
9. Byte when immediate or register direct, otherwise, word
10. Only @ERn+ is available
11. Only @–ERn is available
12. Not available in this LSI.

Table 2.2 Combinations of Instructions and Addressing Modes (2)

			Addressing Mode							
Classification	**Instruction**	**Size**	**@ERn**	**@(d,PC)**	**@(RnL.B/ Rn.W/ ERn.L, PC)**	**@aa:24**	**@aa:32**	**@@aa:8**	**@@vec:7**	**—**
Branch	BRA/BS, BRA/BC	—		O						
	BSR/BS, BSR/BC	—		O						
	Bcc	—		O						
	BRA	—		O	O					
	BRA/S	—		O*						
	JMP	—	O			O	O	O	O	
	BSR	—		O						
	JSR	—	O			O	O	O	O	
	RTS, RTS/L	—								O
System control	TRAPA	—								O
	RTE, RTE/L	—								O

[Legend]

d: d:8 or d:16

Note: * Only @(d:8, PC) is available.

2.7.2 Table of Instructions Classified by Function

Tables 2.4 to 2.11 summarize the instructions in each functional category. The notation used in these tables is defined in table 2.3.

Table 2.3 Operation Notation

Operation Notation	Description
Rd	General register (destination)*
Rs	General register (source)*
Rn	General register*
ERn	General register (32-bit register)
(EAd)	Destination operand
(EAs)	Source operand
EXR	Extended control register
CCR	Condition-code register
VBR	Vector base register
SBR	Short address base register
N	N (negative) flag in CCR
Z	Z (zero) flag in CCR
V	V (overflow) flag in CCR
C	C (carry) flag in CCR
PC	Program counter
SP	Stack pointer
#IMM	Immediate data
disp	Displacement
+	Addition
−	Subtraction
×	Multiplication
÷	Division
∧	Logical AND
∨	Logical OR
⊕	Logical exclusive OR
→	Move
~	Logical not (logical complement)
:8/:16/:24/:32	8-, 16-, 24-, or 32-bit length

Note: * General registers include 8-bit registers (R0H to R7H, R0L to R7L), 16-bit registers (R0 to R7, E0 to E7), and 32-bit registers (ER0 to ER7).

Table 2.4 Data Transfer Instructions

Instruction	Size	Function
MOV	B/W/L	#IMM → (EAd), (EAs) → (EAd) Transfers data between immediate data, general registers, and memory.
MOVFPE*	B	(EAs) → Rd
MOVTPE*	B	Rs → (EAs)
POP	W/L	@SP+ → Rn Restores the data from the stack to a general register.
PUSH	W/L	Rn → @−SP Saves general register contents on the stack.
LDM	L	@SP+ → Rn (register list) Restores the data from the stack to multiple general registers. Two, three, or four general registers which have serial register numbers can be specified.
STM	L	Rn (register list) → @−SP Saves the contents of multiple general registers on the stack. Two, three, or four general registers which have serial register numbers can be specified.
MOVA	B/W	EA → Rd Zero-extends and shifts the contents of a specified general register or memory data and adds them with a displacement. The result is stored in a general register.

Note: * Not available in this LSI.

Table 2.5 Block Transfer Instructions

Instruction	Size	Function
EEPMOV.B EEPMOV.W	B	Transfers a data block. Transfers byte data which begins at a memory location specified by ER5 to a memory location specified by ER6. The number of byte data to be transferred is specified by R4 or R4L.
MOVMD.B	B	Transfers a data block. Transfers byte data which begins at a memory location specified by ER5 to a memory location specified by ER6. The number of byte data to be transferred is specified by R4.
MOVMD.W	W	Transfers a data block. Transfers word data which begins at a memory location specified by ER5 to a memory location specified by ER6. The number of word data to be transferred is specified by R4.
MOVMD.L	L	Transfers a data block. Transfers longword data which begins at a memory location specified by ER5 to a memory location specified by ER6. The number of longword data to be transferred is specified by R4.
MOVSD.B	B	Transfers a data block with zero data detection. Transfers byte data which begins at a memory location specified by ER5 to a memory location specified by ER6. The number of byte data to be transferred is specified by R4. When zero data is detected during transfer, the transfer stops and execution branches to a specified address.

Table 2.6 Arithmetic Operation Instructions

Instruction	Size	Function
ADD	B/W/L	(EAd) ± #IMM → (EAd), (EAd) ± (EAs) → (EAd)
SUB		Performs addition or subtraction on data between immediate data, general registers, and memory. Immediate byte data cannot be subtracted from byte data in a general register.
ADDX	B/W/L	(EAd) ± #IMM ± C → (EAd), (EAd) ± (EAs) ± C → (EAd)
SUBX		Performs addition or subtraction with carry on data between immediate data, general registers, and memory. The addressing mode which specifies a memory location can be specified as register indirect with post-decrement or register indirect.
INC	B/W/L	Rd ± 1 → Rd, Rd ± 2 → Rd
DEC		Increments or decrements a general register by 1 or 2. (Byte operands can be incremented or decremented by 1 only.)
ADDS	L	Rd ± 1 → Rd, Rd ± 2 → Rd, Rd ± 4 → Rd
SUBS		Adds or subtracts the value 1, 2, or 4 to or from data in a general register.
DAA	B	Rd (decimal adjust) → Rd
DAS		Decimal-adjusts an addition or subtraction result in a general register by referring to the CCR to produce 2-digit 4-bit BCD data.
MULXU	B/W	Rd × Rs → Rd
		Performs unsigned multiplication on data in two general registers: either 8 bits × 8 bits → 16 bits, or 16 bits × 16 bits → 32 bits.
MULU	W/L	Rd × Rs → Rd
		Performs unsigned multiplication on data in two general registers: either 16 bits × 16 bits → 16 bits, or 32 bits × 32 bits → 32 bits.
MULU/U	L	Rd × Rs → Rd
		Performs unsigned multiplication on data in two general registers (32 bits × 32 bits → upper 32 bits).
MULXS	B/W	Rd × Rs → Rd
		Performs signed multiplication on data in two general registers: either 8 bits × 8 bits → 16 bits, or 16 bits × 16 bits → 32 bits.
MULS	W/L	Rd × Rs → Rd
		Performs signed multiplication on data in two general registers: either 16 bits × 16 bits → 16 bits, or 32 bits × 32 bits → 32 bits.
MULS/U	L	Rd × Rs → Rd
		Performs signed multiplication on data in two general registers (32 bits × 32 bits → upper 32 bits).
DIVXU	B/W	Rd ÷ Rs → Rd
		Performs unsigned division on data in two general registers: either 16 bits ÷ 8 bits → 8-bit quotient and 8-bit remainder, or 32 bits ÷ 16 bits → 16-bit quotient and 16-bit remainder.

Instruction	Size	Function
DIVU	W/L	Rd ÷ Rs → Rd Performs unsigned division on data in two general registers: either 16 bits ÷ 16 bits → 16-bit quotient, or 32 bits ÷ 32 bits → 32-bit quotient.
DIVXS	B/W	Rd ÷ Rs → Rd Performs signed division on data in two general registers: either 16 bits ÷ 8 bits → 8-bit quotient and 8-bit remainder, or 32 bits ÷ 16 bits → 16-bit quotient and 16-bit remainder.
DIVS	W/L	Rd ÷ Rs → Rd Performs signed division on data in two general registers: either 16 bits ÷ 16 bits → 16-bit quotient, or 32 bits ÷ 32 bits → 32-bit quotient.
CMP	B/W/L	(EAd) – #IMM, (EAd) – (EAs) Compares data between immediate data, general registers, and memory and stores the result in CCR.
NEG	B/W/L	0 – (EAd) → (EAd) Takes the two's complement (arithmetic complement) of data in a general register or the contents of a memory location.
EXTU	W/L	(EAd) (zero extension) → (EAd) Performs zero-extension on the lower 8 or 16 bits of data in a general register or memory to word or longword size. The lower 8 bits to word or longword, or the lower 16 bits to longword can be zero-extended.
EXTS	W/L	(EAd) (sign extension) → (EAd) Performs sign-extension on the lower 8 or 16 bits of data in a general register or memory to word or longword size. The lower 8 bits to word or longword, or the lower 16 bits to longword can be sign-extended.
TAS	B	@ERd – 0, 1 → (<bit 7> of @EAd) Tests memory contents, and sets the most significant bit (bit 7) to 1.
MAC	—	(EAs) × (EAd) + MAC → MAC Performs signed multiplication on memory contents and adds the result to MAC.
CLRMAC	—	0 → MAC Clears MAC to zero.
LDMAC	—	Rs → MAC Loads data from a general register to MAC.
STMAC	—	MAC → Rd Stores data from MAC to a general register.

Table 2.7 Logic Operation Instructions

Instruction	Size	Function
AND	B/W/L	(EAd) ∧ #IMM → (EAd), (EAd) ∧ (EAs) → (EAd) Performs a logical AND operation on data between immediate data, general registers, and memory.
OR	B/W/L	(EAd) ∨ #IMM → (EAd), (EAd) ∨ (EAs) → (EAd) Performs a logical OR operation on data between immediate data, general registers, and memory.
XOR	B/W/L	(EAd) ⊕ #IMM → (EAd), (EAd) ⊕ (EAs) → (EAd) Performs a logical exclusive OR operation on data between immediate data, general registers, and memory.
NOT	B/W/L	~ (EAd) → (EAd) Takes the one's complement of the contents of a general register or a memory location.

Table 2.8 Shift Operation Instructions

Instruction	Size	Function
SHLL	B/W/L	(EAd) (shift) → (EAd)
SHLR		Performs a logical shift on the contents of a general register or a memory location. The contents of a general register or a memory location can be shifted by 1, 2, 4, 8, or 16 bits. The contents of a general register can be shifted by any bits. In this case, the number of bits is specified by 5-bit immediate data or the lower 5 bits of the contents of a general register.
SHAL	B/W/L	(EAd) (shift) → (EAd)
SHAR		Performs an arithmetic shift on the contents of a general register or a memory location. 1-bit or 2-bit shift is possible.
ROTL	B/W/L	(EAd) (rotate) → (EAd)
ROTR		Rotates the contents of a general register or a memory location. 1-bit or 2-bit rotation is possible.
ROTXL	B/W/L	(EAd) (rotate) → (EAd)
ROTXR		Rotates the contents of a general register or a memory location with the carry bit. 1-bit or 2-bit rotation is possible.

Table 2.9 Bit Manipulation Instructions

Instruction	Size	Function
BSET	B	1 → (<bit-No.> of <EAd>) Sets a specified bit in the contents of a general register or a memory location to 1. The bit number is specified by 3-bit immediate data or the lower three bits of a general register.
BSET/cc	B	if cc, 1 → (<bit-No.> of <EAd>) If the specified condition is satisfied, this instruction sets a specified bit in a memory location to 1. The bit number can be specified by 3-bit immediate data, or by the lower three bits of a general register. The Z flag status can be specified as a condition.
BCLR	B	0 → (<bit-No.> of <EAd>) Clears a specified bit in the contents of a general register or a memory location to 0. The bit number is specified by 3-bit immediate data or the lower three bits of a general register.
BCLR/cc	B	if cc, 0 → (<bit-No.> of <EAd>) If the specified condition is satisfied, this instruction clears a specified bit in a memory location to 0. The bit number can be specified by 3-bit immediate data, or by the lower three bits of a general register. The Z flag status can be specified as a condition.
BNOT	B	~ (<bit-No.> of <EAd>) → (<bit-No.> of <EAd>) Inverts a specified bit in the contents of a general register or a memory location. The bit number is specified by 3-bit immediate data or the lower three bits of a general register.
BTST	B	~ (<bit-No.> of <EAd>) → Z Tests a specified bit in the contents of a general register or a memory location and sets or clears the Z flag accordingly. The bit number is specified by 3-bit immediate data or the lower three bits of a general register.
BAND	B	C ∧ (<bit-No.> of <EAd>) → C ANDs the carry flag with a specified bit in the contents of a general register or a memory location and stores the result in the carry flag. The bit number is specified by 3-bit immediate data.
BIAND	B	C ∧ [~ (<bit-No.> of <EAd>)] → C ANDs the carry flag with the inverse of a specified bit in the contents of a general register or a memory location and stores the result in the carry flag. The bit number is specified by 3-bit immediate data.
BOR	B	C ∨ (<bit-No.> of <EAd>) → C ORs the carry flag with a specified bit in the contents of a general register or a memory location and stores the result in the carry flag. The bit number is specified by 3-bit immediate data.

Instruction	Size	Function
BIOR	B	C ∨ [~ (<bit-No.> of <EAd>)] → C ORs the carry flag with the inverse of a specified bit in the contents of a general register or a memory location and stores the result in the carry flag. The bit number is specified by 3-bit immediate data.
BXOR	B	C ⊕ (<bit-No.> of <EAd>) → C Exclusive-ORs the carry flag with a specified bit in the contents of a general register or a memory location and stores the result in the carry flag. The bit number is specified by 3-bit immediate data.
BIXOR	B	C ⊕ [~ (<bit-No.> of <EAd>)] → C Exclusive-ORs the carry flag with the inverse of a specified bit in the contents of a general register or a memory location and stores the result in the carry flag. The bit number is specified by 3-bit immediate data.
BLD	B	(<bit-No.> of <EAd>) → C Transfers a specified bit in the contents of a general register or a memory location to the carry flag. The bit number is specified by 3-bit immediate data.
BILD	B	~ (<bit-No.> of <EAd>) → C Transfers the inverse of a specified bit in the contents of a general register or a memory location to the carry flag. The bit number is specified by 3-bit immediate data.
BST	B	C → (<bit-No.> of <EAd>) Transfers the carry flag value to a specified bit in the contents of a general register or a memory location. The bit number is specified by 3-bit immediate data.
BSTZ	B	Z → (<bit-No.> of <EAd>) Transfers the zero flag value to a specified bit in the contents of a memory location. The bit number is specified by 3-bit immediate data.
BIST	B	~ C → (<bit-No.> of <EAd>) Transfers the inverse of the carry flag value to a specified bit in the contents of a general register or a memory location. The bit number is specified by 3-bit immediate data.

Instruction	Size	Function
BISTZ	B	~ Z → (<bit-No.> of <EAd>) Transfers the inverse of the zero flag value to a specified bit in the contents of a memory location. The bit number is specified by 3-bit immediate data.
BFLD	B	(EAs) (bit field) → Rd Transfers a specified bit field in memory location contents to the lower bits of a specified general register.
BFST	B	Rs → (EAd) (bit field) Transfers the lower bits of a specified general register to a specified bit field in memory location contents.

Table 2.10 Branch Instructions

Instruction	Size	Function
BRA/BS BRA/BC	B	Tests a specified bit in memory location contents. If the specified condition is satisfied, execution branches to a specified address.
BSR/BS BSR/BC	B	Tests a specified bit in memory location contents. If the specified condition is satisfied, execution branches to a subroutine at a specified address.
Bcc	—	Branches to a specified address if the specified condition is satisfied.
BRA/S	—	Branches unconditionally to a specified address after executing the next instruction. The next instruction should be a 1-word instruction except for the block transfer and branch instructions.
JMP	—	Branches unconditionally to a specified address.
BSR	—	Branches to a subroutine at a specified address.
JSR	—	Branches to a subroutine at a specified address.
RTS	—	Returns from a subroutine.
RTS/L	—	Returns from a subroutine, restoring data from the stack to multiple general registers.

Table 2.11 System Control Instructions

Instruction	Size	Function
TRAPA	—	Starts trap-instruction exception handling.
RTE	—	Returns from an exception-handling routine.
RTE/L	—	Returns from an exception-handling routine, restoring data from the stack to multiple general registers.
SLEEP	—	Causes a transition to a power-down state.
LDC	B/W	#IMM → CCR, (EAs) → CCR, #IMM → EXR, (EAs) → EXR Loads immediate data or the contents of a general register or a memory location to CCR or EXR. Although CCR and EXR are 8-bit registers, word-size transfers are performed between them and memory. The upper 8 bits are valid.
	L	Rs → VBR, Rs → SBR Transfers the general register contents to VBR or SBR.
STC	B/W	CCR → (EAd), EXR → (EAd) Transfers the contents of CCR or EXR to a general register or memory. Although CCR and EXR are 8-bit registers, word-size transfers are performed between them and memory. The upper 8 bits are valid.
	L	VBR → Rd, SBR → Rd Transfers the contents of VBR or SBR to a general register.
ANDC	B	CCR ∧ #IMM → CCR, EXR ∧ #IMM → EXR Logically ANDs the CCR or EXR contents with immediate data.
ORC	B	CCR ∨ #IMM → CCR, EXR ∨ #IMM → EXR Logically ORs the CCR or EXR contents with immediate data.
XORC	B	CCR ⊕ #IMM → CCR, EXR ⊕ #IMM → EXR Logically exclusive-ORs the CCR or EXR contents with immediate data.
NOP	—	PC + 2 → PC Only increments the program counter.

2.7.3 Basic Instruction Formats

The H8SX CPU instructions consist of 2-byte (1-word) units. An instruction consists of an operation field (op field), a register field (r field), an effective address extension (EA field), and a condition field (cc).

Figure 2.14 shows examples of instruction formats.

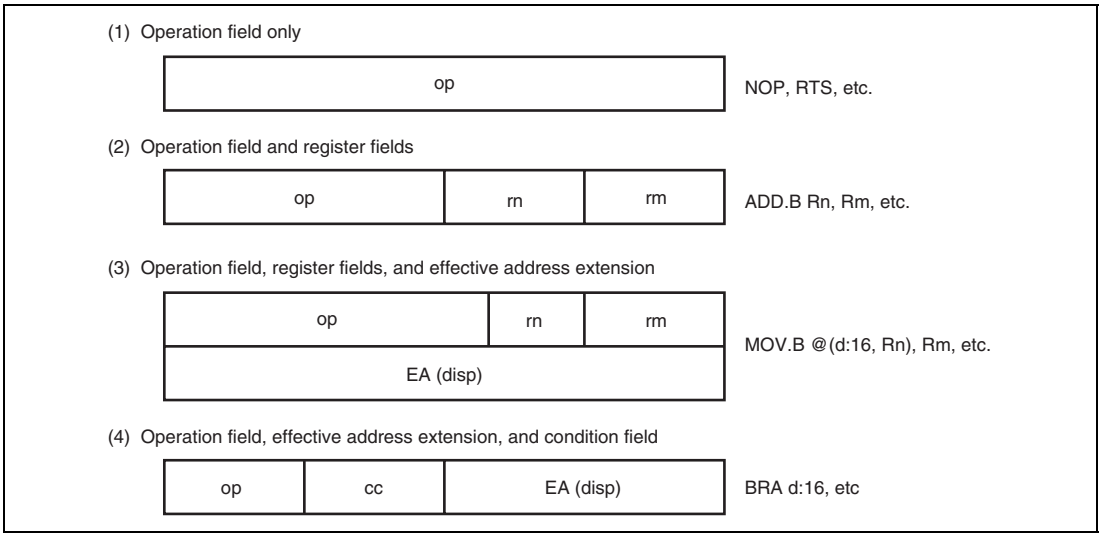


Figure 2.14 Instruction Formats

- Operation Field

  Indicates the function of the instruction, and specifies the addressing mode and operation to be carried out on the operand. The operation field always includes the first four bits of the instruction. Some instructions have two operation fields.
- Register Field

  Specifies a general register. Address registers are specified by 3 bits, data registers by 3 bits or 4 bits. Some instructions have two register fields. Some have no register field.
- Effective Address Extension

  8, 16, or 32 bits specifying immediate data, an absolute address, or a displacement.
- Condition Field

  Specifies the branch condition of Bcc instructions.

2.8 Addressing Modes and Effective Address Calculation

The H8SX CPU supports the 11 addressing modes listed in table 2.12. Each instruction uses a subset of these addressing modes.

Bit manipulation instructions use register direct, register indirect, or absolute addressing mode to specify an operand, and register direct (BSET, BCLR, BNOT, and BTST instructions) or immediate (3-bit) addressing mode to specify a bit number in the operand.

Table 2.12 Addressing Modes

No.	Addressing Mode	Symbol
1	Register direct	Rn
2	Register indirect	@ERn
3	Register indirect with displacement	@(d:2,ERn)/@(d:16,ERn)/@(d:32,ERn)
4	Index register indirect with displacement	@(d:16, RnL.B)/@(d:16,Rn.W)/@(d:16,ERn.L) @(d:32, RnL.B)/@(d:32,Rn.W)/@(d:32,ERn.L)
5	Register indirect with post-increment	@ERn+
	Register indirect with pre-decrement	@–ERn
	Register indirect with pre-increment	@+ERn
	Register indirect with post-decrement	@ERn–
6	Absolute address	@aa:8/@aa:16/@aa:24/@aa:32
7	Immediate	#xx:3/#xx:4/#xx:8/#xx:16/#xx:32
8	Program-counter relative	@(d:8,PC)/@(d:16,PC)
9	Program-counter relative with index register	@(RnL.B,PC)/@(Rn.W,PC)/@(ERn.L,PC)
10	Memory indirect	@@aa:8
11	Extended memory indirect	@@vec:7

2.8.1 Register Direct—Rn

The operand value is the contents of an 8-, 16-, or 32-bit general register which is specified by the register field in the instruction code.

R0H to R7H and R0L to R7L can be specified as 8-bit registers.

R0 to R7 and E0 to E7 can be specified as 16-bit registers.

ER0 to ER7 can be specified as 32-bit registers.

2.8.2 Register Indirect—@ERn

The operand value is the contents of the memory location which is pointed to by the contents of an address register (ERn). ERn is specified by the register field of the instruction code.

In advanced mode, if this addressing mode is used in a branch instruction, the lower 24 bits are valid and the upper 8 bits are all assumed to be 0 (H'00).

2.8.3 Register Indirect with Displacement —@(d:2, ERn), @(d:16, ERn), or @(d:32, ERn)

The operand value is the contents of a memory location which is pointed to by the sum of the contents of an address register (ERn) and a 16- or 32-bit displacement. ERn is specified by the register field of the instruction code. The displacement is included in the instruction code and the 16-bit displacement is sign-extended when added to ERn.

This addressing mode has a short format (@(d:2, ERn)). The short format can be used when the displacement is 1, 2, or 3 and the operand is byte data, when the displacement is 2, 4, or 6 and the operand is word data, or when the displacement is 4, 8, or 12 and the operand is longword data.

2.8.4 Index Register Indirect with Displacement—@(d:16,RnL.B), @(d:32,RnL.B), @(d:16,Rn.W), @(d:32,Rn.W), @(d:16,ERn.L), or @(d:32,ERn.L)

The operand value is the contents of a memory location which is pointed to by the sum of the following operation result and a 16- or 32-bit displacement: a specified bits of the contents of an address register (RnL, Rn, ERn) specified by the register field in the instruction code are zero-extended to 32-bit data and multiplied by 1, 2, or 4. The displacement is included in the instruction code and the 16-bit displacement is sign-extended when added to ERn. If the operand is byte data, ERn is multiplied by 1. If the operand is word or longword data, ERn is multiplied by 2 or 4, respectively.

2.8.5 Register Indirect with Post-Increment, Pre-Decrement, Pre-Increment, or Post-Decrement—@ERn+, @–ERn, @+ERn, or @ERn–

- Register indirect with post-increment—@ERn+

  The operand value is the contents of a memory location which is pointed to by the contents of an address register (ERn). ERn is specified by the register field of the instruction code. After the memory location is accessed, 1, 2, or 4 is added to the address register contents and the sum is stored in the address register. The value added is 1 for byte access, 2 for word access, or 4 for longword access.
- Register indirect with pre-decrement—@–ERn

  The operand value is the contents of a memory location which is pointed to by the following operation result: the value 1, 2, or 4 is subtracted from the contents of an address register (ERn). ERn is specified by the register field of the instruction code. After that, the operand value is stored in the address register. The value subtracted is 1 for byte access, 2 for word access, or 4 for longword access.
- Register indirect with pre-increment—@+ERn

  The operand value is the contents of a memory location which is pointed to by the following operation result: the value 1, 2, or 4 is added to the contents of an address register (ERn). ERn is specified by the register field of the instruction code. After that, the operand value is stored in the address register. The value added is 1 for byte access, 2 for word access, or 4 for longword access.
- Register indirect with post-decrement—@ERn–

  The operand value is the contents of a memory location which is pointed to by the contents of an address register (ERn). ERn is specified by the register field of the instruction code. After the memory location is accessed, 1, 2, or 4 is subtracted from the address register contents and the remainder is stored in the address register. The value subtracted is 1 for byte access, 2 for word access, or 4 for longword access.

Using this addressing mode, data to be written is the contents of the general register after calculating an effective address. If the same general register is specified in an instruction and two effective addresses are calculated, the contents of the general register after the first calculation of an effective address is used in the second calculation of an effective address.

Example 1:

MOV.W R0, @ER0+
When ER0 before execution is H'12345678, H'567A is written at H'12345678.

Example 2:

MOV.B @ER0+, @ER0+
When ER0 before execution is H'00001000, H'00001000 is read and the contents is written at H'00001001.
After execution, ER0 is H'00001002.

2.8.6 Absolute Address—@aa:8, @aa:16, @aa:24, or @aa:32

The operand value is the contents of a memory location which is pointed to by an absolute address included in the instruction code.

There are 8-bit (@aa:8), 16-bit (@aa:16), 24-bit (@aa:24), and 32-bit (@aa:32) absolute addresses.

To access the data area, the absolute address of 8 bits (@aa:8), 16 bits (@aa:16), or 32 bits (@aa:32) is used. For an 8-bit absolute address, the upper 24 bits are specified by SBR. For a 16-bit absolute address, the upper 16 bits are sign-extended. A 32-bit absolute address can access the entire address space.

To access the program area, the absolute address of 24 bits (@aa:24) or 32 bits (@aa:32) is used. For a 24-bit absolute address, the upper 8 bits are all assumed to be 0 (H'00).

Table 2.13 shows the accessible absolute address ranges.

Table 2.13 Absolute Address Access Ranges

Absolute Address		Normal Mode	Middle Mode	Advanced Mode	Maximum Mode
Data area	8 bits (@aa:8)	A consecutive 256-byte area (the upper address is set in SBR)			
	16 bits (@aa:16)	H'0000 to H'FFFF	H'000000 to H'007FFF, H'FF8000 to H'FFFFFF	H'00000000 to H'00007FFF, H'FFFF8000 to H'FFFFFFFF	
	32 bits (@aa:32)			H'00000000 to H'FFFFFFFF	
Program area	24 bits (@aa:24)		H'000000 to H'FFFFFF	H'00000000 to H'00FFFFFF	
	32 bits (@aa:32)			H'00000000 to H'00FFFFFF	H'00000000 to H'FFFFFFFF

2.8.7 Immediate—#xx

The operand value is 8-bit (#xx:8), 16-bit (#xx:16), or 32-bit (#xx:32) data included in the instruction code.

This addressing mode has short formats in which 3- or 4-bit immediate data can be used.

When the size of immediate data is less than that of the destination operand value (byte, word, or longword) the immediate data is zero-extended.

The ADDS, SUBS, INC, and DEC instructions contain immediate data implicitly. Some bit manipulation instructions contain 3-bit immediate data in the instruction code, for specifying a bit number. The BFLD and BFST instructions contain 8-bit immediate data in the instruction code, for specifying a bit field. The TRAPA instruction contains 2-bit immediate data in the instruction code, for specifying a vector address.

2.8.8 Program-Counter Relative—@(d:8, PC) or @(d:16, PC)

This mode is used in the Bcc and BSR instructions. The operand value is a 32-bit branch address, which is the sum of an 8- or 16-bit displacement in the instruction code and the 32-bit address of the PC contents. The 8-bit or 16-bit displacement is sign-extended to 32 bits when added to the PC contents. The PC contents to which the displacement is added is the address of the first byte of the next instruction, so the possible branching range is −126 to +128 bytes (−63 to +64 words) or −32766 to +32768 bytes (−16383 to +16384 words) from the branch instruction. The resulting value should be an even number. In advanced mode, only the lower 24 bits of this branch address are valid; the upper 8 bits are all assumed to be 0 (H'00).

2.8.9 Program-Counter Relative with Index Register—@(RnL.B, PC), @(Rn.W, PC), or @(ERn.L, PC)

This mode is used in the Bcc and BSR instructions. The operand value is a 32-bit branch address, which is the sum of the following operation result and the 32-bit address of the PC contents: the contents of an address register specified by the register field in the instruction code (RnL, Rn, or ERn) is zero-extended and multiplied by 2. The PC contents to which the displacement is added is the address of the first byte of the next instruction. In advanced mode, only the lower 24 bits of this branch address are valid; the upper 8 bits are all assumed to be 0 (H'00).

2.8.10 Memory Indirect—@@aa:8

This mode can be used by the JMP and JSR instructions. The operand value is a branch address, which is the contents of a memory location pointed to by an 8-bit absolute address in the instruction code.

The upper bits of an 8-bit absolute address are all assumed to be 0, so the address range is 0 to 255 (H'0000 to H'00FF in normal mode, H'000000 to H'0000FF in other modes).

In normal mode, the memory location is pointed to by word-size data and the branch address is 16 bits long. In other modes, the memory location is pointed to by longword-size data. In middle or advanced mode, the first byte of the longword-size data is assumed to be all 0 (H'00).

Note that the top part of the address range is also used as the exception handling vector area. A vector address of an exception handling other than a reset or a CPU address error can be changed by VBR.

Figure 2.15 shows an example of specification of a branch address using this addressing mode.

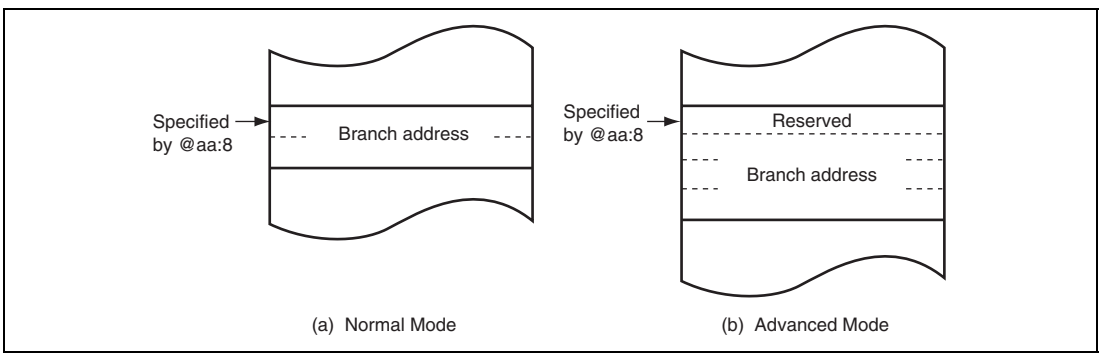


Figure 2.15 Branch Address Specification in Memory Indirect Mode

2.8.11 Extended Memory Indirect—@@vec:7

This mode can be used by the JMP and JSR instructions. The operand value is a branch address, which is the contents of a memory location pointed to by the following operation result: the sum of 7-bit data in the instruction code and the value of H'80 is multiplied by 2 or 4.

The address range to store a branch address is H'0100 to H'01FF in normal mode and H'000200 to H'0003FF in other modes. In assembler notation, an address to store a branch address is specified.

In normal mode, the memory location is pointed to by word-size data and the branch address is 16 bits long. In other modes, the memory location is pointed to by longword-size data. In middle or advanced mode, the first byte of the longword-size data is assumed to be all 0 (H'00).

2.8.12 Effective Address Calculation

Tables 2.14 and 2.15 show how effective addresses are calculated in each addressing mode. The lower bits of the effective address are valid and the upper bits are ignored (zero extended or sign extended) according to the CPU operating mode.

The valid bits in middle mode are as follows:

- The lower 16 bits of the effective address are valid and the upper 16 bits are sign-extended for the transfer and operation instructions.
- The lower 24 bits of the effective address are valid and the upper eight bits are zero-extended for the branch instructions.

Table 2.14 Effective Address Calculation for Transfer and Operation Instructions

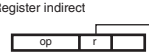
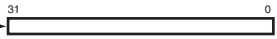
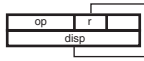
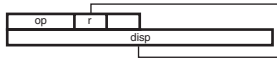
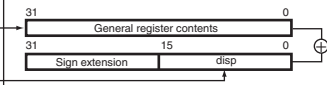
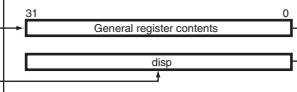
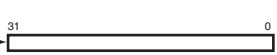
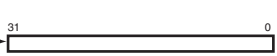
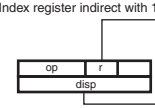
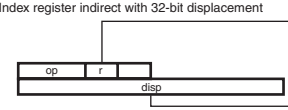
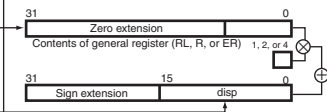
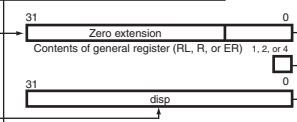
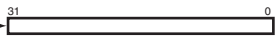
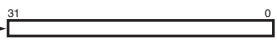
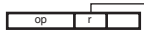
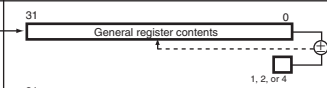
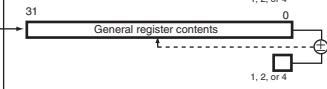
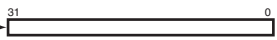
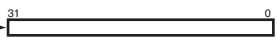
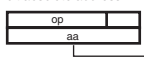
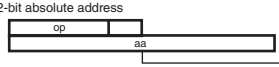
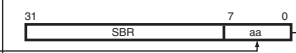
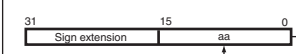
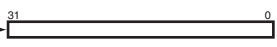
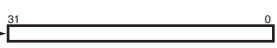
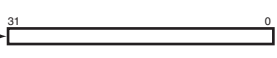
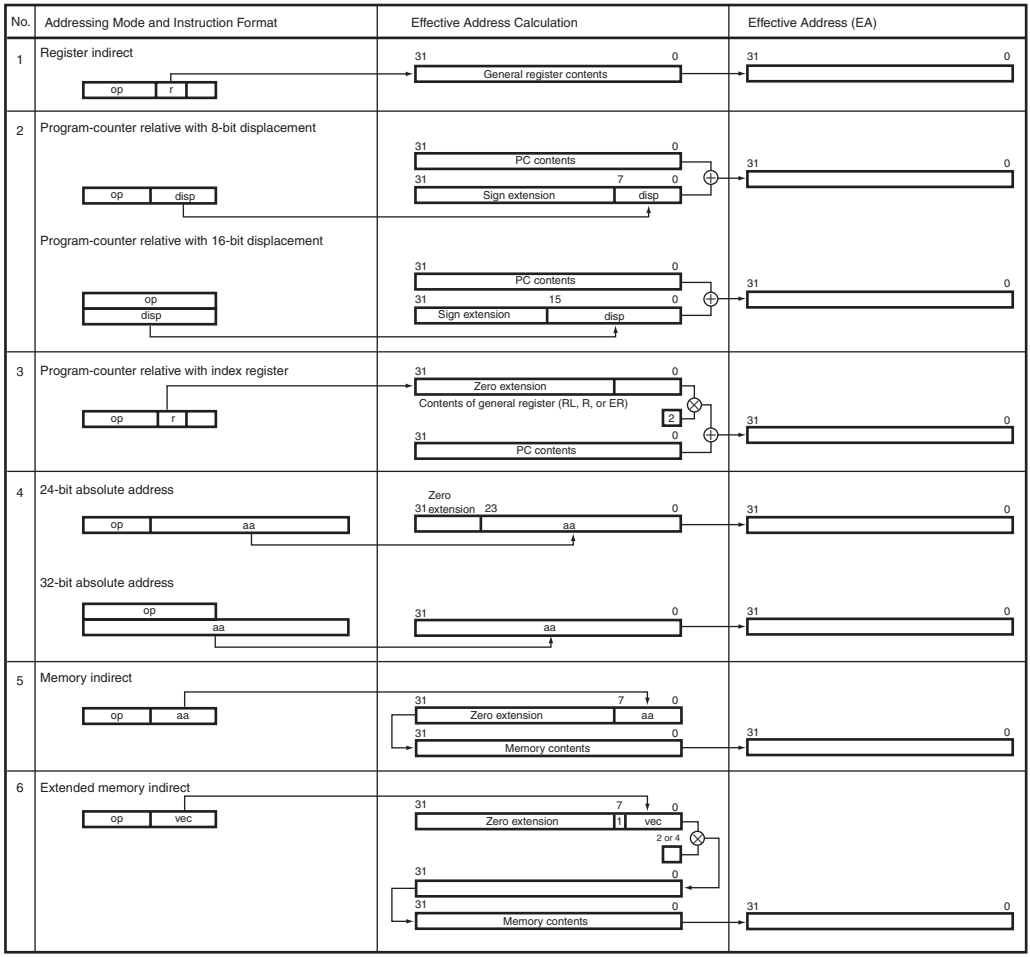
No.	Addressing Mode and Instruction Format	Effective Address Calculation	Effective Address (EA)
1	Immediate 		
2	Register direct 		
3	Register indirect 		
4	Register indirect with 16-bit displacement  Register indirect with 32-bit displacement 	 	 
5	Index register indirect with 16-bit displacement  Index register indirect with 32-bit displacement 	 	 
6	Register indirect with post-increment or post-decrement  Register indirect with pre-increment or pre-decrement 	 	 
7	8-bit absolute address  16-bit absolute address  32-bit absolute address 	  	  

Table 2.15 Effective Address Calculation for Branch Instructions



2.8.13 MOVA Instruction

The MOVA instruction stores the effective address in a general register.

1. Firstly, data is obtained by the addressing mode shown in item 2 of table 2.14.
2. Next, the effective address is calculated using the obtained data as the index by the addressing mode shown in item 5 of table 2.14. The obtained data is used instead of the general register. The result is stored in a general register. For details, see H8SX Family Software Manual.

2.9 Processing States

The H8SX CPU has five main processing states: the reset state, exception-handling state, program execution state, bus-released state, and program stop state. Figure 2.16 indicates the state transitions.

- Reset state

  In this state, the CPU and internal peripheral modules are all initialized and stopped. When the $\overline{\text{RES}}$ input goes low, all current processing stops and the CPU enters the reset state. All interrupts are masked in the reset state. Reset exception handling starts when the $\overline{\text{RES}}$ signal changes from low to high. For details, see section 4, Exception Handling.

  The reset state can also be entered by a watchdog timer overflow when available.
- Exception-handling state

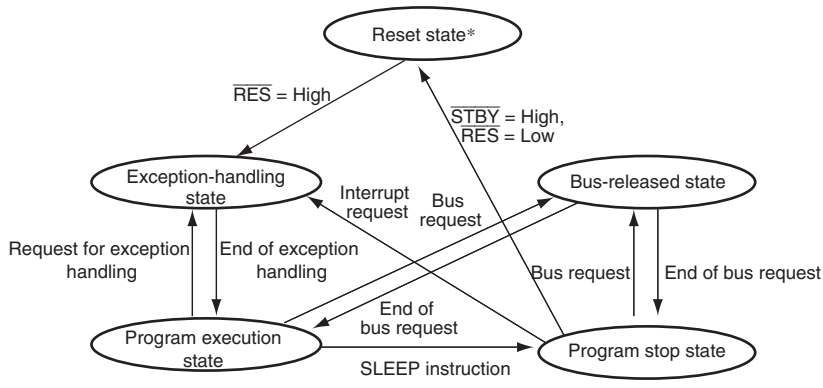
  The exception-handling state is a transient state that occurs when the CPU alters the normal processing flow due to activation of an exception source, such as, a reset, trace, interrupt, or trap instruction. The CPU fetches a start address (vector) from the exception handling vector table and branches to that address. For details, see section 4, Exception Handling.
- Program execution state

  In this state, the CPU executes program instructions in sequence.
- Bus-released state

  The bus-released state occurs when the bus has been released in response to a bus request from a bus master other than the CPU. While the bus is released, the CPU halts operations.
- Program stop state

  This is a power-down state in which the CPU stops operating. The program stop state occurs when a SLEEP instruction is executed or the CPU enters hardware standby mode*. For details, see section 23, Power-Down Modes.

Note: * This LSI does not support hardware standby mode.



Notes: In any state, when the STBY signal goes low, the hardware standby mode is entered.

* From any state except hardware standby mode, a transition to the reset state occurs whenever the RES signal goes low. A transition can also be made to the reset state when the watchdog timer overflows is canceled.

Figure 2.16 State Transitions

Section 3 MCU Operating Modes

3.1 Operating Mode Selection

This LSI has three operating modes (modes 1 to 3). The operating mode is selected by the setting of mode pins (MD1 and MD0). Table 3.1 lists MCU operating mode settings. Do not make settings other than those listed in this table.

In this LSI, advanced mode for the CPU operating mode and 16-Mbyte address space are available. LSI initiation mode can be selected from boot mode for programming/erasing the flash memory and single chip mode.

Table 3.1 MCU Operating Mode Settings

MCU Operating Mode	MD1	MD0	CPU Operating Mode	Address Space	Description	On-Chip ROM
1	0	1	—	—	Reserved	—
2	1	0	Advanced	16 Mbytes	Boot mode	Enabled
3	1	1			Single-chip mode	Enabled

Mode 1 is reserved. This LSI does not have mode 1. Do not set this mode.

Mode 2 is a boot mode in which flash memory can be programmed or erased. For details on boot mode, see section 20, Flash Memory.

In mode 3, this LSI operates in single-chip mode.

3.2 Register Descriptions

The following registers are related to the operating mode setting.

- Mode control register (MDCR)
- System control register (SYSCR0)
- System control register (SYSCR1)

3.2.1 Mode Control Register (MDCR)

MDCR indicates the current operating mode. When MDCR is read, the states of signals input on pins MD1 and MD0 are latched. The latch is released by a reset.

Bit	15	14	13	12	11	10	9	8
Bit Name	—	—	—	—	MDS3	MDS2	MDS1	MDS0
Initial Value	0	1	0	1	Undefined*	Undefined*	Undefined*	Undefined*
R/W	R	R	R	R	R	R	R	R

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	—
Initial Value	0	1	0	1	Undefined*	Undefined*	Undefined*	Undefined*
R/W	R	R	R	R	R	R	R	R

Note: * Determined by the states on pins MD1 and MD0.

Bit	Bit Name	Initial Value	R/W	Descriptions
15	—	0	R	Reserved
14	—	1	R	These are read-only bits and cannot be modified.
13	—	0	R	
12	—	1	R	
11	MDS3	Undefined*	R	Mode Select 3 to 0
10	MDS2	Undefined*	R	These bits indicate the operating mode selected by the mode pins (MD1 and MD0) (see table 3.2).
9	MDS1	Undefined*	R	
8	MDS0	Undefined*	R	

Bit	Bit Name	Initial Value	R/W	Descriptions
7	—	0	R	Reserved
6	—	1	R	These are read-only bits and cannot be modified.
5	—	0	R	
4	—	1	R	
3	—	Undefined*	R	
2	—	Undefined*	R	
1	—	Undefined*	R	
0	—	Undefined*	R	

Note: * Determined by the states on pins MD1 and MD0.

Table 3.2 Settings of Bits MDS3 to MDS0

			MDCR			
MCU Operating Mode	**MD1**	**MD0**	**MDS3**	**MDS2**	**MDS1**	**MDS0**
1	0	1	Reserved	Reserved	Reserved	Reserved
2	1	0	0	0	1	0
3	1	1	0	0	1	1

3.2.2 System Control Register (SYSCR0)

SYSCR0 controls MAC saturation operation and enables/disables the on-chip RAM.

Bit	15	14	13	12	11	10	9	8
Bit Name	—	—	MACS	—	—	—	—	RAME
Initial Value	1	1	0	1	0	1	0	1
R/W	R	R	R/W	R	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	—
Initial Value	0	0	0	0	0	0	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Descriptions
15, 14	—	All 1	R	Reserved These are read-only bits and cannot be modified.
13	MACS	0	R/W	MAC Saturation Operation Control Selects either saturation operation or non-saturation operation for the MAC instruction. 0: MAC instruction is non-saturation operation 1: MAC instruction is saturation operation
12	—	1	R	Reserved This is a read-only bit and cannot be modified.
11	—	0	R/W	Reserved This bit is always read as 0. The write value should always be 0.
10	—	1	R/W	Reserved This bit is always read as 1. The write value should always be 1.
9	—	0	R/W	Reserved This bit is always read as 0. The write value should always be 0.
8	RAME	1	R/W	RAM Enable Enables or disables the on-chip RAM. This bit is initialized when the reset state is released. Do not write 0 during access to the on-chip RAM. 0: On-chip RAM disabled 1: On-chip RAM enabled
7 to 2	—	All 0	R/W	Reserved These bits are always read as 0. The write value should always be 0.
1, 0	—	All 1	R/W	Reserved This bit is always read as 1. The write value should always be 1.

3.2.3 System Control Register (SYSCR1)

SYSCR1 controls the access state of data flash.

Bit	15	14	13	12	11	10	9	8
Bit Name	—	—	—	—	—	—	—	—
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	EEPWT
Initial Value	0	0	0	0	0	0	0	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Descriptions
15	—	0	R/W	Reserved
14	—	0	R/W	These bits are always read as 0. The write value should always be 0.
13	—	0	R/W	
12	—	0	R/W	
11	—	0	R/W	
10	—	0	R/W	
9	—	0	R/W	
8	—	0	R/W	
7	—	0	R/W	
6	—	0	R/W	
5	—	0	R/W	
4	—	0	R/W	
3	—	0	R/W	
2	—	0	R/W	
1	—	0	R/W	

Bit	Bit Name	Initial Value	R/W	Descriptions
0	EEPWT	1	R/W	Data Flash Wait Control This bit inserts one wait cycle when accessing the data flash. 0: No wait cycle is inserted (2-state access) 1: One wait cycle is inserted (3-state access) See section 21, Data Flash (EEPROM), for details.

3.3 Operating Mode Descriptions

3.3.1 Mode 1

Mode 1 is reserved. This LSI does not have mode 1. Do not set this mode.

3.3.2 Mode 2

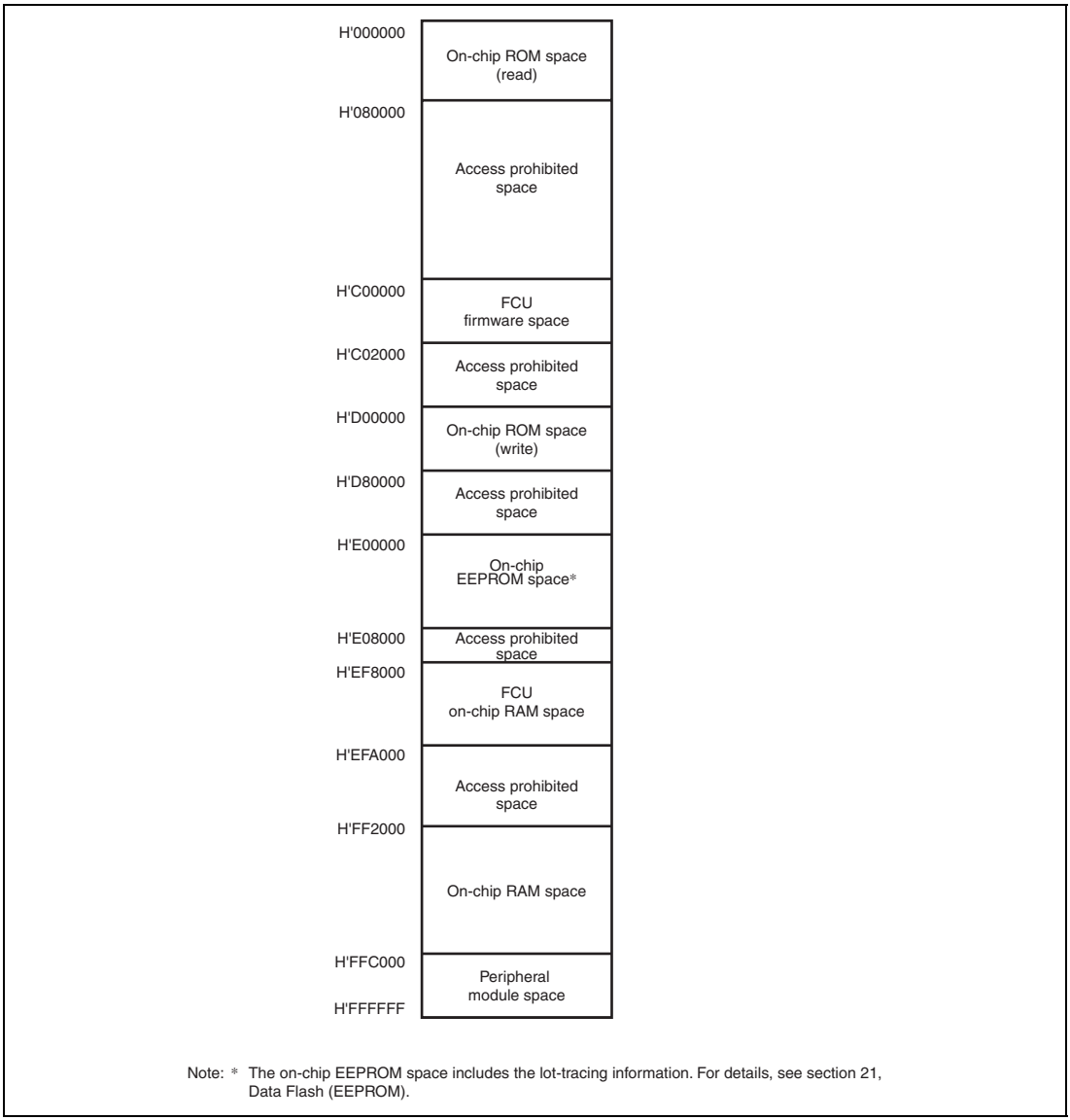
Mode 2 is the boot mode for the flash memory. The operations are the same as that in mode 3 except the flash memory programming/erasing.

3.3.3 Mode 3

Mode 3 is the single-chip mode in which the CPU operates in advanced mode, address space is 16 Mbytes, and the on-chip ROM is enabled.

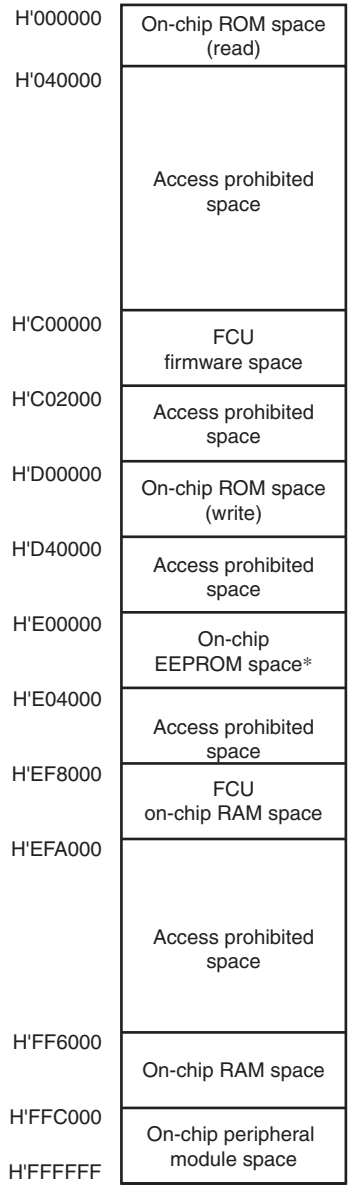
3.3.4 Address Map

Figures 3.1 and 3.2 show the address maps.



Note: * The on-chip EEPROM space includes the lot-tracing information. For details, see section 21, Data Flash (EEPROM).

Figure 3.1 Address Map (H8SX/1727S)



Note: * The on-chip EEPROM space includes the lot-tracing information. For details, see section 21, Data Flash (EEPROM).

Figure 3.2 Address Map (H8SX/1725S)

Section 4 Exception Handling

4.1 Exception Handling Types and Priority

As table 4.1 indicates, exception handling is caused by a reset, a trace, an address error, an interrupt, a trap instruction, and illegal instructions (general illegal instruction and slot illegal instruction). Exception handling is prioritized as shown in table 4.1. If two or more exceptions occur simultaneously, they are accepted and processed in order of priority. Exception sources, the stack structure, and operation of the CPU vary depending on the interrupt control mode. For details on the interrupt control mode, see section 5, Interrupt Controller.

Table 4.1 Exception Types and Priority

Priority	Exception Type	Exception Handling Start Timing
High ↑ Low	Reset	Exception handling starts at the timing of level change from low to high on the $\overline{\text{RES}}$ pin, or when the watchdog timer overflows. The CPU enters the reset state when the $\overline{\text{RES}}$ pin is low.
	Illegal instruction	Exception handling starts when an undefined code is executed.
	Trace*1	Exception handling starts after execution of the current instruction or exception handling, if the trace (T) bit in EXR is set to 1.
	Address error	After an address error has occurred, exception handling starts on completion of instruction execution.
	Interrupt	Exception handling starts after execution of the current instruction or exception handling, if an interrupt request has occurred.*2
	Trap instruction*3	Exception handling starts by execution of a trap instruction (TRAPA).

Notes: 1. Traces are enabled only in interrupt control mode 2. Trace exception handling is not executed after execution of an RTE instruction.
2. Interrupt detection is not performed on completion of ANDC, ORC, XORC, or LDC instruction execution, or on completion of reset exception handling.
3. Trap instruction exception handling requests are accepted at all times in program execution state.

4.2 Exception Sources and Exception Handling Vector Table

Different vector table address offsets are assigned to different exception sources. The vector table addresses are calculated from the contents of the vector base register (VBR) and vector table address offset of the vector number. The start address of the exception service routine is fetched from the exception handling vector table indicated by this vector table address.

Table 4.2 shows the correspondence between the exception sources and vector table address offsets. Table 4.3 shows the calculation method of exception handling vector table addresses.

Since the usable modes differ depending on the product, for details on the available modes, see section 3, MCU Operating Modes.

Table 4.2 Exception Handling Vector Table

		Vector Table Address Offset*1
Exception Source	**Vector Number**	**Advanced Mode**
Reset	0	H'0000 to H'0003
Reserved for system use	1	H'0004 to H'0007
	2	H'0008 to H'000B
	3	H'000C to H'000F
Illegal instruction	4	H'0010 to H'0013
Trace	5	H'0014 to H'0017
Reserved for system use	6	H'0018 to H'001B
Interrupt (NMI)	7	H'001C to H'001F
Trap instruction (#0)	8	H'0020 to H'0023
(#1)	9	H'0024 to H'0027
(#2)	10	H'0028 to H'002B
(#3)	11	H'002C to H'002F
CPU address error	12	H'0030 to H'0033
DMA address error*2	13	H'0034 to H'0037
Reserved for system use	14	H'0038 to H'003B
	⎪	⎪
	63	H'00FC to H'00FF

Exception Source		Vector Number	Vector Table Address Offset*1 Advanced Mode
External interrupt	IRQ0	64	H'0100 to H'0103
	IRQ1	65	H'0104 to H'0107
	IRQ2	66	H'0108 to H'010B
	IRQ3	67	H'010C to H'010F
	IRQ4	68	H'0110 to H'0113
	IRQ5	69	H'0114 to H'0117
	IRQ6	70	H'0118 to H'011B
	IRQ7	71	H'011C to H'011F
	IRQ8	72	H'0120 to H'0123
	IRQ9	73	H'0124 to H'0127
	IRQ10	74	H'0128 to H'012B
	IRQ11	75	H'012C to H'012F
	IRQ12	76	H'0130 to H'0133
	IRQ13	77	H'0134 to H'0137
	IRQ14	78	H'0138 to H'013B
Reserved for system use		79	H'013C to H'013F
Internal interrupt*3		80 ⎮ 255	H'0140 to H'0143 ⎮ H'03FC to H'03FF

Notes: 1. Lower 16 bits of the address.
2. A DMA address error is generated by the DTC and DMAC.
3. For details of internal interrupt vectors, see section 5.5, Interrupt Exception Handling Vector Table.

Table 4.3 Calculation Method of Exception Handling Vector Table Address

Exception Source	Calculation Method of Vector Table Address
Reset, CPU address error	Vector table address = (vector table address offset)
Other than above	Vector table address = VBR + (vector table address offset)

[Legend]
VBR: Vector base register
Vector table address offset: See table 4.2.

4.3 Reset

A reset has priority over any other exception. When the $\overline{\text{RES}}$ pin goes low, all processing halts and this LSI enters the reset state. When operation is in progress, hold the $\overline{\text{RES}}$ pin low for at least 100 cycles.

The chip can also be reset by overflow of the watchdog timer. For details, see section 12, Watchdog Timer (WDT).

A reset initializes the internal state of the CPU and the registers of the on-chip peripheral modules. The interrupt control mode is 0 immediately after a reset.

4.3.1 Reset Exception Handling

When the $\overline{\text{RES}}$ pin goes high after being held low for the necessary time, this LSI starts reset exception handling as follows:

1. The internal state of the CPU and the registers of the on-chip peripheral modules are initialized, VBR is cleared to H'00000000, the T bit is cleared to 0 in EXR, and the I bits are set to 1 in EXR and CCR.
2. The reset exception handling vector address is read and transferred to the PC, and program execution starts from the address indicated by the PC.

Figure 4.1 shows an example of the reset sequence.

4.3.2 Interrupts after Reset

If an interrupt is accepted after a reset but before the stack pointer (SP) is initialized, the PC and CCR will not be saved correctly, leading to a program crash. To prevent this, all interrupt requests, including NMI, are disabled immediately after a reset. Since the first instruction of a program is always executed immediately after the reset state ends, make sure that this instruction initializes the stack pointer (example: MOV.L #xx: 32, SP).

4.3.3 On-Chip Peripheral Functions after Reset Release

After the reset state is released, MSTPCRA is initialized to H'1FFF, MSTPCRB to H'FFFF, MSTPCRC to H'FF00, MSTPCRD to H'FF00, and MSTPCRE to H'FF00, and all modules except the DMAC and DTC enter module stop mode.

Consequently, on-chip peripheral module registers cannot be read or written to. Register reading and writing is enabled when module stop mode is canceled.

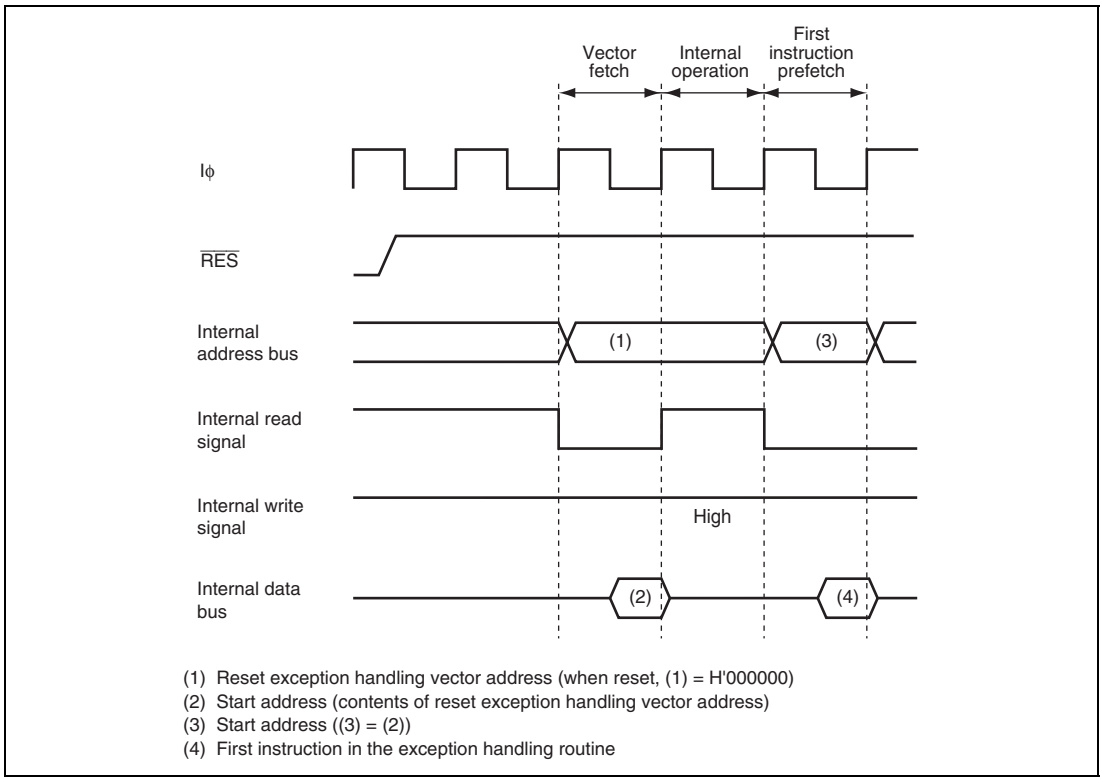


Figure 4.1 Reset Sequence (On-chip ROM Enabled Advanced Mode)

4.4 Traces

Traces are enabled in interrupt control mode 2. Trace mode is not activated in interrupt control mode 0, irrespective of the state of the T bit. For details on interrupt control modes, see section 5, Interrupt Controller. The interrupt control mode should be changed while the T bit is cleared 0.

If the T bit in EXR is set to 1, trace mode is activated. In trace mode, a trace exception occurs on completion of each instruction. Trace mode is not affected by interrupt masking by CCR. Table 4.4 shows the state of CCR and EXR after execution of trace exception handling. Trace mode is canceled by clearing the T bit in EXR to 0 during the trace exception handling. However, the T bit saved on the stack retains its value of 1, and when control is returned from the trace exception handling routine by the RTE instruction, trace mode resumes. Trace exception handling is not carried out after execution of the RTE instruction.

Interrupts are accepted even within the trace exception handling routine.

Table 4.4 Status of CCR and EXR after Trace Exception Handling

	CCR		EXR	
Interrupt Control Mode	**I**	**UI**	**I2 to I0**	**T**
0	Trace exception handling cannot be used.			
2	1	—	—	0

[Legend]
1: Set to 1
0: Cleared to 0
—: Retains the previous value.

4.5 Address Error

4.5.1 Address Error Source

Instruction fetch, stack operation, or data read/write shown in table 4.5 may cause an address error.

Table 4.5 Bus Cycle and Address Error

Bus Cycle			
Type	**Bus Master**	**Description**	**Address Error**
Instruction fetch	CPU	Fetches instructions from even addresses	No (normal)
		Fetches instructions from odd addresses	Occurs
		Fetches instructions from on-chip ROM space (write), on-chip EEPROM space (read/write), FCU on-chip RAM space, and on-chip peripheral module space*[1]	Occurs
		Fetches instructions from space other than on-chip ROM space (write), on-chip EEPROM space (read/write), FCU on-chip RAM space, and on-chip peripheral module space*[1]	No (normal)
		Fetches instructions from access prohibited space*[2]	Occurs
Stack operation	CPU	Accesses stack when the stack pointer value is even address	No (normal)
		Accesses stack when the stack pointer value is odd	Occurs
Data read/write	CPU	Accesses word data from even addresses	No (normal)
		Accesses word data from odd addresses	No (normal)
		Accesses external memory space in single-chip mode	Occurs
		Accesses to access prohibited space*[3]	Occurs

Bus Cycle			
Type	**Bus Master**	**Description**	**Address Error**
Data read/write	DTC/DMAC	Accesses word data from even addresses	No (normal)
		Accesses word data from odd addresses	No (normal)
		Accesses external memory space in single-chip mode	Occurs
		Accesses to access prohibited space*3	Occurs

Notes: 1. For on-chip peripheral module space, see section 6, Bus Controller (BSC).
2. For the access prohibited space, see section 3.3.4, Address Map. However, an address error does not occur when the following access prohibited spaces are accessed: H'800000 to H'BFFFFF and H'C01800 to H'CFFFFF.
3. For the access prohibited space, see section 3.3.4, Address Map. However, an address error does not occur when the following access prohibited spaces are accessed: H'800000 to H'BFFFFF, H'C01800 to H'CFFFFF, H'D40000 to H'DFFFFF, H'E04000 to H'EF7FFF, and H'EF9800 to H'EFFFFF.

4.5.2 Address Error Exception Handling

When an address error occurs, address error exception handling starts after the bus cycle causing the address error ends and current instruction execution completes. The address error exception handling is as follows:

1. The contents of PC, CCR, and EXR are saved in the stack.
2. The interrupt mask bit is updated and the T bit is cleared to 0.
3. An exception handling vector table address corresponding to the address error is generated, the start address of the exception service routine is loaded from the vector table to PC, and program execution starts from that address.

Even though an address error occurs during a transition to an address error exception handling, the address error is not accepted. This prevents an address error from occurring due to stacking for exception handling, thereby preventing infinitive stacking.

If the SP contents are not a multiple of 2 when an address error exception handling occurs, the stacked values (PC, CCR, and EXR) are undefined.

When an address error occurs, the following is performed to halt the DTC and DMAC.

- The ERR bit in DTCCR of the DTC is set to 1.
- The ERRF bit in DMDR_0 of the DMAC is set to 1.

- The DTE bits in DMDRs for all channels of the DMAC are cleared to 0, and transfer is terminated.

Table 4.6 shows the states of CCR and EXR after the address error exception handling.

Table 4.6 States of CCR and EXR after Address Error Exception Handling

	CCR		EXR	
Interrupt Control Mode	**I**	**UI**	**I2 to I0**	**T**
0	1	—	—	—
2	1	—	7	0

[Legend]

1: Set to 1.

0: Cleared to 0.

—: Retains the previous value.

4.6 Interrupts

4.6.1 Interrupt Sources

Interrupt sources are NMI, IRQ0 to IRQ14, and on-chip peripheral modules, as shown in table 4.7.

Table 4.7 Interrupt Sources

Type	Source	Number of Sources
NMI	NMI pin (external input)	1
IRQ0 to IRQ14	Pins $\overline{\text{IRQ0}}$ to $\overline{\text{IRQ14}}$ (external input)	15
On-chip peripheral module	Watchdog timer (WDT)	1
	Memory error (ROM/RAM)	2
	A/D converter	2
	Clock pulse generator (external oscillation stoppage detection)	1
	16-bit timer pulse unit (TPU)	52
	DMA controller (DMAC)	8
	Controller area network (RCAN-TL1)	4
	Hardware LIN	1
	Serial communication interface (SCI)	8
	Renesas serial peripheral interface (RSPI)	16

Different vector numbers and vector table offsets are assigned to different interrupt sources. For vector number and vector table offset, see table 5.2.

4.6.2 Interrupt Exception Handling

Interrupts are controlled by the interrupt controller. The interrupt controller has two interrupt control modes and can assign interrupts other than NMI to eight priority/mask levels to enable multiple-interrupt control. The source to start interrupt exception handling and the vector address differ depending on the product. For details, see section 5, Interrupt Controller.

The interrupt exception handling is as follows:

1. The contents of PC, CCR, and EXR are saved in the stack.
2. The interrupt mask bit is updated and the T bit is cleared to 0.

3. An exception handling vector table address corresponding to the interrupt source is generated, the start address of the exception service routine is loaded from the vector table to PC, and program execution starts from that address.

4.7 Instruction Exception Handling

There are two instructions that cause exception handling: trap instruction and illegal instruction.

4.7.1 Trap Instruction

Trap instruction exception handling starts when a TRAPA instruction is executed. Trap instruction exception handling can be executed at all times in the program execution state. The trap instruction exception handling is as follows:

1. The contents of PC, CCR, and EXR are saved in the stack.
2. The interrupt mask bit is updated and the T bit is cleared to 0.
3. An exception handling vector table address corresponding to the vector number specified in the TRAPA instruction is generated, the start address of the exception service routine is loaded from the vector table to PC, and program execution starts from that address.

A start address is read from the vector table corresponding to a vector number from 0 to 3, as specified in the instruction code.

Table 4.8 shows the states of CCR and EXR after the trap instruction exception handling.

Table 4.8 Status of CCR and EXR after Trap Instruction Exception Handling

	CCR		EXR	
Interrupt Control Mode	**I**	**UI**	**I2 to I0**	**T**
0	1	—	—	—
2	1	—	7	0

[Legend]

1: Set to 1.

0: Cleared to 0.

—: Retains the previous value.

4.7.2 Exception Handling by Illegal Instruction

The illegal instructions are general illegal instructions and slot illegal instructions.

The exception handling by the general illegal instruction starts when an undefined code is executed. The exception handling by the slot illegal instruction starts when the following instruction which is placed in a delay slot (immediately after a delayed branch instruction) is executed: an instruction which consists of two words or more or which changes the contents of PC.

The exception handling by the general illegal instruction and slot illegal instruction is always executable in the program execution state.

The exception handling for the general illegal and slot illegal instructions is as follows:

1. The contents of PC, CCR, and EXR are saved in the stack.
2. The interrupt mask bit is updated and the T bit is cleared to 0.
3. An exception handling vector table address corresponding to the occurred exception is generated, the start address of the exception service routine is loaded from the vector table to PC, and program execution starts from that address.

Table 4.9 shows the states of CCR and EXR after illegal instruction exception handling.

Table 4.9 States of CCR and EXR after Illegal Instruction Exception Handling

	CCR		EXR	
Interrupt Control Mode	**I**	**UI**	**I2 to I0**	**T**
0	1	—	—	—
2	1	—	—	0

[Legend]
1: Set to 1.
0: Cleared to 0.
—: Retains the previous value.

4.8 Stack Status after Exception Handling

Figure 4.2 shows the stack after completion of exception handling.

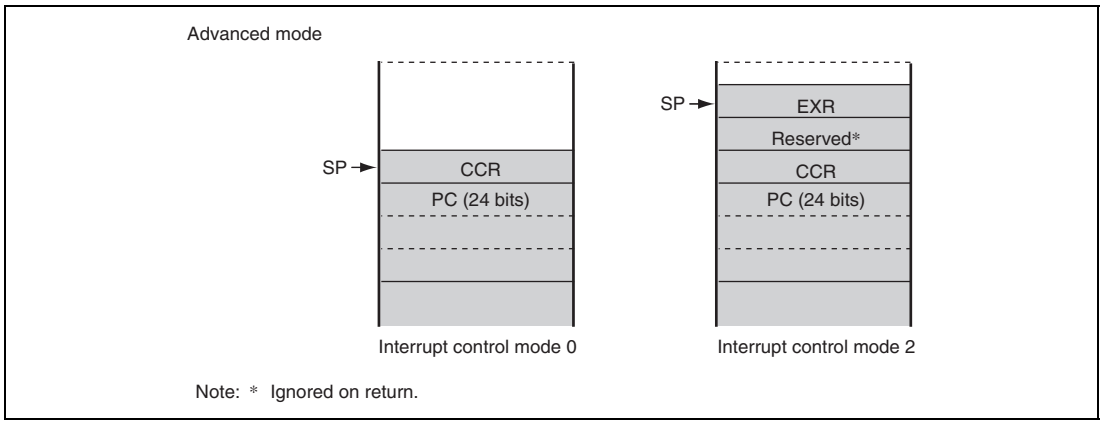


Figure 4.2 Stack Status after Exception Handling

4.9 Usage Note

When performing stack-manipulating access, this LSI assumes that the lowest address bit is 0. The stack should always be accessed by a word transfer instruction or a longword transfer instruction, and the value of the stack pointer (SP: ER7) should always be kept even. Use the following instructions to save registers:

- PUSH.W Rn (or MOV.W Rn, @-SP)
- PUSH.L ERn (or MOV.L ERn, @-SP)

Use the following instructions to restore registers:

- POP.W Rn (or MOV.W @SP+, Rn)
- POP.L ERn (or MOV.L @SP+, ERn)

Performing stack manipulation while SP is set to an odd value leads to an address error. Figure 4.3 shows an example of operation when the SP value is odd.

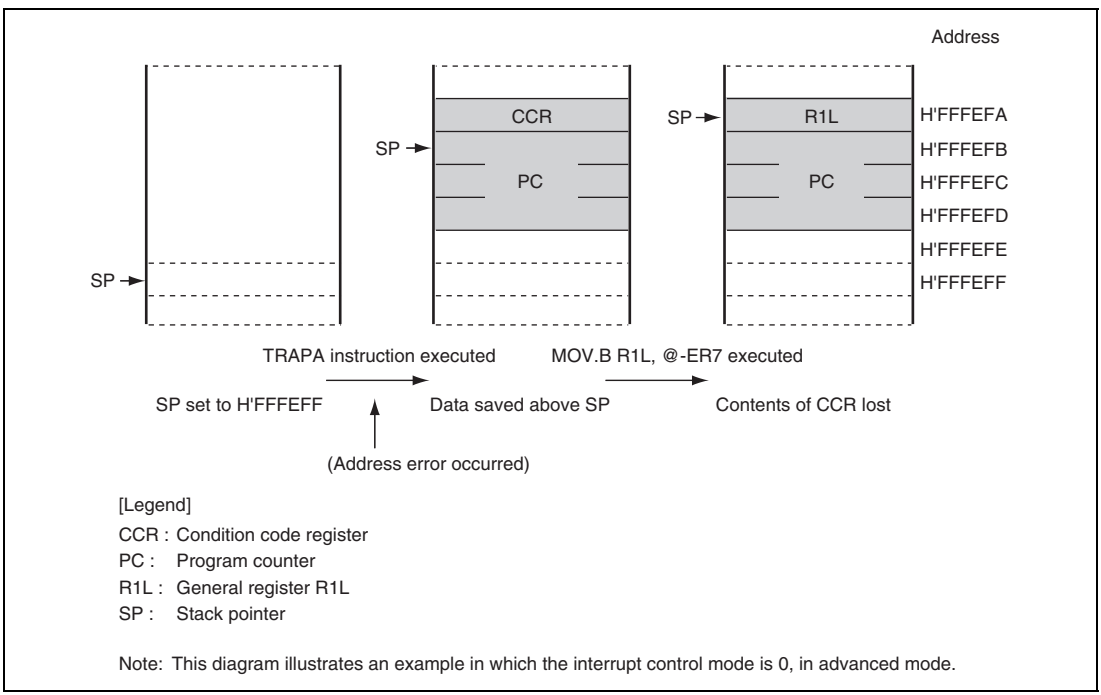


Figure 4.3 Operation when SP Value is Odd

Section 5 Interrupt Controller

5.1 Features

- Two interrupt control modes

  Any of two interrupt control modes can be set by means of bits INTM1 and INTM0 in the interrupt control register (INTCR).
- Priority can be assigned by the interrupt priority register (IPR)

  IPR provides for setting interrupt priority. Eight levels can be set for each module for all interrupts except for the interrupt requests listed below. The following six interrupt requests are given the highest priority of 8, therefore they are accepted at all times.

  — NMI

  — Illegal instructions

  — Trace

  — Trap instructions

  — CPU address error

  — DMA address error*
- Independent vector addresses

  All interrupt sources are assigned independent vector addresses, making it unnecessary for the source to be identified in the interrupt handling routine.
- Sixteen external interrupts

  NMI is the highest-priority interrupt, and is accepted at all times. Rising edge or falling edge detection can be selected for NMI. Falling edge, rising edge, or both edge detection, or level sensing, can be selected for $\overline{\text{IRQ14}}$ to $\overline{\text{IRQ0}}$.
- DTC and DMAC control

  DTC and DMAC can be activated by means of interrupts.
- CPU priority control function

  The priority levels between the CPU, DTC, and DMAC can be set. The priority level of the CPU can be automatically assigned on an exception generation. Priority can be given to the CPU interrupt exception handling over that of the DTC and DMAC transfer.

Note: * A DMA address error is generated by the DTC and DMAC.

A block diagram of the interrupt controller is shown in figure 5.1.

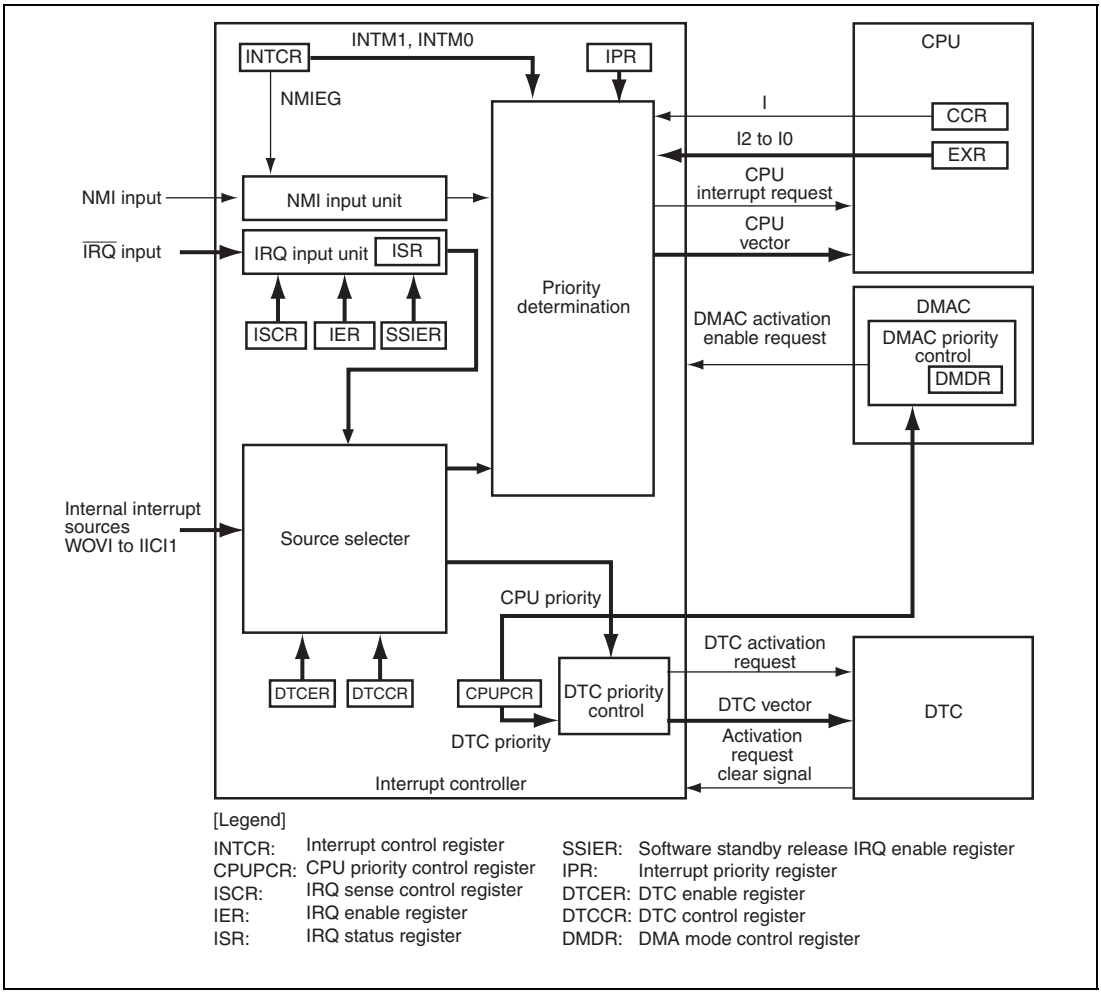


Figure 5.1 Block Diagram of Interrupt Controller

5.2 Input/Output Pins

Table 5.1 shows the pin configuration of the interrupt controller.

Table 5.1 Pin Configuration

Name	I/O	Function
NMI	Input	Nonmaskable External Interrupt Rising or falling edge can be selected.
$\overline{\text{IRQ14}}$ to $\overline{\text{IRQ0}}$	Input	Maskable External Interrupts Rising, falling, or both edges, or level sensing, can be selected.

5.3 Register Descriptions

The interrupt controller has the following registers.

- Interrupt control register (INTCR)
- CPU priority control register (CPUPCR)
- Interrupt priority registers A to P (IPRA to IPRP)
- IRQ enable register (IER)
- IRQ sense control registers H and L (ISCRH, ISCRL)
- IRQ status register (ISR)
- Software standby release IRQ enable register (SSIER)

5.3.1 Interrupt Control Register (INTCR)

INTCR selects the interrupt control mode, and the detected edge for NMI.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	INTM1	INTM0	NMIEG	—	—	—
Initial Value	0	0	0	0	0	0	0	0
R/W	R	R	R/W	R/W	R/W	R	R	R

Bit	Bit Name	Initial Value	R/W	Description
7, 6	—	All 0	R	Reserved These bits are always read as 0 and cannot be modified.
5	INTM1	0	R/W	Interrupt Control Select Mode 1 and 0
4	INTM0	0	R/W	These bits select either of two interrupt control modes for the interrupt controller. 00: Interrupt control mode 0 Interrupts are controlled by I bit in CCR. 01: Setting prohibited. 10: Interrupt control mode 2 Interrupts are controlled by bits I2 to I0 in EXR, and IPR. 11: Setting prohibited.
3	NMIEG	0	R/W	NMI Edge Select Selects the input edge for the NMI pin. 0: Interrupt request generated at falling edge of NMI input 1: Interrupt request generated at rising edge of NMI input
2 to 0	—	All 0	R	Reserved These bits are always read as 0 and cannot be modified.

5.3.2 CPU Priority Control Register (CPUPCR)

CPUPCR sets whether or not the CPU has priority over the DTC and DMAC. The interrupt exception handling by the CPU can be given priority over that of the DTC or DMAC transfer. The priority level of the DTC is assigned by the bits DTCP2 to DTCP0 in CPUPCR. The priority level of the DMAC is assigned by the control register of the DMAC in each channel.

Bit	7	6	5	4	3	2	1	0
Bit Name	CPUPCE	DTCP2	DTCP1	DTCP0	IPSETE	CPUP2	CPUP1	CPUP0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/(W)*	R/(W)*	R/(W)*

Note: * When the IPSETE bit is set to 1, the CPU priority is automatically updated, so these bits cannot be modified.

Bit	Bit Name	Initial Value	R/W	Description
7	CPUPCE	0	R/W	CPU Priority Control Enable Controls the CPU priority control function. Setting this bit to 1 enables the CPU priority control over the DTC and DMAC. 0: CPU always has the lowest priority 1: CPU priority control enabled
6	DTCP2	0	R/W	DTC Priority Level 2 to 0 These bits set the DTC priority level. 000: Priority level 0 (lowest) 001: Priority level 1 010: Priority level 2 011: Priority level 3 100: Priority level 4 101: Priority level 5 110: Priority level 6 111: Priority level 7 (highest)
5	DTCP1	0	R/W	
4	DTCP0	0	R/W	

Bit	Bit Name	Initial Value	R/W	Description
3	IPSETE	0	R/W	Interrupt Priority Set Enable Controls the function which automatically assigns the interrupt priority level of the CPU. Setting this bit to 1 automatically sets bits CPUP2 to CPUP0 by the CPU interrupt mask bit (I bit in CCR or bits I2 to I0 in EXR). 0: Bits CPUP2 to CPUP0 are not updated automatically 1: The interrupt mask bit value is reflected in bits CPUP2 to CPUP0
2	CPUP2	0	R/(W)*	CPU Priority Level 2 to 0 These bits set the CPU priority level. When the CPUPCE is set to 1, the CPU priority control function over the DTC and DMAC becomes valid and the priority of CPU processing is assigned in accordance with the settings of bits CPUP2 to CPUP0. 000: Priority level 0 (lowest) 001: Priority level 1 010: Priority level 2 011: Priority level 3 100: Priority level 4 101: Priority level 5 110: Priority level 6 111: Priority level 7 (highest)
1	CPUP1	0	R/(W)*	
0	CPUP0	0	R/(W)*	

Note: * When the IPSETE bit is set to 1, the CPU priority is automatically updated, so these bits cannot be modified.

5.3.3 Interrupt Priority Registers A to P (IPRA to IPRP)

IPR sets priority (levels 7 to 0) for interrupts other than NMI.

Setting a value in the range from B'000 to B'111 in the 3-bit groups of bits 14 to 12, 10 to 8, 6 to 4, and 2 to 0 assigns a priority level to the corresponding interrupt. For the correspondence between the interrupt sources and the IPR settings, see table 5.2.

Bit	15	14	13	12	11	10	9	8
Bit Name	—	IPR14	IPR13	IPR12	—	IPR10	IPR9	IPR8
Initial Value	0	1	1	1	0	1	1	1
R/W	R	R/W	R/W	R/W	R	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	—	IPR6	IPR5	IPR4	—	IPR2	IPR1	IPR0
Initial Value	0	1	1	1	0	1	1	1
R/W	R	R/W	R/W	R/W	R	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
15	—	0	R	Reserved This bit is always read as 0 and cannot be modified.
14	IPR14	1	R/W	Sets the priority level of the corresponding interrupt source. 000: Priority level 0 (lowest) 001: Priority level 1 010: Priority level 2 011: Priority level 3 100: Priority level 4 101: Priority level 5 110: Priority level 6 111: Priority level 7 (highest)
13	IPR13	1	R/W	
12	IPR12	1	R/W	
11	—	0	R	Reserved This bit is always read as 0 and cannot be modified.

Bit	Bit Name	Initial Value	R/W	Description
10	IPR10	1	R/W	Sets the priority level of the corresponding interrupt source. 000: Priority level 0 (lowest) 001: Priority level 1 010: Priority level 2 011: Priority level 3 100: Priority level 4 101: Priority level 5 110: Priority level 6 111: Priority level 7 (highest)
9	IPR9	1	R/W	
8	IPR8	1	R/W	
7	—	0	R	Reserved This bit is always read as 0 and cannot be modified.
6	IPR6	1	R/W	Sets the priority level of the corresponding interrupt source. 000: Priority level 0 (lowest) 001: Priority level 1 010: Priority level 2 011: Priority level 3 100: Priority level 4 101: Priority level 5 110: Priority level 6 111: Priority level 7 (highest)
5	IPR5	1	R/W	
4	IPR4	1	R/W	
3	—	0	R	Reserved This bit is always read as 0 and cannot be modified.
2	IPR2	1	R/W	Sets the priority level of the corresponding interrupt source. 000: Priority level 0 (lowest) 001: Priority level 1 010: Priority level 2 011: Priority level 3 100: Priority level 4 101: Priority level 5 110: Priority level 6 111: Priority level 7 (highest)
1	IPR1	1	R/W	
0	IPR0	1	R/W	

5.3.4 IRQ Enable Register (IER)

IER enables or disables interrupt requests IRQ14 to IRQ0.

Bit	15	14	13	12	11	10	9	8
Bit Name	—	IRQ14E	IRQ13E	IRQ12E	IRQ11E	IRQ10E	IRQ9E	IRQ8E
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	IRQ7E	IRQ6E	IRQ5E	IRQ4E	IRQ3E	IRQ2E	IRQ1E	IRQ0E
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
15	—	0	R/W	Reserved This bit is always read as 0. The write value should always be 0.
14	IRQ14E	0	R/W	IRQ14 Enable The IRQ14 interrupt request is enabled when this bit is 1.
13	IRQ13E	0	R/W	IRQ13 Enable The IRQ13 interrupt request is enabled when this bit is 1.
12	IRQ12E	0	R/W	IRQ12 Enable The IRQ12 interrupt request is enabled when this bit is 1.
11	IRQ11E	0	R/W	IRQ11 Enable The IRQ11 interrupt request is enabled when this bit is 1.
10	IRQ10E	0	R/W	IRQ10 Enable The IRQ10 interrupt request is enabled when this bit is 1.
9	IRQ9E	0	R/W	IRQ9 Enable The IRQ9 interrupt request is enabled when this bit is 1.
8	IRQ8E	0	R/W	IRQ8 Enable The IRQ8 interrupt request is enabled when this bit is 1.

Bit	Bit Name	Initial Value	R/W	Description
7	IRQ7E	0	R/W	IRQ7 Enable The IRQ7 interrupt request is enabled when this bit is 1.
6	IRQ6E	0	R/W	IRQ6 Enable The IRQ6 interrupt request is enabled when this bit is 1.
5	IRQ5E	0	R/W	IRQ5 Enable The IRQ5 interrupt request is enabled when this bit is 1.
4	IRQ4E	0	R/W	IRQ4 Enable The IRQ4 interrupt request is enabled when this bit is 1.
3	IRQ3E	0	R/W	IRQ3 Enable The IRQ3 interrupt request is enabled when this bit is 1.
2	IRQ2E	0	R/W	IRQ2 Enable The IRQ2 interrupt request is enabled when this bit is 1.
1	IRQ1E	0	R/W	IRQ1 Enable The IRQ1 interrupt request is enabled when this bit is 1.
0	IRQ0E	0	R/W	IRQ0 Enable The IRQ0 interrupt request is enabled when this bit is 1.

5.3.5 IRQ Sense Control Registers H and L (ISCRH, ISCRL)

ISCRH and ISCRL select the source that generates an interrupt request on pins $\overline{\text{IRQ14}}$ to $\overline{\text{IRQ0}}$.

Upon changing the setting of ISCR, IRQnF (n = 0 to 14) in ISR is often set to 1 accidentally through an internal operation. In this case, an interrupt exception handling is executed if an IRQn interrupt request is enabled. In order to prevent such an accidental interrupt from occurring, the setting of ISCR should be changed while the IRQn interrupt is disabled, and then the IRQnF in ISR should be cleared to 0.

- ISCRH

Bit	15	14	13	12	11	10	9	8
Bit Name	—	—	IRQ14SR	IRQ14SF	IRQ13SR	IRQ13SF	IRQ12SR	IRQ12SF
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	IRQ11SR	IRQ11SF	IRQ10SR	IRQ10SF	IRQ9SR	IRQ9SF	IRQ8SR	IRQ8SF
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- ISCRL

Bit	15	14	13	12	11	10	9	8
Bit Name	IRQ7SR	IRQ7SF	IRQ6SR	IRQ6SF	IRQ5SR	IRQ5SF	IRQ4SR	IRQ4SF
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	IRQ3SR	IRQ3SF	IRQ2SR	IRQ2SF	IRQ1SR	IRQ1SF	IRQ0SR	IRQ0SF
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- ISCRH

Bit	Bit Name	Initial Value	R/W	Description
15, 14	—	All 0	R/W	Reserved These bits are always read as 0. The write value should always be 0.
13	IRQ14SR	0	R/W	IRQ14 Sense Control Rise
12	IRQ14SF	0	R/W	IRQ14 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ14}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ14}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ14}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ14}}$
11	IRQ13SR	0	R/W	IRQ13 Sense Control Rise
10	IRQ13SF	0	R/W	IRQ13 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ13}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ13}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ13}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ13}}$
9	IRQ12SR	0	R/W	IRQ12 Sense Control Rise
8	IRQ12SF	0	R/W	IRQ12 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ12}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ12}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ12}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ12}}$

Bit	Bit Name	Initial Value	R/W	Description
7	IRQ11SR	0	R/W	IRQ11 Sense Control Rise
6	IRQ11SF	0	R/W	IRQ11 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ11}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ11}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ11}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ11}}$
5	IRQ10SR	0	R/W	IRQ10 Sense Control Rise
4	IRQ10SF	0	R/W	IRQ10 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ10}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ10}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ10}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ10}}$
3	IRQ9SR	0	R/W	IRQ9 Sense Control Rise
2	IRQ9SF	0	R/W	IRQ9 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ9}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ9}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ9}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ9}}$
1	IRQ8SR	0	R/W	IRQ8 Sense Control Rise
0	IRQ8SF	0	R/W	IRQ8 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ8}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ8}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ8}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ8}}$

- ISCRL

Bit	Bit Name	Initial Value	R/W	Description
15	IRQ7SR	0	R/W	IRQ7 Sense Control Rise
14	IRQ7SF	0	R/W	IRQ7 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ7}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ7}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ7}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ7}}$
13	IRQ6SR	0	R/W	IRQ6 Sense Control Rise
12	IRQ6SF	0	R/W	IRQ6 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ6}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ6}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ6}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ6}}$
11	IRQ5SR	0	R/W	IRQ5 Sense Control Rise
10	IRQ5SF	0	R/W	IRQ5 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ5}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ5}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ5}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ5}}$
9	IRQ4SR	0	R/W	IRQ4 Sense Control Rise
8	IRQ4SF	0	R/W	IRQ4 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ4}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ4}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ4}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ4}}$

Bit	Bit Name	Initial Value	R/W	Description
7	IRQ3SR	0	R/W	IRQ3 Sense Control Rise
6	IRQ3SF	0	R/W	IRQ3 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ3}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ3}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ3}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ3}}$
5	IRQ2SR	0	R/W	IRQ2 Sense Control Rise
4	IRQ2SF	0	R/W	IRQ2 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ2}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ2}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ2}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ2}}$
3	IRQ1SR	0	R/W	IRQ1 Sense Control Rise
2	IRQ1SF	0	R/W	IRQ1 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ1}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ1}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ1}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ1}}$
1	IRQ0SR	0	R/W	IRQ0 Sense Control Rise
0	IRQ0SF	0	R/W	IRQ0 Sense Control Fall 00: Interrupt request generated by low level of $\overline{\text{IRQ0}}$ 01: Interrupt request generated at falling edge of $\overline{\text{IRQ0}}$ 10: Interrupt request generated at rising edge of $\overline{\text{IRQ0}}$ 11: Interrupt request generated at both falling and rising edges of $\overline{\text{IRQ0}}$

5.3.6 IRQ Status Register (ISR)

ISR is an IRQ14 to IRQ0 interrupt request register.

Bit	15	14	13	12	11	10	9	8
Bit Name	—	IRQ14F	IRQ13F	IRQ12F	IRQ11F	IRQ10F	IRQ9F	IRQ8F
Initial Value	0	0	0	0	0	0	0	0
R/W	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*

Bit	7	6	5	4	3	2	1	0
Bit Name	IRQ7F	IRQ6F	IRQ5F	IRQ4F	IRQ3F	IRQ2F	IRQ1F	IRQ0F
Initial Value	0	0	0	0	0	0	0	0
R/W	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*

Note: * Only 0 can be written, to clear the flag.

Bit	Bit Name	Initial Value	R/W	Description
15	—	0	R/W	Reserved This bit is always read as 0. The write value should always be 0.
14	IRQ14F	0	R/(W)*	[Setting condition] • When the interrupt selected by ISCR occurs [Clearing conditions] • Writing 0 after reading IRQnF = 1 • When interrupt exception handling is executed when low-level sensing is selected and $\overline{\text{IRQn}}$ input is high • When IRQn interrupt exception handling is executed when falling-, rising-, or both-edge sensing is selected • When the DTC is activated by an IRQn interrupt, and the DISEL bit in MRB of the DTC is cleared to 0
13	IRQ13F	0	R/(W)*	
12	IRQ12F	0	R/(W)*	
11	IRQ11F	0	R/(W)*	
10	IRQ10F	0	R/(W)*	
9	IRQ9F	0	R/(W)*	
8	IRQ8F	0	R/(W)*	
7	IRQ7F	0	R/(W)*	
6	IRQ6F	0	R/(W)*	
5	IRQ5F	0	R/(W)*	
4	IRQ4F	0	R/(W)*	
3	IRQ3F	0	R/(W)*	
2	IRQ2F	0	R/(W)*	
1	IRQ1F	0	R/(W)*	
0	IRQ0F	0	R/(W)*	

Note: * Only 0 can be written to clear the flag.

5.3.7 Software Standby Release IRQ Enable Register (SSIER)

SSIER selects pins used to leave software standby mode from pins $\overline{\text{IRQ14}}$ to $\overline{\text{IRQ0}}$.

The IRQ interrupt used to leave software standby mode should not be set as the DTC activation source.

Bit	15	14	13	12	11	10	9	8
Bit Name	—	SSI14	SSI13	SSI12	SSI11	SSI10	SSI9	SSI8
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	SSI7	SSI6	SSI5	SSI4	SSI3	SSI2	SSI1	SSI0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
15	—	0	R	Reserved This bit is always read as 0. The write value should always be 0.
14	SSI14	0	R/W	Software Standby Release IRQ Setting These bits select the $\overline{\text{IRQn}}$ pins used to leave software standby mode (n = 14 to 0). 0: IRQn requests are not sampled in software standby mode 1: When an IRQn request occurs in software standby mode, this LSI leaves software standby mode after the oscillation settling time has elapsed
13	SSI13	0	R/W	
12	SSI12	0	R/W	
11	SSI11	0	R/W	
10	SSI10	0	R/W	
9	SSI9	0	R/W	
8	SSI8	0	R/W	
7	SSI7	0	R/W	
6	SSI6	0	R/W	
5	SSI5	0	R/W	
4	SSI4	0	R/W	
3	SSI3	0	R/W	
2	SSI2	0	R/W	
1	SSI1	0	R/W	
0	SSI0	0	R/W	

5.4 Interrupt Sources

5.4.1 External Interrupts

There are sixteen external interrupts: NMI and IRQ14 to IRQ0. These interrupts can be used to leave software standby mode.

(1) NMI Interrupts

Nonmaskable interrupt request (NMI) is the highest-priority interrupt, and is always accepted by the CPU regardless of the interrupt control mode or the settings of the CPU interrupt mask bits. The NMIEG bit in INTCR selects whether an interrupt is requested at the rising or falling edge on the NMI pin.

When an NMI interrupt is generated, the interrupt controller determines that an error has occurred, and performs the following procedure.

- Sets the ERR bit in DTCCR of the DTC to 1.
- Sets the ERRF bit in DMDR_0 of the DMAC to 1.
- The DTE bits in DMDRs for all channels of the DMAC are cleared to 0, and forced transfer is terminated.

(2) IRQn Interrupts

An IRQn interrupt is requested by a signal input on pins $\overline{\text{IRQn}}$ (n = 14 to 0). The IRQn have the following features:

- Using ISCR, it is possible to select whether an interrupt is generated by a low level, falling edge, rising edge, or both edges, on pins $\overline{\text{IRQn}}$.
- Enabling or disabling of interrupt requests IRQn can be selected by IER.
- The interrupt priority can be set by IPR.
- The status of interrupt requests IRQn is indicated in ISR. ISR flags can be cleared to 0 by software. The bit manipulation instructions and memory operation instructions should be used to clear ISR flags.

Detection of IRQn interrupts is enabled through the P1ICR, P5ICR and P6ICR register settings, and does not change regardless of the output setting. However, when a pin is used as an external interrupt input pin, the pin must not be used as an I/O pin for another function by clearing the corresponding DDR bit to 0.

A block diagram of interrupts IRQn is shown in figure 5.2.

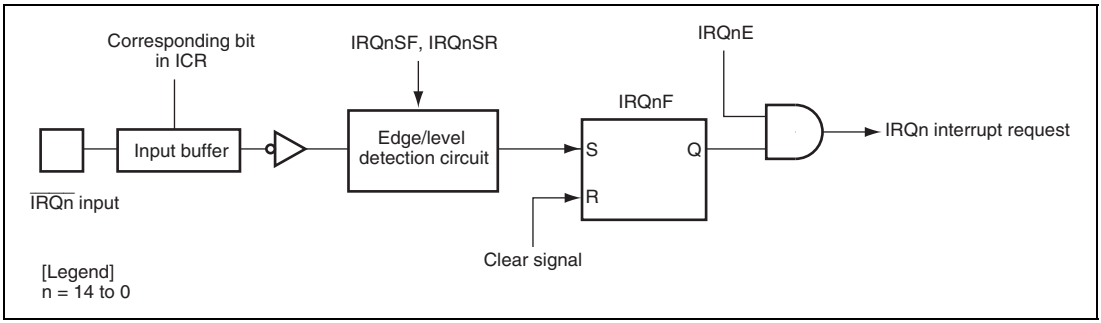


Figure 5.2 Block Diagram of Interrupts IRQn

When the IRQ sensing control in ISCR is set to a low level of signal $\overline{\text{IRQn}}$, the level of $\overline{\text{IRQn}}$ should be held low until an interrupt handling starts. Then set the corresponding input signal $\overline{\text{IRQn}}$ to high in the interrupt handling routine and clear the IRQnF to 0. Interrupts may not be executed when the corresponding input signal $\overline{\text{IRQn}}$ is set to high before the interrupt handling begins.

5.4.2 Internal Interrupts

The sources for internal interrupts from on-chip peripheral modules have the following features:

- For each on-chip peripheral module there are flags that indicate the interrupt request status, and enable bits that enable or disable these interrupts. They can be controlled independently. When the enable bit is set to 1, an interrupt request is issued to the interrupt controller.
- The interrupt priority can be set by means of IPR.
- The DTC and DMAC can be activated by a TPU, SCI, or other interrupt request.
- DTC or DMAC activation can be controlled by the CPU priority control function over DTC or DMAC.

5.5 Interrupt Exception Handling Vector Table

Table 5.2 lists interrupt exception handling sources, vector address offsets, and interrupt priority.

In the default priority order, a lower vector number corresponds to a higher priority. When interrupt control mode 2 is set, priority levels can be changed by setting the IPR contents. The priority for interrupt sources allocated to the same level in IPR follows the default priority, that is, they are fixed.

Table 5.2 Interrupt Sources, Vector Address Offsets, and Interrupt Priority

Classification	Interrupt Source	Vector Number	Vector Table Address Offset* Advanced Mode	IPR	Priority	DTC Activation	DMAC Activation
External pins	NMI	7	H'001C	—	High	—	—
	IRQ0	64	H'0100	IPRA14 to IPRA12		O	—
	IRQ1	65	H'0104	IPRA10 to IPRA8		O	—
	IRQ2	66	H'0108	IPRA6 to IPRA4		O	—
	IRQ3	67	H'010C	IPRA2 to IPRA0		O	—
	IRQ4	68	H'0110	IPRB14 to IPRB12		O	—
	IRQ5	69	H'0114	IPRB10 to IPRB8		O	—
	IRQ6	70	H'0118	IPRB6 to IPRB4		O	—
	IRQ7	71	H'011C	IPRB2 to IPRB0		O	—
	IRQ8	72	H'0120	IPRC14 to IPRC12		O	—
	IRQ9	73	H'0124	IPRC10 to IPRC8		O	—
	IRQ10	74	H'0128	IPRC6 to IPRC4		O	—
	IRQ11	75	H'012C	IPRC2 to IPRC0		O	—
	IRQ12	76	H'0130	IPRD14 to IPRD12		O	—
	IRQ13	77	H'0134	IPRD10 to IPRD8		O	—
	IRQ14	78	H'0138	IPRD6 to IPRD4		O	—
—	Reserved for system use	79	H'013C	IPRD2 to IPRD0		—	—
		80	H'0140	IPRE14 to IPRE12		—	—
WDT	WOVI	81	H'0144	IPRE10 to IPRE8	Low	—	—

Classification	Interrupt Source	Vector Number	Vector Table Address Offset* Advanced Mode	IPR	Priority	DTC Activation	DMAC Activation
Memory error	RAME	82	H'0148	IPRE6 to IPRE4	High	—	—
	Reserved for system use	83	H'014C			—	—
		84	H'0150			—	—
	FIFE	85	H'0154			—	—
A/D_0	ADI0	86	H'0158	IPRF10 to IPRF8		O	O
A/D_1	ADI1	87	H'015C			O	O
TPU_0	TGI0A	88	H'0160	IPRF6 to IPRF4		O	O
	TGI0B	89	H'0164			O	—
	TGI0C	90	H'0168			O	—
	TGI0D	91	H'016C			O	—
	TCI0V	92	H'0170			—	—
TPU_1	TGI1A	93	H'0174	IPRF2 to IPRF0		O	O
	TGI1B	94	H'0178			O	—
	TCI1V	95	H'017C			—	—
	TCI1U	96	H'0180			—	—
TPU_2	TGI2A	97	H'0184	IPRG14 to IPRG12		O	O
	TGI2B	98	H'0188			O	—
	TCI2V	99	H'018C			—	—
	TCI2U	100	H'0190			—	—
TPU_3	TGI3A	101	H'0194	IPRG10 to IPRG8		O	O
	TGI3B	102	H'0198			O	—
	TGI3C	103	H'019C			O	—
	TGI3D	104	H'01A0			O	—
	TCI3V	105	H'01A4			—	—
TPU_4	TGI4A	106	H'01A8	IPRG6 to IPRG4		O	O
	TGI4B	107	H'01AC			O	—
	TCI4V	108	H'01B0			—	—
	TCI4U	109	H'01B4		Low	—	—

Classification	Interrupt Source	Vector Number	Vector Table Address Offset* Advanced Mode	IPR	Priority	DTC Activation	DMAC Activation
TPU_5	TGI5A	110	H'01B8	IPRG2 to IPRG0	High	O	O
	TGI5B	111	H'01BC			O	—
	TCI5V	112	H'01C0			—	—
	TCI5U	113	H'01C4			—	—
Clock Pulse Generator	OSCERI	114	H'01C8	IPRE2 to IPRE0		—	—
—	Reserved for system use	115	H'01CC	—		—	—
		116	H'01D0			—	—
		117	H'01D4			—	—
		118	H'01D8			—	—
		119	H'01DC			—	—
		120	H'01E0			—	—
		121	H'01E4			—	—
		122	H'01E8			—	—
		123	H'01EC			—	—
		124	H'01F0			—	—
		125	H'01F4			—	—
		126	H'01F8			—	—
		127	H'01FC			—	—
DMAC	DMTEND0	128	H'0200	IPRI14 to IPRI12		O	—
	DMTEND1	129	H'0204	IPRI10 to IPRI8		O	—
	DMTEND2	130	H'0208	IPRI6 to IPRI4		O	—
	DMTEND3	131	H'020C	IPRI2 to IPRI0		O	—
RCAN-TL1_0	RM0_0	132	H'0210	IPRJ14 to IPRJ12		O	O
	ERS0_0/OVR0_0/ RM1_0/SLE0_0	133	H'0214			O	—
RCAN-TL1_1	RM0_1	134	H'0218	IPRJ10 to IPRJ8		O	O
	ERS0_1/OVR0_1/ RM1_1/SLE0_1	135	H'021C		Low	O	—

Classification	Interrupt Source	Vector Number	Vector Table Address Offset* Advanced Mode	IPR	Priority	DTC Activation	DMAC Activation
DMAC	DMEEND0	136	H'0220	IPRJ6 to IPRJ4	High	O	—
	DMEEND1	137	H'0224			O	—
	DMEEND2	138	H'0228			O	—
	DMEEND3	139	H'022C			O	—
HW_LIN	HW_LIN	140	H'0230	IPRJ2 to IPRJ0		O	—
—	Reserved for system use	141	H'0234	—		—	—
		142	H'0238			—	—
—	Reserved for system use	143	H'023C	—		—	—
		144	H'0240			—	—
		145	H'0244			—	—
		146	H'0248			—	—
		147	H'024C			—	—
		148	H'0250			—	—
		149	H'0254			—	—
		150	H'0258			—	—
		151	H'025C			—	—
		152	H'0260			—	—
		153	H'0264			—	—
		154	H'0268			—	—
		155	H'026C			—	—
SCI_3	ERI3	156	H'0270	IPRL10 to IPRL8		—	—
	RXI3	157	H'0274			O	O
	TXI3	158	H'0278			O	O
	TEI3	159	H'027C			—	—
SCI_4	ERI4	160	H'0280	IPRL6 to IPRL4		—	—
	RXI4	161	H'0284			O	O
	TXI4	162	H'0288			O	O
	TEI4	163	H'028C		Low	—	—

Classification	Interrupt Source	Vector Number	Vector Table Address Offset* Advanced Mode	IPR	Priority	DTC Activation	DMAC Activation
TPU_6	TGI6A	164	H'0290	IPRL2 to IPRL0	High	O	O
	TGI6B	165	H'0294			O	—
	TGI6C	166	H'0298			O	—
	TGI6D	167	H'029C			O	—
	TGI6V	168	H'02A0	IPRM14 to IPRM2		—	—
TPU_7	TGI7A	169	H'02A4	IPRM10 to IPRM8		O	O
	TGI7B	170	H'02A8			O	—
	TCI7V	171	H'02AC	IPRM6 to IPRM4		—	—
	TCI7U	172	H'02B0			—	—
TPU_8	TGI8A	173	H'02B4	IPRM2 to IPRM0		O	O
	TGI8B	174	H'02B8			O	—
	TCI8V	175	H'02BC	IPRN14 to IPRN12		—	—
	TCI8U	176	H'02C0			—	—
TPU_9	TGI9A	177	H'02C4	IPRN10 to IPRN8		O	O
	TGI9B	178	H'02C8			O	—
	TGI9C	179	H'02CC			O	—
	TGI9D	180	H'02D0			O	—
	TCI9V	181	H'02D4	IPRN6 to IPRN4		—	—
TPU_10	TGI10A	182	H'02D8	IPRN2 to IPRN0		O	O
	TGI10B	183	H'02DC			O	—
	Reserved for system use	184	H'02E0			O	—
	Reserved for system use	185	H'02E4			O	—
	TCI10V	186	H'02E8	IPRO14 to IPRO12		O	—
	TCI10U or reserved for system use	187	H'02EC			—	—
TPU_11	TGI11A	188	H'02F0	IPRO10 to IPRO8		O	O
	TGI11B	189	H'02F4			O	—
	TCI11V	190	H'02F8	IPRO6 to IPRO4		—	—
	TCI11U	191	H'02FC		Low	—	—

Classification	Interrupt Source	Vector Number	Vector Table Address Offset* Advanced Mode	IPR	Priority	DTC Activation	DMAC Activation
—	Reserved for system use	192	H'0300	—	High	—	—
		193	H'0304			—	—
		194	H'0308			—	—
		195	H'030C			—	—
RSPI_0	SPEI_0	196	H'0310	IPRP14 to IPRP12		—	—
	SPRI_0	197	H'0314			O	O
	SPTI_0	198	H'0318			O	O
RSPI_1	SPEI_1	199	H'031C	IPRP10 to IPRP8		—	—
	SPRI_1	200	H'0320			O	O
	SPTI_1	201	H'0324			O	O
RSPI_2	SPEI_2	202	H'0328	IPRP6 to IPRP4		—	—
	SPRI_2	203	H'032C			O	O
	SPTI_2	204	H'0330			O	O
RSPI_3	SPEI_3	205	H'0334	IPRP2 to IPRP0		—	—
	SPRI_3	206	H'0338			O	O
	SPTI_3	207	H'033C			O	O
—	Reserved for system use	208	H'0340	—		—	—
		209	H'0344			—	—
		210	H'0348			—	—
		211	H'034C			—	—
		212	H'0350			—	—
		213	H'0354			—	—
		214	H'0358			—	—
		215	H'035C			—	—
		216	H'0360			—	—
		217	H'0364			—	—
		218	H'0368			—	—
		219	H'036C		Low	—	—

Classification	Interrupt Source	Vector Number	Vector Table Address Offset* Advanced Mode	IPR	Priority	DTC Activation	DMAC Activation
—	Reserved for system use	220	H'0370	—	High	—	—
		221	H'0374			—	—
		222	H'0378			—	—
		223	H'037C			—	—
		224	H'0380			—	—
		225	H'0384			—	—
		226	H'0388			—	—
		227	H'038C			—	—
		228	H'0390			—	—
		229	H'0394			—	—
		230	H'0398			—	—
		231	H'039C			—	—
		232	H'03A0			—	—
		233	H'03A4			—	—
		234	H'03A8			—	—
		235	H'03AC			—	—
		236	H'03B0			—	—
		237	H'03B4			—	—
		238	H'03B8			—	—
		239	H'03BC			—	—
RSPI_0	SPII_0	240	H'03C0	IPRP14 to IPRP12		—	—
RSPI_1	SPII_1	241	H'03C4	IPRP10 to IPRP8		—	—
RSPI_2	SPII_2	242	H'03C8	IPRP6 to IPRP4		—	—
RSPI_3	SPII_3	243	H'03CC	IPRP2 to IPRP0		—	—
—	Reserved for system use	244	H'03D0	—		—	—
		245	H'03D4			—	—
		246	H'03D8			—	—
		247	H'03DC			—	—
		248	H'03E0			—	—
		249	H'03E4		Low	—	—

Classification	Interrupt Source	Vector Number	Vector Table Address Offset* Advanced Mode	IPR	Priority	DTC Activation	DMAC Activation
—	Reserved for system use	250	H'03E8	—	High	—	—
		251	H'03EC			—	—
		252	H'03F0			—	—
		253	H'03F4			—	—
		254	H'03F8			—	—
		255	H'03FC		Low	—	—

Note: * The lower 16 bits of the start address are indicated.

5.6 Interrupt Control Modes and Interrupt Operation

The interrupt controller has two interrupt control modes: interrupt control mode 0 and interrupt control mode 2. The interrupt control mode is selected by INTCR. Table 5.3 shows the differences between interrupt control mode 0 and interrupt control mode 2.

Table 5.3 Interrupt Control Modes

Interrupt Control Mode	Priority Setting Register	Interrupt Mask Bit	Description
0	Default	I	The priority levels of the interrupt sources are fixed default settings. The interrupts except for NMI is masked by the I bit.
2	IPR	I2 to I0	Eight priority levels can be set for interrupt sources except for NMI with IPR. 8-level interrupt mask control is performed by bits I2 to I0.

5.6.1 Interrupt Control Mode 0

In interrupt control mode 0, interrupt requests except for NMI are masked by the I bit in CCR of the CPU. Figure 5.3 shows a flowchart of the interrupt acceptance operation in this case.

1. If an interrupt request occurs when the corresponding interrupt enable bit is set to 1, the interrupt request is sent to the interrupt controller.
2. If the I bit in CCR is set to 1, only an NMI interrupt is accepted, and other interrupt requests are held pending. If the I bit is cleared to 0, an interrupt request is accepted.
3. For multiple interrupt requests, the interrupt controller selects the interrupt request with the highest priority, sends the request to the CPU, and holds other interrupt requests pending.
4. When the CPU accepts the interrupt request, it starts interrupt exception handling after execution of the current instruction has been completed.
5. The PC and CCR contents are saved to the stack area during the interrupt exception handling. The PC contents saved on the stack is the address of the first instruction to be executed after returning from the interrupt handling routine.
6. Next, the I bit in CCR is set to 1. This masks all interrupts except NMI.
7. The CPU generates a vector address for the accepted interrupt and starts execution of the interrupt handling routine at the address indicated by the contents of the vector address in the vector table.

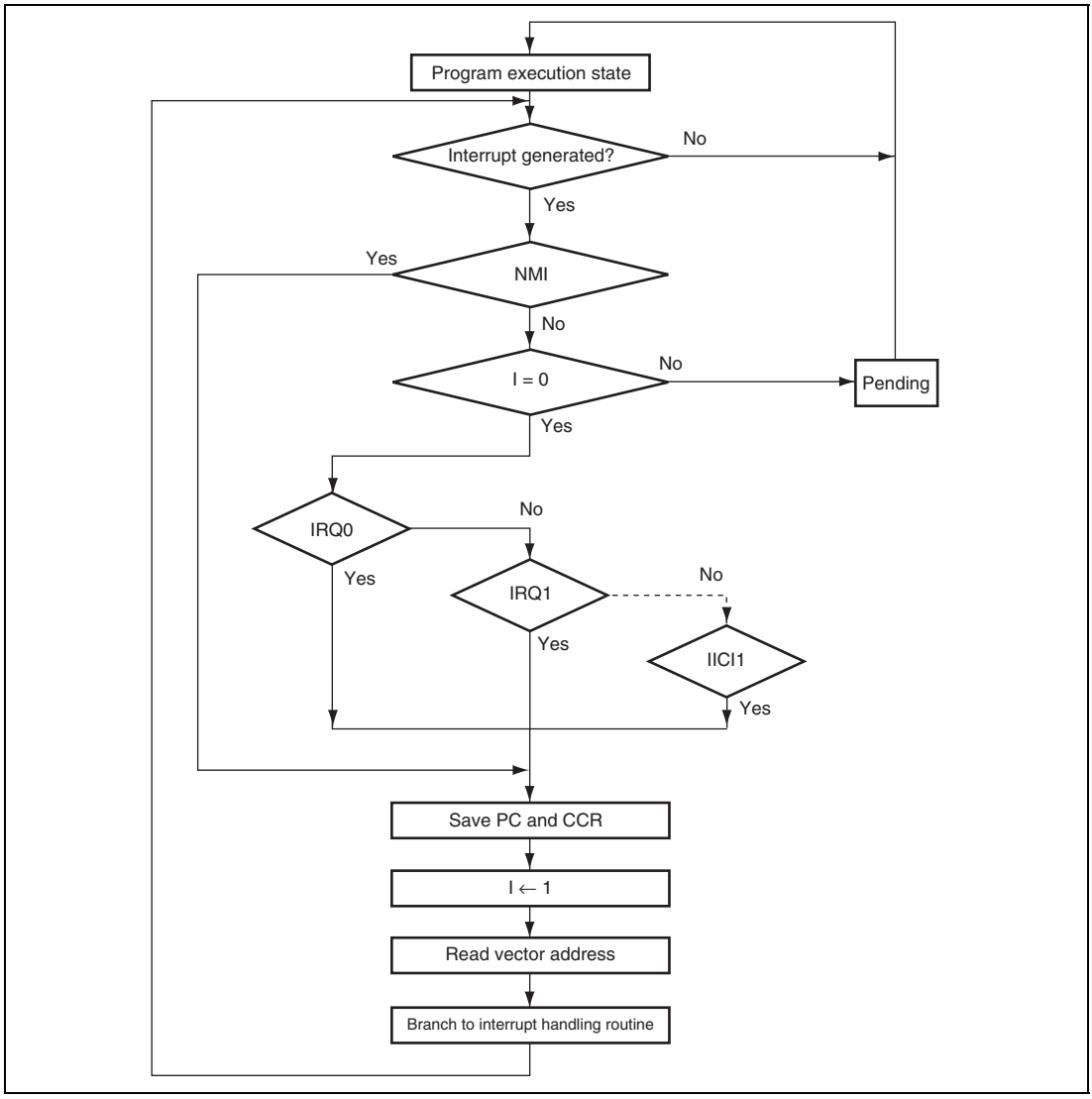


Figure 5.3 Flowchart of Procedure Up to Interrupt Acceptance in Interrupt Control Mode 0

5.6.2 Interrupt Control Mode 2

In interrupt control mode 2, interrupt requests except for NMI are masked by comparing the interrupt mask level (I2 to I0 bits) in EXR of the CPU and the IPR setting. There are eight levels in mask control. Figure 5.4 shows a flowchart of the interrupt acceptance operation in this case.

1. If an interrupt request occurs when the corresponding interrupt enable bit is set to 1, an interrupt request is sent to the interrupt controller.
2. For multiple interrupt requests, the interrupt controller selects the interrupt request with the highest priority according to the IPR setting, and holds other interrupt requests pending. If multiple interrupt requests have the same priority, an interrupt request is selected according to the default setting shown in table 5.2.
3. Next, the priority of the selected interrupt request is compared with the interrupt mask level set in EXR. When the interrupt request does not have priority over the mask level set, it is held pending, and only an interrupt request with a priority over the interrupt mask level is accepted.
4. When the CPU accepts an interrupt request, it starts interrupt exception handling after execution of the current instruction has been completed.
5. The PC, CCR, and EXR contents are saved to the stack area during interrupt exception handling. The PC saved on the stack is the address of the first instruction to be executed after returning from the interrupt handling routine.
6. The T bit in EXR is cleared to 0. The interrupt mask level is rewritten with the priority of the accepted interrupt. If the accepted interrupt is NMI, the interrupt mask level is set to H'7.
7. The CPU generates a vector address for the accepted interrupt and starts execution of the interrupt handling routine at the address indicated by the contents of the vector address in the vector table.

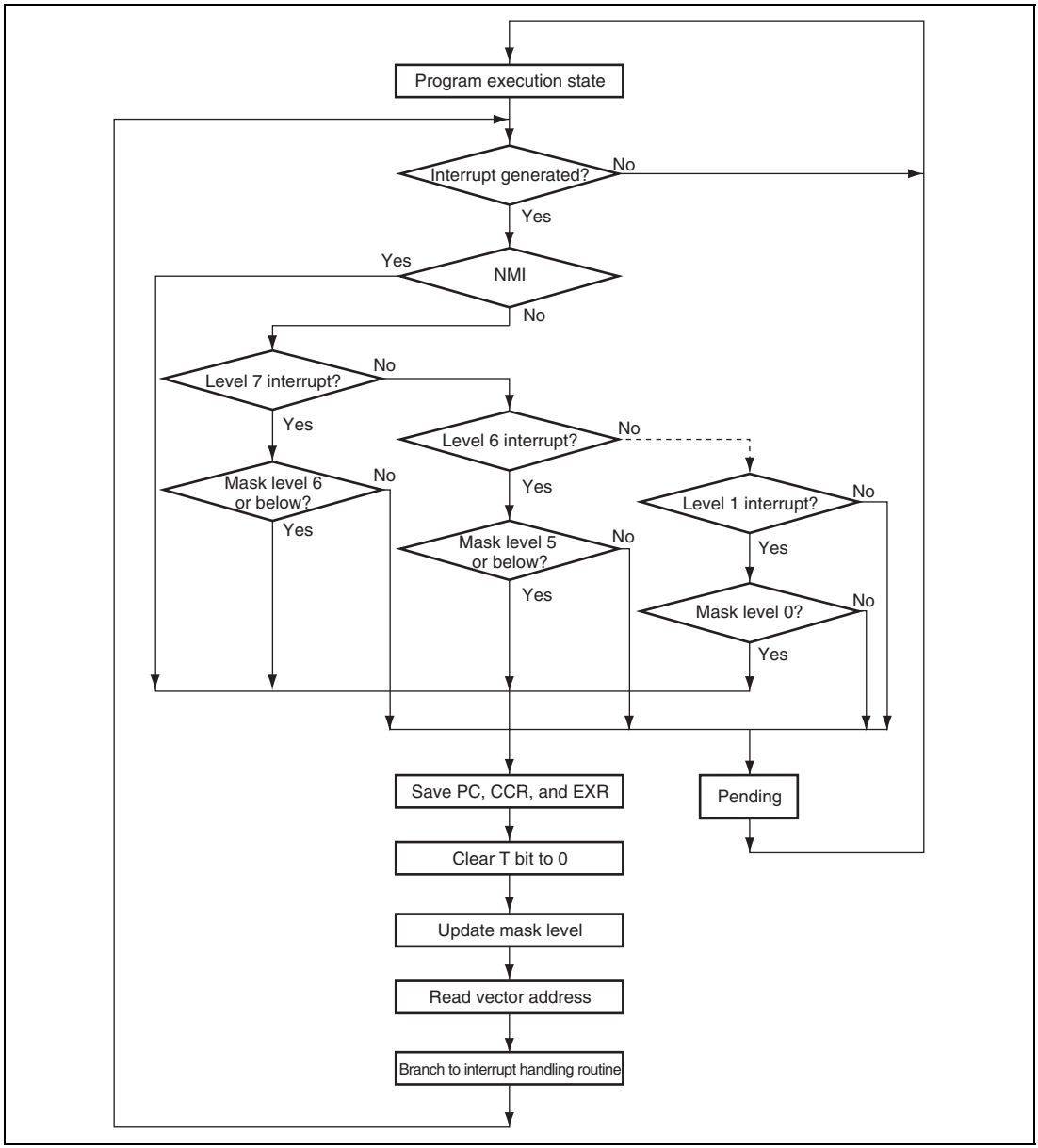


Figure 5.4 Flowchart of Procedure Up to Interrupt Acceptance in Interrupt Control Mode 2

5.6.3 Interrupt Exception Handling Sequence

Figure 5.5 shows the interrupt exception handling sequence. The example is for the case where interrupt control mode 0 is set in advanced mode, and the program area and stack area are in on-chip memory.

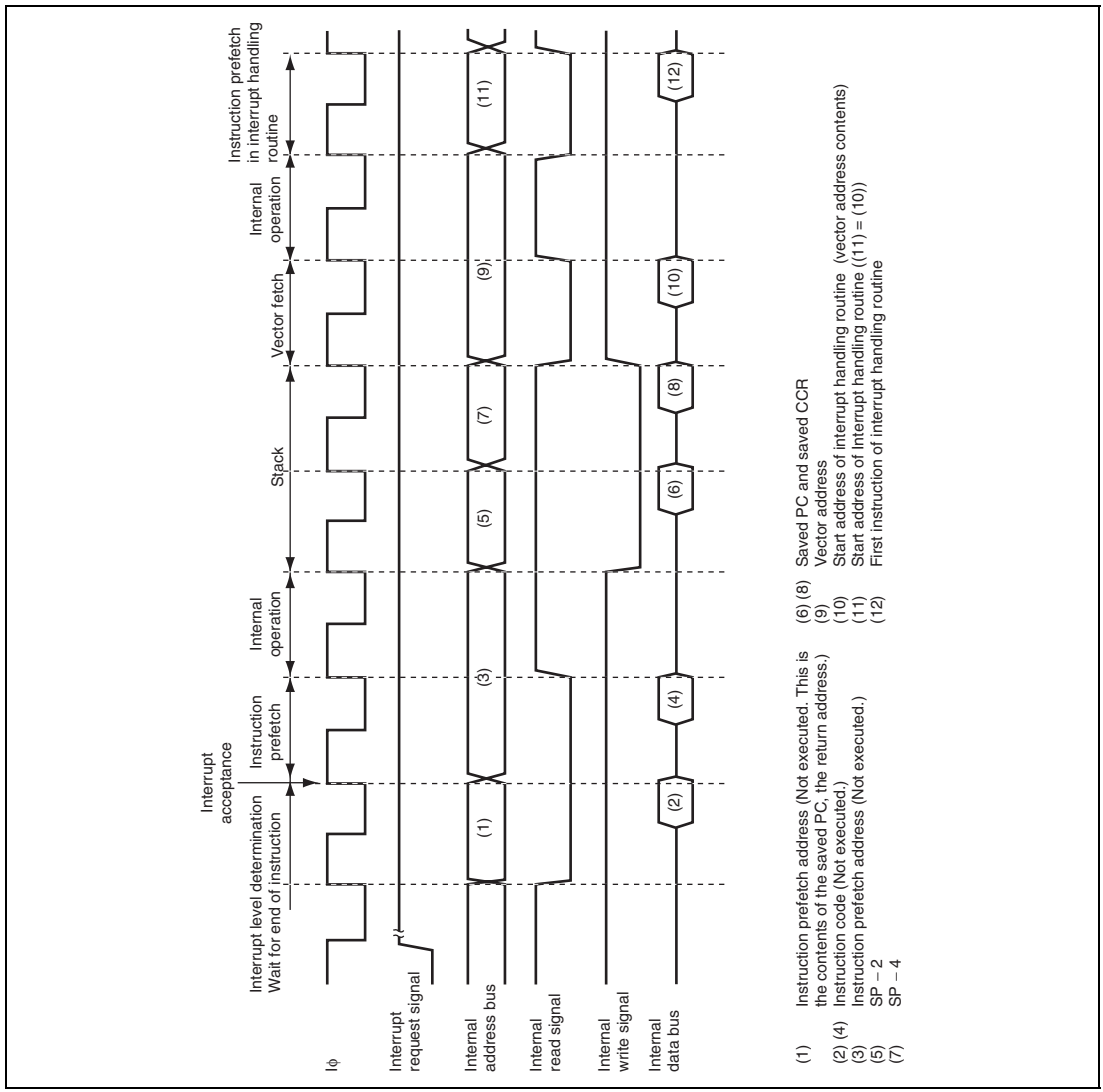


Figure 5.5 Interrupt Exception Handling

5.6.4 Interrupt Response Times

Table 5.4 shows interrupt response times – the interval between generation of an interrupt request and execution of the first instruction in the interrupt handling routine. The symbols for execution states used in table 5.4 are explained in table 5.5.

This LSI is capable of fast word transfer to on-chip memory, so allocating the program area in on-chip ROM and the stack area in on-chip RAM enables high-speed processing.

Table 5.4 Interrupt Response Times

	Advanced Mode	
Execution State	**Interrupt Control Mode 0**	**Interrupt Control Mode 2**
Interrupt priority determination*1	3	3
Number of states until executing instruction ends*2	1 to 19 + 2·S_I	1 to 19 + 2·S_I
PC, CCR, EXR stacking	S_K to 2·S_K*5	2·S_K
Vector fetch	S_h	S_h
Instruction fetch*3	2·S_I	2·S_I
Internal processing*4	2	2
Total (using on-chip memory)	10 to 31	11 to 31

Notes: 1. Two states for an internal interrupt.
2. In the case of the MULXS or DIVXS instruction
3. Prefetch after interrupt acceptance or for an instruction in the interrupt handling routine.
4. Internal operation after interrupt acceptance or after vector fetch
5. When setting the SP value to 4n, the interrupt response time is S_K; when setting to 4n + 2, the interrupt response time is 2·S_K.

Table 5.5 Number of Execution States in Interrupt Handling Routine

	Object of Access				
		External Device			
		8-Bit Bus		16-Bit Bus	
Symbol	On-Chip Memory	2-State Access	3-State Access	2-State Access	3-State Access
Vector fetch S_h	1	8	12 + 4m	4	6 + 2m
Instruction fetch S_I	1	4	6 + 2m	2	3 + m
Stack manipulation S_K	1	8	12 + 4m	4	6 + 2m

[Legend]
m: Number of wait cycles in an external device access.

5.6.5 Activation of DTC and DMAC by Interrupt

The DTC and DMAC can be activated by an interrupt. In this case, the following options are available:

- Interrupt request to the CPU
- Activation request to the DTC
- Activation request to the DMAC
- Combination of the above

For details on interrupt requests that can be used to activate the DTC and DMAC, see table 5.2, section 7, DMA Controller (DMAC), and section 8, Data Transfer Controller (DTC).

Figure 5.6 shows a block diagram of the DTC, DMAC, and interrupt controller.

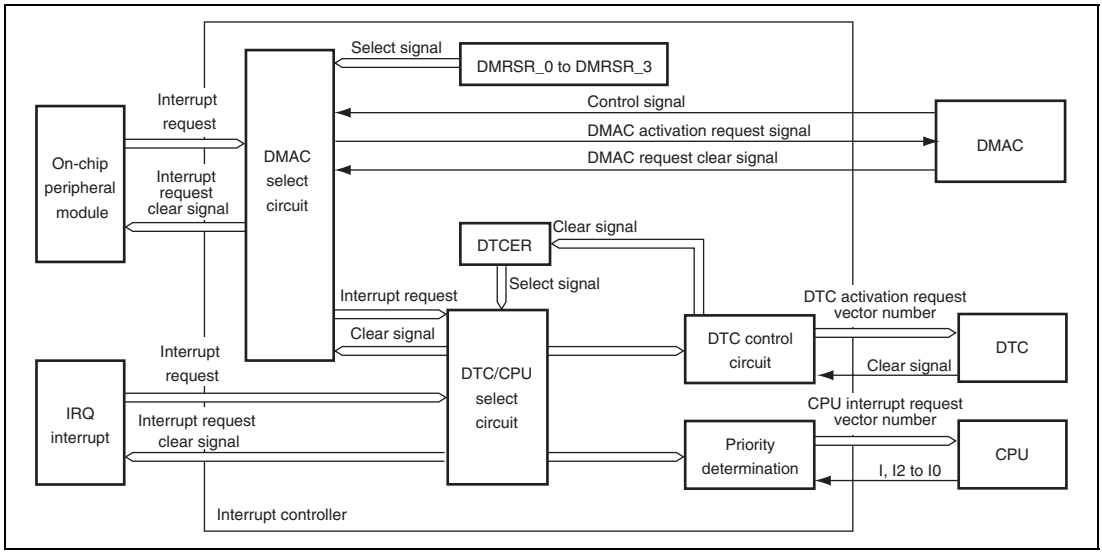


Figure 5.6 Block Diagram of DTC, DMAC, and Interrupt Controller

(1) Selection of Interrupt Sources

The DMAC activation source in each channel is selected by DMRSR. The selected activation source is input to the DMAC through the select circuit. When transfer by an on-chip module interrupt is enabled (DTF1 = 1, DTF0 = 0, and DTA = 1 in DMDR), and the DTE bit in DMDR is set to 1, the DMAC controls the interrupt source that has activated the DMAC, hence this interrupt source cannot become the DTC activation source or the CPU interrupt source.

Each interrupt source other than the interrupts controlled by the DMAC is set for a DTC activation request or a CPU interrupt request by the DTCE bit in DTCERA to DTCERG of the DTC.

Specifying the DISEL bit in MRB of the DTC generates an interrupt request to the CPU by clearing the DTCE bit to 0 after the individual DTC data transfer.

Note that when the DTC performs a predetermined number of data transfers and the transfer counter indicates 0, an interrupt request is also made to the CPU by clearing the DTCE bit to 0 after the DTC data transfer.

When the same interrupt source is set as both the DTC activation source and CPU interrupt source, the DTC must be given priority over the CPU. If IPSETE in CPUPCR is set to 1, the priority is determined according to the IPR setting. Therefore, the CPUP setting or the IPR setting

corresponding to the interrupt source must be set to lower than or equal to the DMAP and DTCP setting. If the CPU is given the higher priority over the DTC, the DTC and DMAC may not be activated, and the data transfer may not be performed.

(2) Priority Determination

The DTC activation source is selected according to the default priority, and the selection is not affected by its mask level or priority level. For respective priority levels, see table 8.1.

(3) Operation Order

If the same interrupt is selected as both the DTC activation source and CPU interrupt source, the CPU interrupt exception handling is performed after the DTC data transfer. If the same interrupt is selected as the DTC and DMAC activation source or CPU interrupt source, respective operations are performed independently.

Table 5.6 lists the selection of interrupt sources and interrupt source clear control by means of the setting of the DTA bit in DMDR of the DMAC, and the DTCE bit in DTCERA to DTCERG and the DISEL bit in MRB of the DTC.

Table 5.6 Interrupt Source Selection and Clear Control

Setting					
DMAC	DTC		Interrupt Source Selection/Clear Control		
DTA	DTCE	DISEL	DMAC	DTC	CPU
0	0	*	O	X	√
	1	0	O	√	X
		1	O	O	√
1	*	*	√	X	X
0	*	*	X	X	X

[Legend]

√: The corresponding interrupt is used. The interrupt source is cleared.
(The interrupt source flag must be cleared in the CPU interrupt handling routine.)

O: The corresponding interrupt is used. The interrupt source is not cleared.

X: The corresponding interrupt is not available.

*: Don't care.

(4) Usage Note

The interrupt sources of the SCI, RCAN, RSPI, TPU, and A/D converter are cleared according to the setting shown in table 5.6, when the DTC or DMAC reads/writes the prescribed register.

To initiate multiple channels for the DTC or DMAC with the same interrupt, the same priority should be assigned.

5.7 Priority Control Function of DTC and DMAC Over CPU

The interrupt controller has a function to control the priority among the DTC, DMAC, and the CPU by assigning a priority levels to the DTC or DMAC over the CPU. Since the priority level can automatically be assigned to the CPU on an interrupt occurrence, it is possible to execute the CPU interrupt exception handling prior to the DTC or DMAC transfer.

The priority level of the CPU is assigned by the bits CPUP2 to CPUP0 in CPUPCR. The priority level of the DTC is assigned by the bits DTCP2 to DTCP0 in CPUPCR. The priority level of the DMAC is assigned by the bits DMAP2 to DMAP0 in DMA mode control registers 0 to 3 (DMDR_0 to DMDR_3) for each channel.

The priority control function of DTC or DMAC over the CPU is enabled by setting the CPUPCE bit in CPUPCR to 1. When the CPUPCE bit is 1, the activation source for the DTC or DMAC is controlled according to the respective priority level.

The DTC activation source is controlled according to the priority level of the CPU indicated by bits CPUP2 to CPUP0 and the priority level of the DTC indicated by bits DTCP2 to DTCP0. If the CPU has priority, the DTC activation source is held. The DTC is activated when the condition by which the activation source is held is cancelled (CPUPCE = 1 and value of bits CPUP2 to CPUP0 is greater than that of bits DTCP2 to DTCP0). The priority level of the DTC is assigned by the DTCP2 to DTCP0 bits in CPUPCR regardless of the activation source.

The DMAC priority level can be assigned in each channel. The DMAC activation source is controlled by both the DMAC priority level, which is assigned by the bits DMAP2 to DMAP0 in the corresponding channel, and the CPU priority level. If the CPU has priority, the activation source for the corresponding channel is held. The activation source is re-enabled when the condition that has held the activation source is cancelled (CPUPCE = 1 and the value of the bits CPUP2 to CPUP0 is greater than that of the bits DMAP2 to DMAP0). When different priority level is assigned for each channel, channels having higher priority than CPU continue transferring processing, while activation sources for channels with lower priority should only be held.

There are two methods for assigning the priority level to the CPU by the IPSETE bit in CPUPCR. Setting the IPSETE bit to 1 enables a function to automatically assign the value of the interrupt mask bit of the CPU to the CPU priority level. Clearing the IPSETE bit to 0 disables the function to automatically assign the priority level. Therefore, the priority level is assigned directly by software rewriting bits CPUP2 to CPUP0. Even if the IPSETE bit is 1, the priority level of the CPU is software assignable by rewriting the interrupt mask bit of the CPU (I bit in CCR or I2 to I0 bits in EXR).

The priority level which is automatically assigned when the IPSETE bit is 1 differs according to the interrupt control mode.

In interrupt control mode 0, the I bit in CCR of the CPU is reflected in the CPUP2 bit. The bits CPUP1 and CPUP0 are fixed 0. In interrupt control mode 2, the values of bits I2 to I0 in EXR of the CPU are reflected in bits CPUP2 to CPUP0.

Table 5.7 shows the CPU priority control.

Table 5.7 CPU Priority Control

Interrupt Control Mode	Interrupt Priority	Interrupt Mask Bit	IPSETE in CPUPCR	Control Status	
				CPUP2 to CPUP0	Updating of CPUP2 to CPUP0
0	Default	I = any	0	B'111 to B'000	Enabled
		I = 0	1	B'000	Disabled
		I = 1	1	B'100	Disabled
2	IPR setting	I2 to I0	0	B'111 to B'000	Enabled
			1	I2 to I0	Disabled

Table 5.8 shows a setting example of the priority control function for DTC and DMAC over the CPU along with the transfer request control state. Although the table shows only one channel example, each channel in the DMAC can have different priority and controls the transfer independently.

Table 5.8 Example of Priority Control Function Setting for DTC and DMAC over CPU and Control State

Interrupt Control Mode	CPUPCE in CPUPCR	CPUP2 to CPUP0	DTCP2 to DTCP0	DMAP2 to DMAP0	Transfer Request Control State	
					DTC	DMAC
0	0	Any	Any	Any	Enabled	Enabled
	1	B'000	B'000	B'000	Enabled	Enabled
		B'100	B'000	B'000	Masked	Masked
		B'100	B'000	B'011	Masked	Masked
		B'100	B'111	B'101	Enabled	Enabled
		B'000	B'111	B'101	Enabled	Enabled
2	0	Any	Any	Any	Enabled	Enabled
	1	B'000	B'000	B'000	Enabled	Enabled
		B'000	B'011	B'101	Enabled	Enabled
		B'011	B'011	B'101	Enabled	Enabled
		B'100	B'011	B'101	Masked	Enabled
		B'101	B'011	B'101	Masked	Enabled
		B'110	B'011	B'101	Masked	Masked
		B'111	B'011	B'101	Masked	Masked
		B'101	B'011	B'101	Masked	Enabled
		B'101	B'110	B'101	Enabled	Enabled

5.8 Usage Notes

5.8.1 Conflict between Interrupt Generation and Disabling

When an interrupt enable bit is cleared to 0 to mask the interrupt, the masking becomes effective after execution of the instruction. When an interrupt enable bit is cleared to 0 by an instruction such as BCLR or MOV, if an interrupt is generated during execution of the instruction, the interrupt concerned will still be enabled on completion of the instruction, and so interrupt exception handling for that interrupt will be executed on completion of the instruction. If this conflict occurs in interrupt control mode 2, the interrupt exception handling with priority equal to or lower than the CPU interrupt mask level specified in the I2 to I0 bits in EXR may be executed on completion of the interrupt disabling instruction. However, if there is an interrupt request with priority over that interrupt, interrupt exception handling will be executed for the interrupt with priority, and another interrupt will be ignored. The same also applies when an interrupt source flag is cleared to 0. In interrupt control mode 2, the same also applies when the interrupt is disabled by modifying the IPR setting. Figure 5.7 shows an example in which the TCIEV bit in TIER of the TPU is cleared to 0. The above conflict will not occur if an enable bit or interrupt source flag is cleared to 0 or IPR is modified while the interrupt is masked. Figure 5.8 shows the interrupt masking procedure to avoid this conflict.

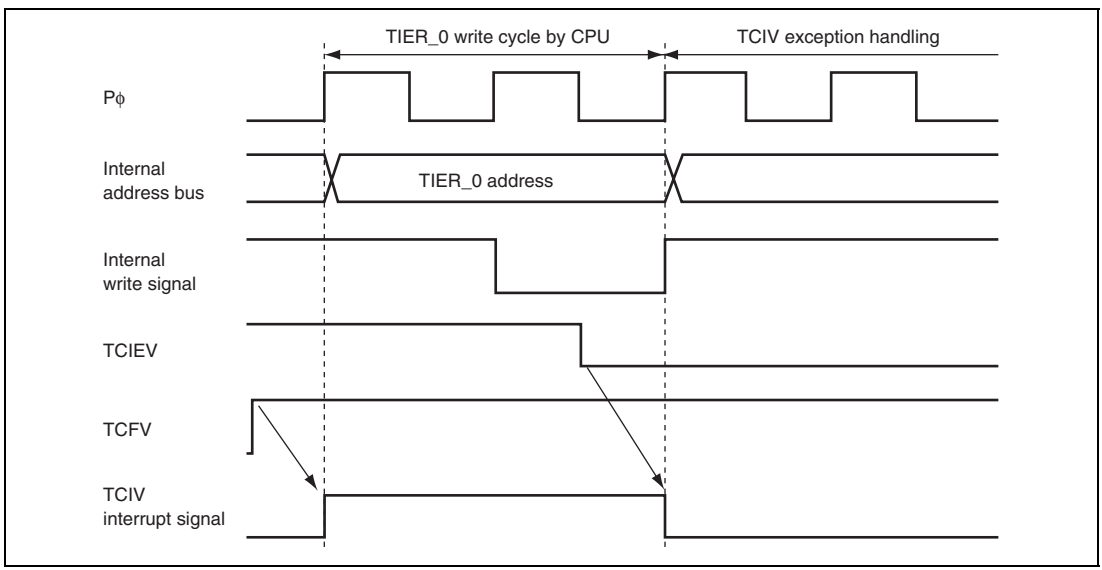


Figure 5.7 Conflict between Interrupt Generation and Disabling

Similarly, when an interrupt is requested immediately before the DTC enable bit is changed to activate the DTC, the DTC activation and the interrupt exception handling by the CPU are both executed. When changing the DTC enable bit, make sure that an interrupt is not requested.

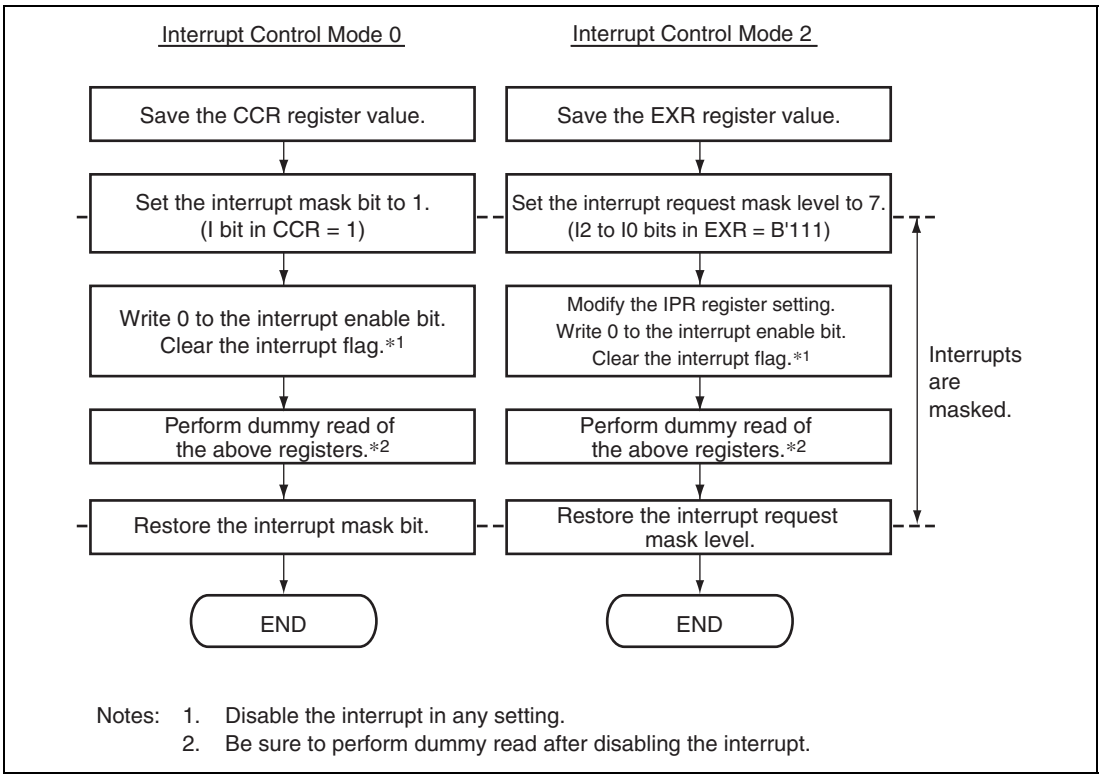


Figure 5.8 Procedure for Avoiding Interrupt Conflict

5.8.2 Instructions that Disable Interrupts

Instructions that disable interrupts immediately after execution are LDC, ANDC, ORC, and XORC. After any of these instructions is executed, all interrupts including NMI are disabled and the next instruction is always executed. When the I bit is set by one of these instructions, the new value becomes valid two states after execution of the instruction ends.

5.8.3 Times when Interrupts are Disabled

There are times when interrupt acceptance is disabled by the interrupt controller. The interrupt controller disables interrupt acceptance for a 3-state period after the CPU has updated the mask level with an LDC, ANDC, ORC, or XORC instruction, and for a period of writing to the registers of the interrupt controller.

5.8.4 Interrupts during Execution of EEPMOV Instruction

Interrupt operation differs between the EEPMOV.B and the EEPMOV.W instructions.

With the EEPMOV.B instruction, an interrupt request (including NMI) issued during the transfer is not accepted until the transfer is completed.

With the EEPMOV.W instruction, if an interrupt request is issued during the transfer, interrupt exception handling starts at the end of the individual transfer cycle. The PC value saved on the stack in this case is the address of the next instruction. Therefore, if an interrupt is generated during execution of an EEPMOV.W instruction, the following coding should be used.

```
L1:     EEPMOV.W
        MOV.W  R4,R4
        BNE    L1
```

5.8.5 Interrupts during Execution of MOVMD and MOVSD Instructions

With the MOVMD or MOVSD instruction, if an interrupt request is issued during the transfer, interrupt exception handling starts at the end of the individual transfer cycle. The PC value saved on the stack in this case is the address of the MOVMD or MOVSD instruction. The transfer of the remaining data is resumed after returning from the interrupt handling routine.

5.8.6 Interrupt Flags of Peripheral Modules

When using the CPU to clear an interrupt source flag and an interrupt enable bit of a peripheral module, be sure to obtain synchronization with the peripheral module by reading the register after clearing them from within the interrupt processing routine. This should be done regardless of the division ratio used to obtain the peripheral clock from the system clock.

Section 6 Bus Controller (BSC)

This LSI has a bus controller (BSC) that manages bus arbitration function, and controls the operation of the internal bus masters: CPU, DMAC, and DTC.

6.1 Features

- Write data buffer function

  Write accesses to the on-chip peripheral module and on-chip memory accesses can be executed in parallel.
- Bus arbitration function

  Includes a bus arbiter that arbitrates bus requests among the CPU, DMAC, and DTC bus masters.
- Multi-clock function

  The on-chip peripheral functions can be operated in synchronization with the peripheral module clock.

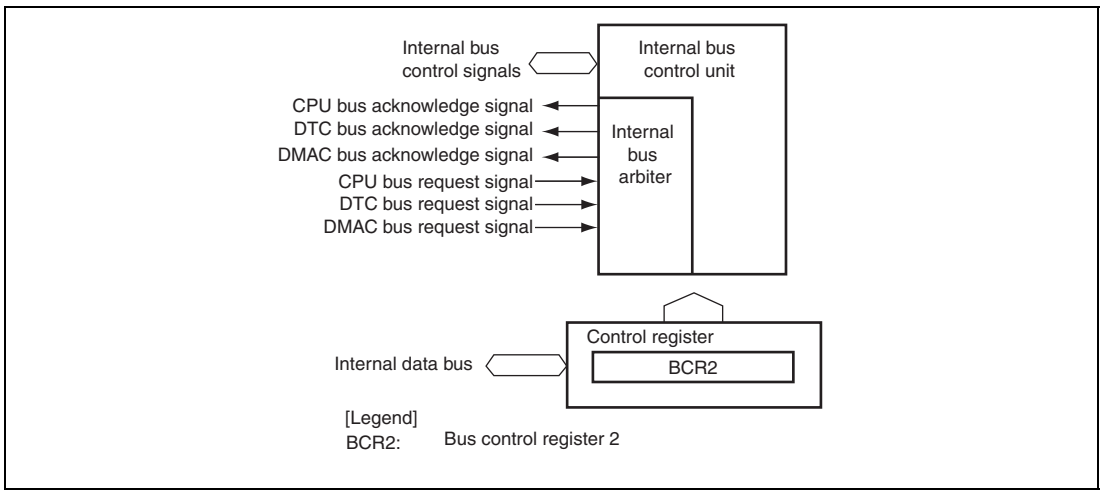


Figure 6.1 Block Diagram of Bus Controller

6.2 Register Descriptions

The bus controller has the following register.

- Bus control register 2 (BCR2)

6.2.1 Bus Control Register 2 (BCR2)

BCR2 is used for bus arbitration control of the CPU, DMAC, and DTC, and the write data buffer function to the peripheral modules.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	IBCCS	—	—	—	PWDBE
Initial Value	0	0	0	0	0	0	1	0
R/W	R	R	R/W	R/W	R	R	R	R/W

Bit	Bit Name	Initial Value	R/W	Description
7, 6	—	All 0	R	Reserved The initial value should not be changed.
5	—	0	R/W	Reserved These bits are always read as 0. The write value should always be 0.
4	IBCCS	0	R/W	Internal Bus Cycle Control Select Selects the method for internal bus arbitration. 0: Releases the bus depending on the priority 1: Executes the bus cycles alternatively when a conflict occurs between a CPU bus request and a bus request by the DMAC or DTC.
3, 2	—	All 0	R	Reserved The initial value should not be changed.
1	—	1	R	Reserved The initial value should not be changed.
0	PWDBE	0	R/W	Peripheral Module Write Data Buffer Enable Specifies whether or not to use the write data buffer function for the peripheral module write cycles. 0: Write data buffer function not used 1: Write data buffer function used

6.3 Bus Configuration

Figure 6.2 shows the internal bus configuration of this LSI. The internal bus of this LSI consists of the following two types.

- Internal bus

  A bus that connects the CPU, DTC, DMAC, on-chip ROM, on-chip RAM, and internal peripheral bus.
- Peripheral bus

  A bus that accesses registers in the DMAC, bus controller, and interrupt controller and registers of peripheral modules such as SCI and timer.

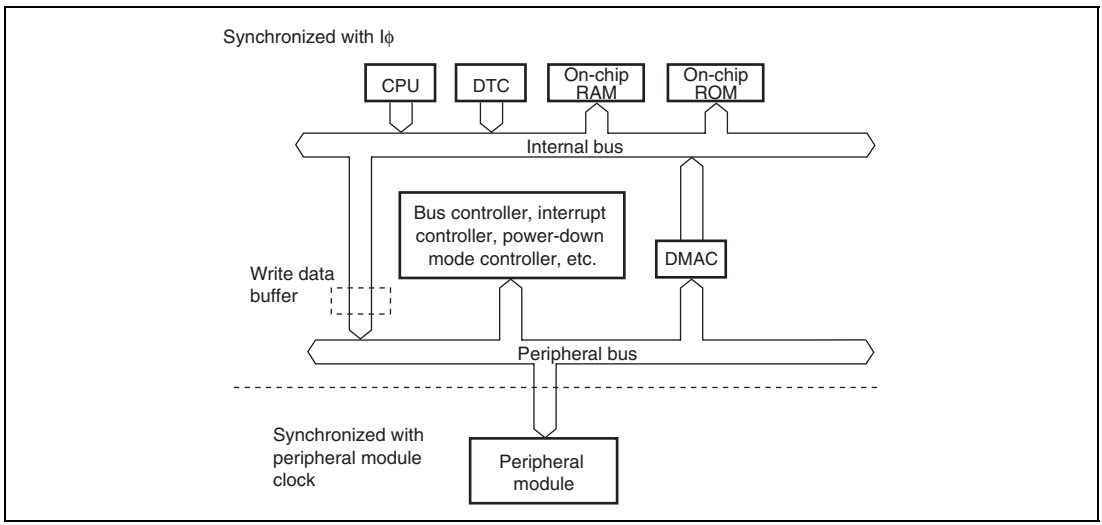


Figure 6.2 Internal Bus Configuration

6.4 Multi-Clock Function and Number of Access Cycles

The internal functions of this LSI operate in synchronization with the system clock (Iϕ) peripheral module clock (Pϕ, Fϕ, Aϕ, or Rϕ) . Table 6.1 shows the synchronization clocks and their corresponding functions.

Table 6.1 Synchronization Clocks and Their Corresponding Functions

Synchronization Clock	Function Name
Iϕ	MCU operating mode, interrupt controller, bus controller, CPU, DMAC, DTC, on-chip memory units, clock pulse generator, and power-down mode control
Pϕ	I/O ports, TPU, PPG, WDT, SCI, RCAN-TL1, LIN, CRC, and RCU
Fϕ	FLASH/EEPROM
Aϕ	A/D
Rϕ	RSPI

The frequency of each synchronization clock (Iϕ, Pϕ, Fϕ, Aϕ, and Rϕ) is specified by the system clock control registers 0 and 1 (SCKCR0 and SCKCR1) independently. For details, see section 22, Clock Pulse Generator.

6.5 Internal Bus

6.5.1 Access to Internal Address Space

The internal address spaces of this LSI are the on-chip ROM space, on-chip RAM space, and register space for the on-chip peripheral modules. The number of cycles necessary for access differs according to the space.

Table 6.2 shows the number of access cycles for each on-chip memory space.

Table 6.2 Number of Access Cycles for On-Chip Memory Spaces

Access Space	Access	Number of Access Cycles
On-chip ROM space	Read	One Iϕ cycle
	Write	Two Iϕ cycles
On-chip RAM space	Read	One Iϕ cycle
	Write	Two Iϕ cycles (One Iϕ cycle)

In access to the registers for on-chip peripheral modules, the number of access cycles differs according to the register to be accessed. When the dividing ratio of the operating clock of a bus master and that of a peripheral module is 1 : n, synchronization cycles using a clock divided by 0 to n-1 are inserted for register access in the same way as for external bus clock division.

Table 6.3 shows the number of access cycles for registers of on-chip peripheral modules.

Table 6.3 Number of Access Cycles for Registers of On-Chip Peripheral Modules

Module to be Accessed	Number of Cycles		Write Data Buffer Function
	Read	Write	
DMAC register	2Iϕ	2Iϕ	Disabled
MCU operating mode, clock pulse generator, power-down control registers, interrupt controller, bus controller, and DTC registers	2Iϕ	3Iϕ	Disabled
I/O port PFCR registers and WDT registers	2Pϕ	3Pϕ	Disabled
RCAN-TL1	2Pϕ	3Pϕ	Enabled
I/O port registers other than PFCR, TPU, PPG, SCI, LIN, CRC, and RCU registers	2Pϕ	2Pϕ	Enabled
FLASH/EEPROM	2Fϕ	3Fϕ	Enabled
A/D	2Aϕ	2Aϕ	Enabled
RSPI	2Rϕ	2Rϕ	Enabled

6.6 Write Data Buffer Function

This LSI has a write data buffer function for the peripheral module access. Using the write data buffer function enables peripheral module writes and on-chip memory access to be executed in parallel. The write data buffer function is made available by setting the PWDBE bit in BCR2 to 1. As for the peripheral module register space in which the write data buffer function is enabled, see table 6.3.

Figure 6.3 shows an example of the timing when the write data buffer function is used. When this function is used, if an internal I/O register write continues for two cycles or longer and then there is an on-chip RAM and an on-chip ROM access, internal I/O register write only is performed in the first two cycles. However, from the next cycle onward an internal memory access and internal I/O register write are executed in parallel rather than waiting until it ends.

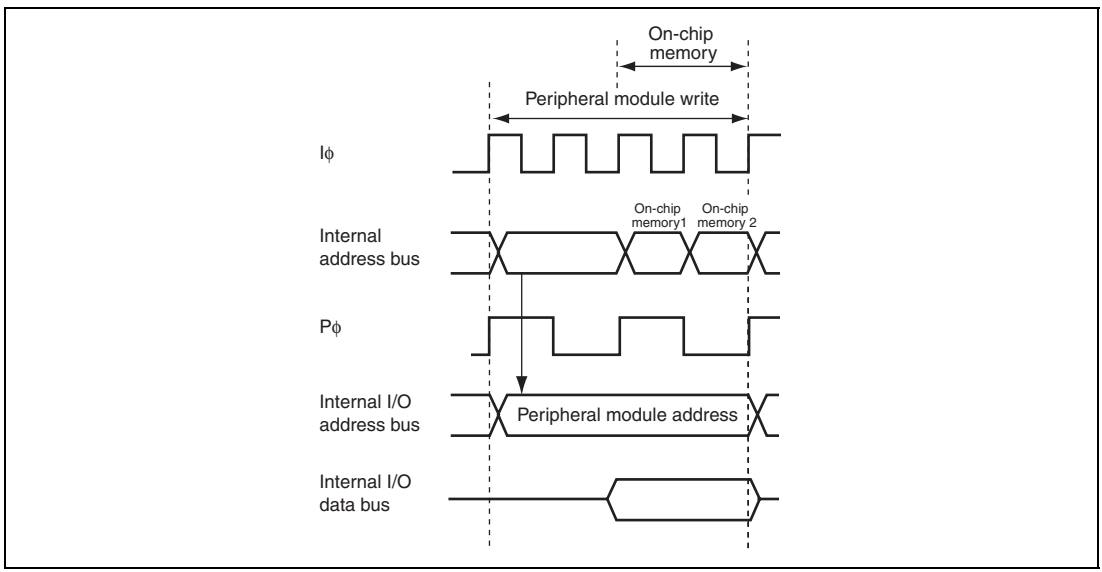


Figure 6.3 Example of Timing when Peripheral Module Write Data Buffer Function is Used

6.7 Bus Arbitration

This LSI has bus arbiters that arbitrate bus mastership operations (bus arbitration). The internal bus arbiter handles accesses by the CPU, DMAC, and DTC.

The bus arbiters determine priorities at the prescribed timing, and permit use of the bus by the bus request acknowledge signal.

6.7.1 Operation

The bus arbiter detects the bus masters' bus request signals, and if the bus is requested, sends a bus request acknowledge signal to the bus master. If there are bus requests from more than one bus master, the bus request acknowledge signal is sent to the one with the highest priority. When a bus master receives the bus request acknowledge signal, it takes possession of the bus until that signal is canceled.

The priority of the internal bus arbitration:

(High) DMAC > DTC > CPU (Low)

If the DMAC or DTC accesses continue, the CPU can be given priority over the DMAC or DTC to execute the bus cycles alternatively between them by setting the IBCCS bit in BCR2 to 1. In this case, the priority between the DMAC and DTC does not change.

6.7.2 Bus Transfer Timing

Even if a bus request is received from a bus master with a higher priority over that of the bus master that has taken control of the bus and is currently operating, the bus is not necessarily transferred immediately. There are specific timings at which each bus master can release the bus.

(1) CPU

The CPU is the lowest-priority bus master, and if a bus request is received from the DMAC or DTC, the bus arbiter transfers the bus to the bus master that issued the request.

The timing for transfer of the bus is at the end of the bus cycle. In sleep mode, the bus is transferred synchronously with the clock.

Note, however, that the bus cannot be transferred in the following cases.

- The word or longword access is performed in some divisions.
- Stack handling is performed in multiple bus cycles.
- Transfer data read or write by memory transfer instructions, block transfer instructions, or TAS instruction.

  (In the block transfer instructions, the bus can be transferred in the write cycle and the following transfer data read cycle.)
- From the target read to write in the bit manipulation instructions or memory operation instructions.

  (In an instruction that performs no write operation according to the instruction condition, up to a cycle corresponding the write cycle)

(2) DTC

The DTC sends the internal bus arbiter a request for the bus when an activation request is generated. When the DTC accesses an external bus space, the DTC first takes control of the bus from the internal bus arbiter and then requests a bus to the external bus arbiter.

Once the DTC takes control of the bus, the DTC continues the transfer processing cycles. If a bus master whose priority is higher than the DTC requests the bus, the DTC transfers the bus to the higher priority bus master. If the IBCSS bit in BCR2 is set to 1, the DTC transfers the bus to the CPU.

Note, however, that the bus cannot be transferred in the following cases.

- During transfer information read
- During the first data transfer
- During transfer information write back

The DTC releases the bus when the consecutive transfer cycles completed.

(3) DMAC

The DMAC sends the internal bus arbiter a request for the bus when an activation request is generated. When the DMAC accesses an external bus space, the DTC first takes control of the bus from the internal bus arbiter and then requests a bus to the external bus arbiter.

Some DMAC transfers are continued once it takes control of the bus. Some DMAC transfers are divided and it releases the bus for each transfer cycle.

- Transfers are continued without bus release between a read cycle and the subsequent write cycle in dual address mode

- Transfers are continued without bus release when a bus master with priority over the DMAC is not requesting the bus, the IBCCS bit in BCR2 is cleared to 0, and either the following conditions are executed
  - While one block of data is being transferred in block transfer mode
  - While data is being transferred in burst mode

Transfer other than above is stopped and the bus is passed when the bus cycle is completed.

6.8 Bus Controller Operation in Reset

In a reset, this LSI, including the bus controller, enters the reset state immediately, and any executing bus cycle is aborted.

6.9 Usage Note

(1) All-Module-Clock-Stop Mode

In this LSI, if the ACSE bit in MSTPCRA is set to 1, and then a SLEEP instruction is executed with the setting for all peripheral module clocks to be stopped (MSTPCRA, MSTPCRB MSTPCRC, MSTPCRD, MSTPCRE = H'FFFFFFFF), and a transition is made to the sleep state, the all-module-clock-stop mode is entered in which the clock supply to the bus controller is also stopped. For details, see section 23, Power-Down Modes.

Section 7 DMA Controller (DMAC)

This LSI has a four-channel DMA controller (DMAC).

7.1 Features

- Maximum of 4-G byte address space can be accessed.
- Byte, word, or longword can be set as data transfer unit.
- Maximum of 4-G bytes (4,294,967,295 bytes) can be set as total transfer size.
  Supports free-running mode in which total transfer size setting is not needed.
- DMAC activation methods are auto-request, on-chip module interrupt, and external request.
  Auto request: Activated by CPU (cycle stealing or burst access can be selected).
  On-chip module interrupt: Interrupt requests from on-chip peripheral modules can be selected as an activation source.
  External request: Low level or falling edge detection of the $\overline{\text{DREQ}}$ signal can be selected (external request is available for all four channels).
- Dual or single address mode can be selected as address mode.
  Dual address mode: Both source and destination are specified by addresses.
  Single address mode: A peripheral device of either source or destination is accessed by the $\overline{\text{DACK}}$ signal and the other is specified by address.
- Normal, repeat, or block transfer can be selected as transfer mode.
  Normal transfer mode: One byte, one word, or one longword data is transferred at a single transfer request.
  Repeat transfer mode: One byte, one word, or one longword data is transferred at a single transfer request.
  Repeat size of data is transferred and then a transfer address returns to the transfer start address.
  Up to 64-Kbyte transfers can be set as repeat size (65,536 bytes/words/longwords).
  Block transfer mode: One block data is transferred at a single transfer request.
  Up to 64-Kbyte data can be set as block size (65,536 bytes/words/longwords).

- Extended repeat area function which repeats the addresses within a specified area using the transfer address with the fixed upper bits (ring buffer transfer can be performed, as an example) is available.

  One bit (two bytes) to 27 bits (128 Mbytes) for transfer source and destination can be set as extended repeat areas.
- Address update can be selected from fixed address, offset addition, and increment or decrement by 1, 2, or 4.

  Address update by offset addition allows transfer data at addresses that are not placed continuously.

Note: External requests and single address mode cannot be used with the H8SX/1720S Group.

- Word or longword data can be transferred to an address that is not aligned with the respective boundary.

  Data is divided according to its address (byte or word) when it is transferred.
- Two types of interrupts can be requested to the CPU.

  A transfer end interrupt is generated after the number of data specified by the transfer counter is transferred.

  A transfer escape end interrupt is generated when the remaining total transfer size is less than the transfer data size at a single transfer request, when the repeat size of data transfer is completed, or when the extended repeat area overflows.

A block diagram of the DMAC is shown in figure 7.1.

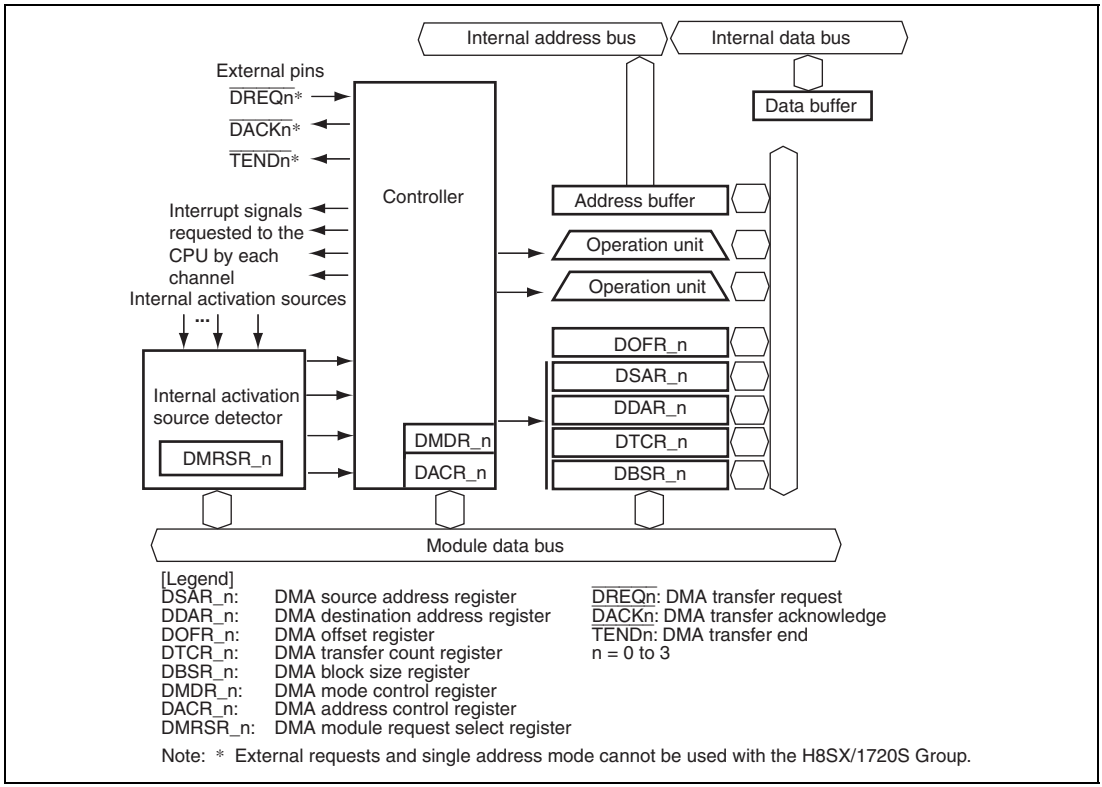


Figure 7.1 Block Diagram of DMAC

7.2 Register Descriptions

The DMAC has the following registers.

Channel 0:

- DMA source address register_0 (DSAR_0)
- DMA destination address register_0 (DDAR_0)
- DMA offset register_0 (DOFR_0)
- DMA transfer count register_0 (DTCR_0)
- DMA block size register_0 (DBSR_0)
- DMA mode control register_0 (DMDR_0)
- DMA address control register_0 (DACR_0)
- DMA module request select register_0 (DMRSR_0)

Channel 1:

- DMA source address register_1 (DSAR_1)
- DMA destination address register_1 (DDAR_1)
- DMA offset register_1 (DOFR_1)
- DMA transfer count register_1 (DTCR_1)
- DMA block size register_1 (DBSR_1)
- DMA mode control register_1 (DMDR_1)
- DMA address control register_1 (DACR_1)
- DMA module request select register_1 (DMRSR_1)

Channel 2:

- DMA source address register_2 (DSAR_2)
- DMA destination address register_2 (DDAR_2)
- DMA offset register_2 (DOFR_2)
- DMA transfer count register_2 (DTCR_2)
- DMA block size register_2 (DBSR_2)
- DMA mode control register_2 (DMDR_2)
- DMA address control register_2 (DACR_2)
- DMA module request select register_2 (DMRSR_2)

Channel 3:

- DMA source address register_3 (DSAR_3)
- DMA destination address register_3 (DDAR_3)
- DMA offset register_3 (DOFR_3)
- DMA transfer count register_3 (DTCR_3)
- DMA block size register_3 (DBSR_3)
- DMA mode control register_3 (DMDR_3)
- DMA address control register_3 (DACR_3)
- DMA module request select register_3 (DMRSR_3)

7.2.1 DMA Source Address Register (DSAR)

DSAR is a 32-bit readable/writable register that specifies the transfer source address. DSAR updates the transfer source address every time data is transferred. When DDAR is specified as the destination address (the DIRS bit in DACR is 1) in single address mode, DSAR is ignored.

Although DSAR can always be read from by the CPU, it must be read from in longwords and must not be written to while data for the channel is being transferred.

Bit	31	30	29	28	27	26	25	24
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	23	22	21	20	19	18	17	16
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	15	14	13	12	11	10	9	8
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7.2.2 DMA Destination Address Register (DDAR)

DDAR is a 32-bit readable/writable register that specifies the transfer destination address. DDAR updates the transfer destination address every time data is transferred. When DSAR is specified as the source address (the DIRS bit in DACR is 0) in single address mode, DDAR is ignored.

Although DDAR can always be read from by the CPU, it must be read from in longwords and must not be written to while data for the channel is being transferred.

Bit	31	30	29	28	27	26	25	24
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	23	22	21	20	19	18	17	16
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	15	14	13	12	11	10	9	8
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7.2.3 DMA Offset Register (DOFR)

DOFR is a 32-bit readable/writable register that specifies the offset to update the source and destination addresses. Although different values are specified for individual channels, the same values must be specified for the source and destination sides of a single channel.

Bit	31	30	29	28	27	26	25	24
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	23	22	21	20	19	18	17	16
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	15	14	13	12	11	10	9	8
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7.2.4 DMA Transfer Count Register (DTCR)

DTCR is a 32-bit readable/writable register that specifies the size of data to be transferred (total transfer size).

To transfer 1-byte data in total, set H'00000001 in DTCR. When H'00000000 is set in this register, it means that the total transfer size is not specified and data is transferred with the transfer counter stopped (free running mode). In this case, no transfer end interrupt by the transfer counter will be generated. When H'FFFFFFFF is set, the total transfer size is 4 Gbytes (4,294,967,295 bytes), which is the maximum size. While data is being transferred, this register indicates the remaining transfer size. The value corresponding to its data access size is subtracted every time data is transferred (byte: −1, word: −2, and longword: −4).

Although DTCR can always be read from by the CPU, it must be read from in longwords and must not be written to while data for the channel is being transferred.

Bit	31	30	29	28	27	26	25	24
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	23	22	21	20	19	18	17	16
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	15	14	13	12	11	10	9	8
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7.2.5 DMA Block Size Register (DBSR)

DBSR specifies the repeat size or block size. DBSR is enabled in repeat transfer mode and block transfer mode and is disabled in normal transfer mode.

Bit	31	30	29	28	27	26	25	24
Bit Name	BKSZH31	BKSZH30	BKSZH29	BKSZH28	BKSZH27	BKSZH26	BKSZH25	BKSZH24
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	23	22	21	20	19	18	17	16
Bit Name	BKSZH23	BKSZH22	BKSZH21	BKSZH20	BKSZH19	BKSZH18	BKSZH17	BKSZH16
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	15	14	13	12	11	10	9	8
Bit Name	BKSZ15	BKSZ14	BKSZ13	BKSZ12	BKSZ11	BKSZ10	BKSZ9	BKSZ8
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	7	6	5	4	3	2	1	0
Bit Name	BKSZ7	BKSZ6	BKSZ5	BKSZ4	BKSZ3	BKSZ2	BKSZ1	BKSZ0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
31 to 16	BKSZH31 to BKSZH16	Undefined	R/W	Specify the repeat size or block size. When H'0001 is set, the repeat or block size is one byte, one word, or one longword. When H'0000 is set, it means the maximum value (refer to table 7.1). While the DMA is in operation, the setting is fixed.
15 to 0	BKSZ15 to BKSZ0	Undefined	R/W	Indicate the remaining repeat or block size while the DMA is in operation. The value is decremented by 1 every time data is transferred. When the remaining size becomes 0, the value of the BKSZH bits is loaded. Set the same value as the BKSZH bits.

Table 7.1 Data Access Size, Valid Bits, and Specifiable Size

Mode	Data Access Size	BKSZH Valid Bits	BKSZ Valid Bits	Specifiable Size (Byte)
Repeat transfer and block transfer	Byte	31 to 16	15 to 0	1 to 65,536
	Word			2 to 131,072
	Longword			4 to 262,144

7.2.6 DMA Mode Control Register (DMDR)

DMDR controls the DMAC operation.

- DMDR0

Bit	31	30	29	28	27	26	25	24
Bit Name	DTE	DACKE	TENDE	—	DREQS	NRD	—	—
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R
Bit	23	22	21	20	19	18	17	16
Bit Name	ACT	—	—	—	ERRF	—	ESIF	DTIF
Initial Value	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/(W)*	R	R/(W)*	R/(W)*
Bit	15	14	13	12	11	10	9	8
Bit Name	DTSZ1	DTSZ0	MDS1	MDS0	TSEIE	—	ESIE	DTIE
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
Bit	7	6	5	4	3	2	1	0
Bit Name	DTF1	DTF0	DTA	—	—	DMAP2	DMAP1	DMAP0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W

Note: * Only 0 can be written to this bit after having been read as 1, to clear the flag.

- DMDR_1 to DMDR_3

Bit	31	30	29	28	27	26	25	24
Bit Name	DTE	DACKE	TENDE	—	DREQS	NRD	—	—
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R
Bit	23	22	21	20	19	18	17	16
Bit Name	ACT	—	—	—	—	—	ESIF	DTIF
Initial Value	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R/(W)*	R/(W)*
Bit	15	14	13	12	11	10	9	8
Bit Name	DTSZ1	DTSZ0	MDS1	MDS0	TSEIE	—	ESIE	DTIE
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
Bit	7	6	5	4	3	2	1	0
Bit Name	DTF1	DTF0	DTA	—	—	DMAP2	DMAP1	DMAP0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W

Note: * Only 0 can be written to this bit after having been read as 1, to clear the flag.

Bit	Bit Name	Initial Value	R/W	Description
31	DTE	0	R/W	Data Transfer Enable Enables/disables a data transfer for the corresponding channel. When this bit is set to 1, it indicates that the DMAC is in operation. Setting this bit to 1 starts a transfer when the auto-request is selected. When the on-chip module interrupt or external request is selected, a transfer request after setting this bit to 1 starts the transfer. While data is being transferred, clearing this bit to 0 stops the transfer. In block transfer mode, if writing 0 to this bit while data is being transferred, this bit is cleared to 0 after the current 1-block size data transfer. If an event which stops (suspends) a transfer occurs externally, this bit is automatically cleared to 0 to stop the transfer. Operating modes and transfer methods must not be changed while this bit is set to 1. 0: Disables a data transfer 1: Enables a data transfer (DMA is in operation) [Clearing conditions] • When the specified size (total transfer size) of transfers is completed • When a transfer is stopped by a repeat size end interrupt • When a transfer is stopped by an extended repeat area overflow interrupt • When a transfer is stopped by a transfer size error interrupt • When clearing this bit to 0 to stop a transfer In block transfer mode, this bit changes after the current block transfer. • When an address error or an NMI interrupt is generated • In the reset state or hardware standby mode* Note: * This LSI does not support hardware standby mode.

Bit	Bit Name	Initial Value	R/W	Description
30	DACKE	0	R/W	$\overline{\text{DACK}}$ Signal Output Enable Enables/disables the $\overline{\text{DACK}}$ signal output in single address mode. This bit is ignored in dual address mode. 0: Disables $\overline{\text{DACK}}$ signal output 1: Enables $\overline{\text{DACK}}$ signal output
29	TENDE	0	R/W	$\overline{\text{TEND}}$ Signal Output Enable Enables/disables the $\overline{\text{TEND}}$ signal output. 0: Disables $\overline{\text{TEND}}$ signal output 1: Enables $\overline{\text{TEND}}$ signal output
28	—	0	R/W	Reserved Initial value should not be changed.
27	DREQS	0	R/W	$\overline{\text{DREQ}}$ Select Selects whether a low level or the falling edge of the $\overline{\text{DREQ}}$ signal used in external request mode is detected. When a block transfer is performed in external request mode, clear this bit to 0. 0: Low level detection 1: Falling edge detection (the first transfer after a transfer enabled is detected on a low level)
26	NRD	0	R/W	Next Request Delay Selects the accepting timing of the next transfer request. 0: Starts accepting the next transfer request after completion of the current transfer 1: Starts accepting the next transfer request one cycle after completion of the current transfer
25, 24	—	All 0	R	Reserved These bits are always read as 0 and cannot be modified.
23	ACT	0	R	Active State Indicates the operating state for the channel. 0: Waiting for a transfer request or in a transfer disabled state by clearing the DTE bit to 0 1: Active state
22 to 20	—	All 0	R	Reserved These bits are always read as 0 and cannot be modified.

Bit	Bit Name	Initial Value	R/W	Description
19	ERRF	0	R/(W)*	System Error Flag Indicates that an address error or an NMI interrupt has been generated. This bit is available only in DMDR_0. Setting this bit to 1 prohibits writing to the DTE bit for all the channels. This bit is reserved in DMDR_1 to DMDR_3. It is always read as 0 and cannot be modified. 0: An address error or an NMI interrupt has not been generated 1: An address error or an NMI interrupt has been generated [Clearing condition] • When clearing to 0 after reading ERRF = 1 [Setting condition] • When an address error or an NMI interrupt has been generated However, when an address error or an NMI interrupt has been generated in DMAC module stop mode, this bit is not set to 1.
18	—	0	R	Reserved This bit is always read as 0 and cannot be modified.
17	ESIF	0	R/(W)*	Transfer Escape Interrupt Flag Indicates that a transfer escape end interrupt has been requested. A transfer escape end means that a transfer is terminated before the transfer counter reaches 0. 0: A transfer escape end interrupt has not been requested 1: A transfer escape end interrupt has been requested [Clearing conditions] • When setting the DTE bit to 1 • When clearing to 0 before reading ESIF = 1 [Setting conditions] • When a transfer size error interrupt is requested • When a repeat size end interrupt is requested • When a transfer end interrupt by an extended repeat area overflow is requested

Bit	Bit Name	Initial Value	R/W	Description
16	DTIF	0	R/(W)*	Data Transfer Interrupt Flag Indicates that a transfer end interrupt by the transfer counter has been requested. 0: A transfer end interrupt by the transfer counter has not been requested 1: A transfer end interrupt by the transfer counter has been requested [Clearing conditions] • When setting the DTE bit to 1 • When clearing to 0 after reading DTIF = 1 [Setting condition] • When DTCR reaches 0 and the transfer is completed
15	DTSZ1	0	R/W	Data Access Size 1 and 0
14	DTSZ0	0	R/W	Select the data access size for a transfer. 00: Byte size (8 bits) 01: Word size (16 bits) 10: Longword size (32 bits) 11: Setting prohibited
13	MDS1	0	R/W	Transfer Mode Select 1 and 0
12	MDS0	0	R/W	Select the transfer mode. 00: Normal transfer mode 01: Block transfer mode 10: Repeat transfer mode 11: Setting prohibited

Bit	Bit Name	Initial Value	R/W	Description
11	TSEIE	0	R/W	Transfer Size Error Interrupt Enable Enables/disables a transfer size error interrupt. When the next transfer is requested while this bit is set to 1 and the contents of the transfer counter is less than the size of data to be transferred at a single transfer request, the DTE bit is cleared to 0. At this time, the ESIF bit is set to 1 to indicate that a transfer size error interrupt has been requested. The sources of a transfer size error are as follows: • In normal or repeat transfer mode, the total transfer size set in DTCR is less than the data access size • In block transfer mode, the total transfer size set in DTCR is less than the block size 0: Disables a transfer size error interrupt request 1: Enables a transfer size error interrupt request
10	—	0	R	Reserved This bit is always read as 0 and cannot be modified.
9	ESIE	0	R/W	Transfer Escape Interrupt Enable Enables/disables a transfer escape end interrupt request. When the ESIF bit is set to 1 with this bit set to 1, a transfer escape end interrupt is requested to the CPU or DTC. The transfer end interrupt request is cleared by clearing this bit or the ESIF bit to 0. 0: Disables a transfer escape end interrupt request 1: Enables a transfer escape end interrupt request
8	DTIE	0	R/W	Data Transfer End Interrupt Enable Enables/disables a transfer end interrupt request by the transfer counter. When the DTIF bit is set to 1 with this bit set to 1, a transfer end interrupt is requested to the CPU or DTC. The transfer end interrupt request is cleared by clearing this bit or the DTIF bit to 0. 0: Disables a transfer end interrupt request 1: Enables a transfer end interrupt request

Bit	Bit Name	Initial Value	R/W	Description
7	DTF1	0	R/W	Data Transfer Factor 1 and 0
6	DTF0	0	R/W	Select a DMAC activation source. When the on-chip module interrupt is selected, the interrupt source should be selected by DMRSR. When the external request setting is selected, the sampling method should be selected by the DREQS bit. 00: Auto request (cycle stealing) 01: Auto request (burst access) 10: On-chip module interrupt 11: External request
5	DTA	0	R/W	Data Transfer Acknowledge This bit is valid in DMA transfer by the on-chip module interrupt source. This bit enables or disables the clearing of the source flag selected by DMRSR. 0: To clear the source in DMA transfer is disabled. Since the on-chip module interrupt source is not cleared in DMA transfer, it should be cleared by the CPU or DTC transfer. 1: To clear the source in DMA transfer is enabled. Since the on-chip module interrupt source is cleared in DMA transfer, it does not require an interrupt by the CPU or DTC transfer.
4, 3	—	All 0	R	Reserved These bits are always read as 0 and cannot be modified.

Bit	Bit Name	Initial Value	R/W	Description
2	DMAP2	0	R/W	DMA Priority Level 2 to 0
1	DMAP1	0	R/W	Select the priority level of the DMAC when using the CPU priority control function over DTC and DMAC. When the CPU has priority over the DMAC, the DMAC masks a transfer request and waits for the timing when the CPU priority becomes lower than the DMAC priority. The priority levels can be set to the individual channels. This bit is valid when the CPUPCE bit in CPUPCR is set to 1. 000: Priority level 0 (low) 001: Priority level 1 010: Priority level 2 011: Priority level 3 100: Priority level 4 101: Priority level 5 110: Priority level 6 111: Priority level 7 (high)
0	DMAP0	0	R/W	

Note: * Only 0 can be written to after reading 1, to clear the flag.

7.2.7 DMA Address Control Register (DACR)

DACR specifies the operating mode and transfer method.

Bit	31	30	29	28	27	26	25	24
Bit Name	AMS	DIRS	—	—	—	RPTIE	ARS1	ARS0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R	R	R	R/W	R/W	R/W

Bit	23	22	21	20	19	18	17	16
Bit Name	—	—	SAT1	SAT0	—	—	DAT1	DAT0
Initial Value	0	0	0	0	0	0	0	0
R/W	R	R	R/W	R/W	R	R	R/W	R/W

Bit	15	14	13	12	11	10	9	8
Bit Name	SARIE	—	—	SARA4	SARA3	SARA2	SARA1	SARA0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R	R	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	DARIE	—	—	DARA4	DARA3	DARA2	DARA1	DARA0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R	R	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
31	AMS	0	R/W	Address Mode Select Selects address mode from single or dual address mode. In single address mode, the $\overline{\text{DACK}}$ pin is enabled according to the DACKE bit in DMDR. 0: Dual address mode 1: Single address mode
30	DIRS	0	R/W	Single Address Direction Select Specifies the data transfer direction in single address mode. This bit is ignored in dual address mode. 0: Specifies DSAR as source address 1: Specifies DDAR as destination address
29 to 27	—	0	R/W	Reserved These bits are always read as 0 and cannot be modified.

Bit	Bit Name	Initial Value	R/W	Description
26	RPTIE	0	R/W	Repeat Size End Interrupt Enable Enables/disables a repeat size end interrupt request. In repeat transfer mode, when the next transfer is requested after completion of a 1-repeat-size data transfer while this bit is set to 1, the DTE bit in DMDR is cleared to 0. At this time, the ESIF bit in DMDR is set to 1 to indicate that a repeat size end interrupt is requested. Even when the repeat area is not specified (ARS1 = 1 and ARS0 = 0), a repeat size end interrupt after a 1-block data transfer can be requested. In addition, in block transfer mode, when the next transfer is requested after 1-block data transfer while this bit is set to 1, the DTE bit in DMDR is cleared to 0. At this time, the ESIF bit in DMDR is set to 1 to indicate that a repeat size end interrupt is requested. 0: Disables a repeat size end interrupt request 1: Enables a repeat size end interrupt request
25	ARS1	0	R/W	Area Select 1 and 0
24	ARS0	0	R/W	Specify the block area or repeat area in block or repeat transfer mode. 00: Specify the block area or repeat area on the source address 01: Specify the block area or repeat area on the destination address 10: Do not specify the block area or repeat area 11: Setting prohibited
23, 22	—	All 0	R	Reserved These bits are always read as 0 and cannot be modified.
21	SAT1	0	R/W	Source Address Update Mode 1 and 0
20	SAT0	0	R/W	Select the update method of the source address (DSAR). When DSAR is not specified as the transfer source in single address mode, this bit is ignored. 00: Source address is fixed 01: Source address is updated by adding the offset 10: Source address is updated by adding 1, 2, or 4 according to the data access size 11: Source address is updated by subtracting 1, 2, or 4 according to the data access size

Bit	Bit Name	Initial Value	R/W	Description
19, 18	—	All 0	R	Reserved These bits are always read as 0 and cannot be modified.
17	DAT1	0	R/W	Destination Address Update Mode 1 and 0
16	DAT0	0	R/W	Select the update method of the destination address (DDAR). When DDAR is not specified as the transfer destination in single address mode, this bit is ignored. 00: Destination address is fixed 01: Destination address is updated by adding the offset 10: Destination address is updated by adding 1, 2, or 4 according to the data access size 11: Destination address is updated by subtracting 1, 2, or 4 according to the data access size
15	SARIE	0	R/W	Source Address Extended Repeat Area Overflow Interrupt Enable Enables/disables an interrupt request for an extended area overflow on the source address. When an extended repeat area overflow on the source address occurs while this bit is set to 1, the DTE bit in DMDR is cleared to 0. At this time, the ESIF bit in DMDR is set to 1 to indicate an interrupt by an extended repeat area overflow on the source address is requested. When block transfer mode is used with the extended repeat area function, an interrupt is requested after completion of a 1-block size transfer. When setting the DTE bit in DMDR of the channel for which a transfer has been stopped to 1, the transfer is resumed from the state when the transfer is stopped. When the extended repeat area is not specified, this bit is ignored. 0: Disables an interrupt request for an extended area overflow on the source address 1: Enables an interrupt request for an extended area overflow on the source address
14, 13	—	All 0	R	Reserved These bits are always read as 0 and cannot be modified.

Bit	Bit Name	Initial Value	R/W	Description
12	SARA4	0	R/W	Source Address Extended Repeat Area
11	SARA3	0	R/W	Specify the extended repeat area on the source address (DSAR). With the extended repeat area, the specified lower address bits are updated and the remaining upper address bits are fixed. The extended repeat area size is specified from 4 to 128 Mbytes in units of byte and a power of 2.
10	SARA2	0	R/W	
9	SARA1	0	R/W	
8	SARA0	0	R/W	
				When the lower address is overflowed from the extended repeat area by address update, the address becomes the start address and the end address of the area for address addition and subtraction, respectively.
				When an overflow in the extended repeat area occurs with the SARIE bit set to 1, an interrupt can be requested. Table 7.2 shows the settings and areas of the extended repeat area.
7	DARIE	0	R/W	Destination Address Extended Repeat Area Overflow Interrupt Enable
				Enables/disables an interrupt request for an extended area overflow on the destination address.
				When an extended repeat area overflow on the destination address occurs while this bit is set to 1, the DTE bit in DMDR is cleared to 0. At this time, the ESIF bit in DMDR is set to 1 to indicate an interrupt by an extended repeat area overflow on the destination address is requested.
				When block transfer mode is used with the extended repeat area function, an interrupt is requested after completion of a 1-block size transfer. When setting the DTE bit in DMDR of the channel for which the transfer has been stopped to 1, the transfer is resumed from the state when the transfer is stopped.
				When the extended repeat area is not specified, this bit is ignored.
				0: Disables an interrupt request for an extended area overflow on the destination address
				1: Enables an interrupt request for an extended area overflow on the destination address
6, 5	—	All 0	R	Reserved
				These bits are always read as 0 and cannot be modified.

Bit	Bit Name	Initial Value	R/W	Description
4	DARA4	0	R/W	Destination Address Extended Repeat Area
3	DARA3	0	R/W	Specify the extended repeat area on the destination address (DDAR). With the extended repeat area, the specified lower address bits are updated and the remaining upper address bits are fixed. The extended repeat area size is specified from 4 to 128 Mbytes in units of byte and a power of 2. When the lower address is overflowed from the extended repeat area by address update, the address becomes the start address and the end address of the area for address addition and subtraction, respectively. When an overflow in the extended repeat area occurs with the DARIE bit set to 1, an interrupt can be requested. Table 7.2 shows the settings and areas of the extended repeat area.
2	DARA2	0	R/W	
1	DARA1	0	R/W	
0	DARA0	0	R/W	

Table 7.2 Settings and Areas of Extended Repeat Area

SARA4 to SARA0 or DARA4 to DARA0	Extended Repeat Area
00000	Not specified
00001	2 bytes specified as extended repeat area by the lower 1 bit of the address
00010	4 bytes specified as extended repeat area by the lower 2 bits of the address
00011	8 bytes specified as extended repeat area by the lower 3 bits of the address
00100	16 bytes specified as extended repeat area by the lower 4 bits of the address
00101	32 bytes specified as extended repeat area by the lower 5 bits of the address
00110	64 bytes specified as extended repeat area by the lower 6 bits of the address
00111	128 bytes specified as extended repeat area by the lower 7 bits of the address
01000	256 bytes specified as extended repeat area by the lower 8 bits of the address
01001	512 bytes specified as extended repeat area by the lower 9 bits of the address
01010	1 Kbyte specified as extended repeat area by the lower 10 bits of the address
01011	2 Kbytes specified as extended repeat area by the lower 11 bits of the address
01100	4 Kbytes specified as extended repeat area by the lower 12 bits of the address
01101	8 Kbytes specified as extended repeat area by the lower 13 bits of the address
01110	16 Kbytes specified as extended repeat area by the lower 14 bits of the address
01111	32 Kbytes specified as extended repeat area by the lower 15 bits of the address
10000	64 Kbytes specified as extended repeat area by the lower 16 bits of the address
10001	128 Kbytes specified as extended repeat area by the lower 17 bits of the address
10010	256 Kbytes specified as extended repeat area by the lower 18 bits of the address
10011	512 Kbytes specified as extended repeat area by the lower 19 bits of the address
10100	1 Mbyte specified as extended repeat area by the lower 20 bits of the address
10101	2 Mbytes specified as extended repeat area by the lower 21 bits of the address
10110	4 Mbytes specified as extended repeat area by the lower 22 bits of the address
10111	8 Mbytes specified as extended repeat area by the lower 23 bits of the address
11000	16 Mbytes specified as extended repeat area by the lower 24 bits of the address
11001	32 Mbytes specified as extended repeat area by the lower 25 bits of the address
11010	64 Mbytes specified as extended repeat area by the lower 26 bits of the address
11011	128 Mbytes specified as extended repeat area by the lower 27 bits of the address
111××	Setting prohibited

[Legend]

×: Don't care

7.2.8 DMA Module Request Select Register (DMRSR)

DMRSR is an 8-bit readable/writable register that specifies the on-chip module interrupt source. The vector number of the interrupt source is specified in eight bits. However, 0 is regarded as no interrupt source. For the vector numbers of the interrupt sources, refer to table 7.6.

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7.3 Transfer Modes

Table 7.3 shows the DMAC transfer modes. The transfer modes can be specified to the individual channels.

Table 7.3 Transfer Modes

Address Mode	Transfer Mode	Activation Source	Common Function	Address Register: Source	Address Register: Destination
Dual address	• Normal transfer • Repeat transfer • Block transfer Repeat or block size = 1 to 65,536 bytes, 1 to 65,536 words, or 1 to 65,536 longwords	• Auto request (activated by CPU) • On-chip module interrupt • External request	• Total transfer size: 1 to 4 Gbytes or not specified • Offset addition • Extended repeat area function	DSAR	DDAR
Single address	• Instead of specifying the source or destination address registers, data is directly transferred from/to the external device using the $\overline{\text{DACK}}$ pin • The same settings as above are available other than address register setting (e.g., above transfer modes can be specified) • One transfer can be performed in one bus cycle (the types of transfer modes are the same as those of dual address modes)			DSAR/ $\overline{\text{DACK}}$	$\overline{\text{DACK}}$/ DDAR

When the auto request setting is selected as the activation source, the cycle stealing or burst access can be selected. When the total transfer size is not specified (DTCR = H'00000000), the transfer counter is stopped and the transfer is continued without the limitation of the transfer count.

7.4 Operations

7.4.1 Address Modes

(1) Dual Address Mode

In dual address mode, the transfer source address is specified in DSAR and the transfer destination address is specified in DDAR. A transfer at a time is performed in two bus cycles (when the data bus width is less than the data access size or the access address is not aligned with the boundary of the data access size, the number of bus cycles are needed more than two because one bus cycle is divided into multiple bus cycles).

In the first bus cycle, data at the transfer source address is read and in the next cycle, the read data is written to the transfer destination address.

The read and write cycles are not separated. Other bus cycles (bus cycle by other bus masters, refresh cycle, and external bus released cycle) are not generated between read and write cycles.

The $\overline{\text{TEND}}$ signal output is enabled or disabled by the TENDE bit in DMDR. The $\overline{\text{TEND}}$ signal is output in two bus cycles. When an idle cycle is inserted before the bus cycle, the $\overline{\text{TEND}}$ signal is also output in the idle cycle. The $\overline{\text{DACK}}$ signal is not output.

Figure 7.2 shows an example of the signal timing in dual address mode and figure 7.3 shows the operation in dual address mode.

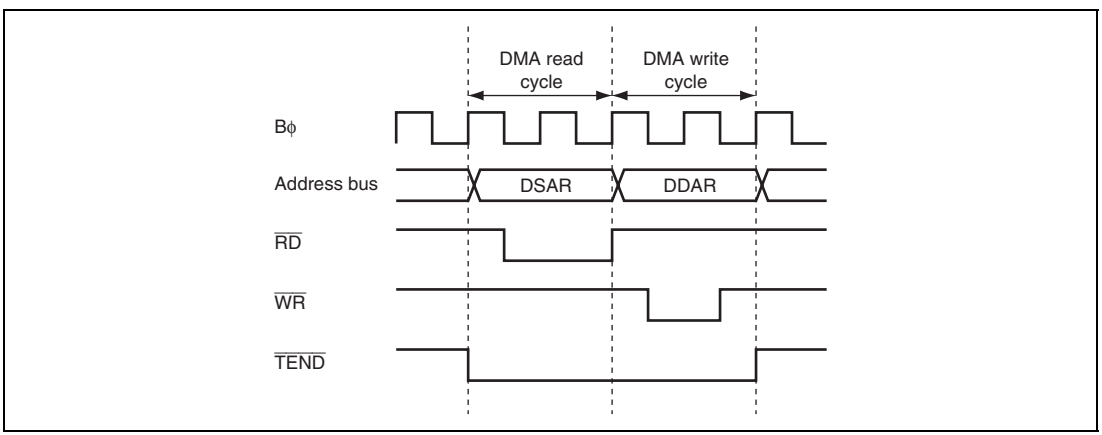


Figure 7.2 Example of Signal Timing in Dual Address Mode

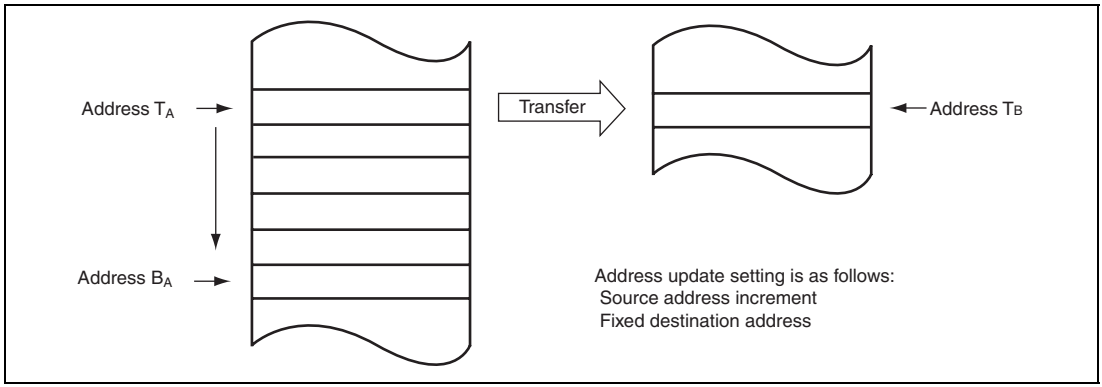


Figure 7.3 Operations in Dual Address Mode

(2) Single Address Mode

In single address mode, data between an external device and an external memory is directly transferred using the $\overline{\text{DACK}}$ pin instead of DSAR or DDAR. A transfer at a time is performed in one bus cycle. In this mode, the data bus width must be the same as the data access size. For details on the data bus width, see section 6, Bus Controller (BSC).

The DMAC accesses an external device as the transfer source or destination by outputting the strobe signal ($\overline{\text{DACK}}$) to the external device with $\overline{\text{DACK}}$ and accesses the other transfer target by outputting the address. Accordingly, the DMA transfer is performed in one bus cycle. Figure 7.4 shows an example of a transfer between an external memory and an external device with the $\overline{\text{DACK}}$ pin. In this example, the external device outputs data on the data bus and the data is written to the external memory in the same bus cycle.

The transfer direction is decided by the DIRS bit in DACR which specifies an external device with the $\overline{\text{DACK}}$ pin as the transfer source or destination. When DIRS = 0, data is transferred from an external memory (DSAR) to an external device with the $\overline{\text{DACK}}$ pin. When DIRS = 1, data is transferred from an external device with the $\overline{\text{DACK}}$ pin to an external memory (DDAR). The settings of registers which are not used as the transfer source or destination are ignored.

The $\overline{\text{DACK}}$ signal output is enabled in single address mode by the DACKE bit in DMDR. The $\overline{\text{DACK}}$ signal is active low.

The $\overline{\text{TEND}}$ signal output is enabled or disabled by the TENDE bit in DMDR. The $\overline{\text{TEND}}$ signal is output in one bus cycle. When an idle cycle is inserted before the bus cycle, the $\overline{\text{TEND}}$ signal is also output in the idle cycle.

Figure 7.5 shows an example of timing charts in single address mode and figure 7.6 shows an example of operation in single address mode.

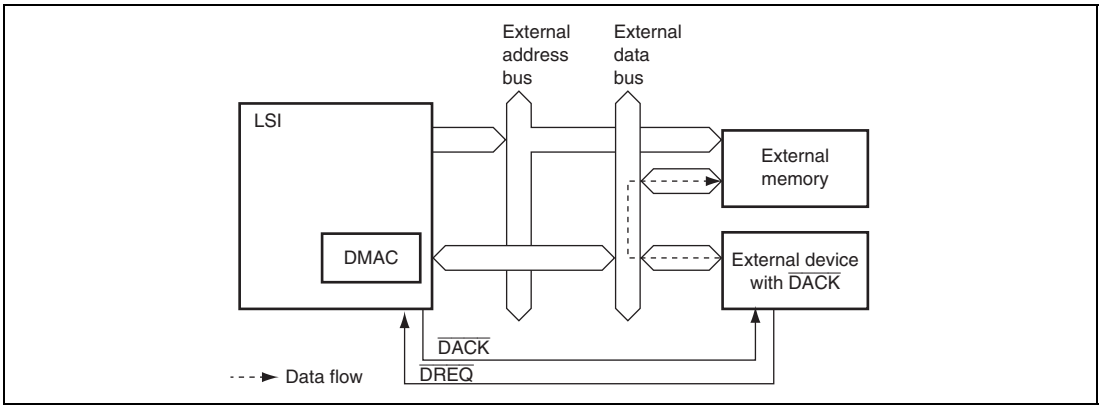


Figure 7.4 Data Flow in Single Address Mode

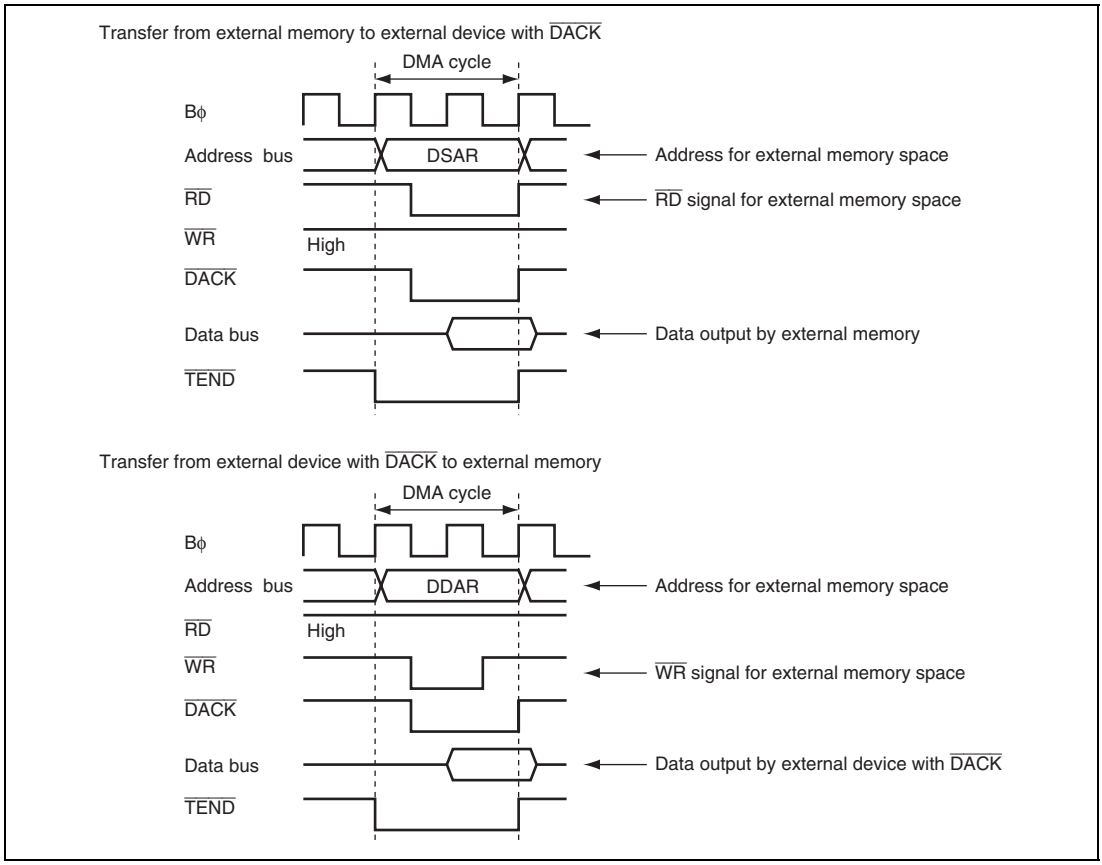


Figure 7.5 Example of Signal Timing in Single Address Mode

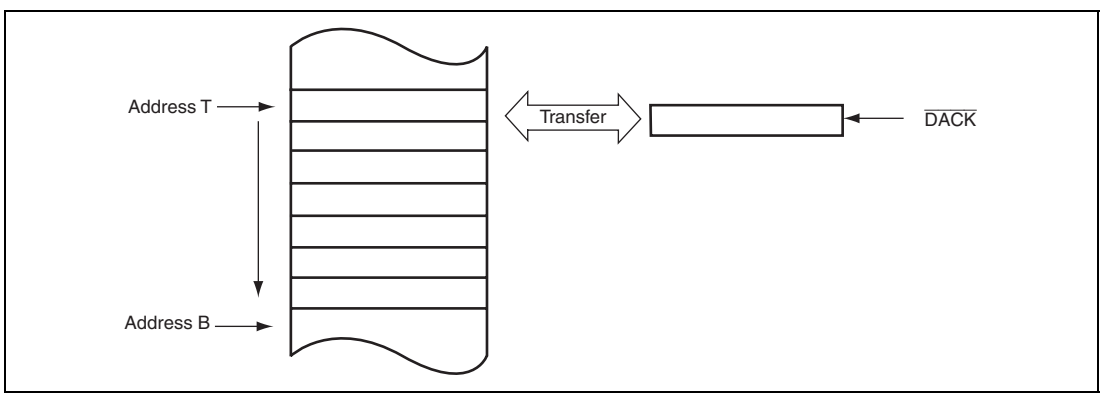


Figure 7.6 Operations in Single Address Mode

7.4.2 Transfer Modes

(1) Normal Transfer Mode

In normal transfer mode, one data access size of data is transferred at a single transfer request. Up to 4 Gbytes can be specified as a total transfer size by DTCR. DBSR is ignored in normal transfer mode.

The $\overline{\text{TEND}}$ signal is output only in the last DMA transfer. The $\overline{\text{DACK}}$ signal is output every time a transfer request is received and a transfer starts.

Figure 7.7 shows an example of the signal timing in normal transfer mode and figure 7.8 shows the operation in normal transfer mode.

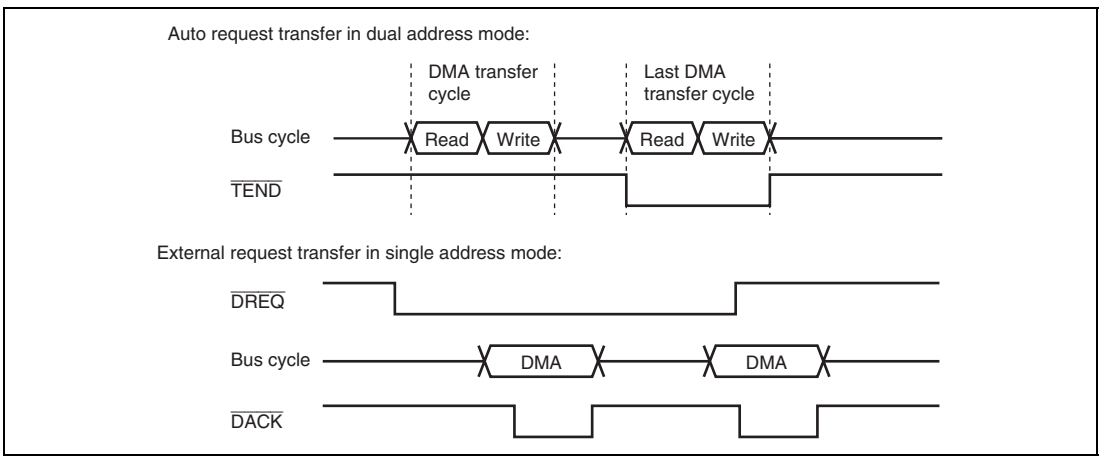


Figure 7.7 Example of Signal Timing in Normal Transfer Mode

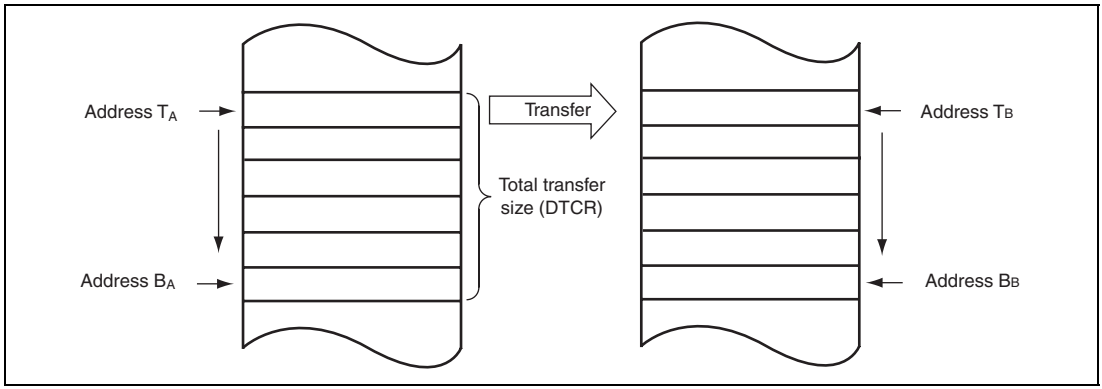


Figure 7.8 Operations in Normal Transfer Mode

(2) Repeat Transfer Mode

In repeat transfer mode, one data access size of data is transferred at a single transfer request. Up to 4 Gbytes can be specified as a total transfer size by DTCR. The repeat size can be specified in DBSR up to 64K × data access size.

The repeat area can be specified for the source or destination address side by bits ARS1 and ARS0 in DACR. The address specified as the repeat area returns to the transfer start address when the repeat size of transfers is completed. This operation is repeated until the total transfer size specified in DTCR is completed. When H'00000000 is specified in DTCR, it is regarded as the free running mode and repeat transfer is continued until the DTE bit in DMDR is cleared to 0.

In addition, a DMA transfer can be stopped and a repeat size end interrupt can be requested to the CPU or DTC when the repeat size of transfers is completed. When the next transfer is requested after completion of a 1-repeat size data transfer while the RPTIE bit is set to 1, the DTE bit in DMDR is cleared to 0 and the ESIF bit in DMDR is set to 1 to complete the transfer. At this time, an interrupt is requested to the CPU or DTC when the ESIE bit in DMDR is set to 1.

DMA transfer timing of the $\overline{\text{TEND}}$ signal output is the same as in normal transfer mode.

Figure 7.9 shows the operation in repeat transfer mode while dual address mode is set.

When the repeat area is specified as neither source nor destination address side, the operation is the same as the normal transfer mode operation shown in figure 7.8. In this case, a repeat size end interrupt can also be requested to the CPU when the repeat size of transfers is completed.

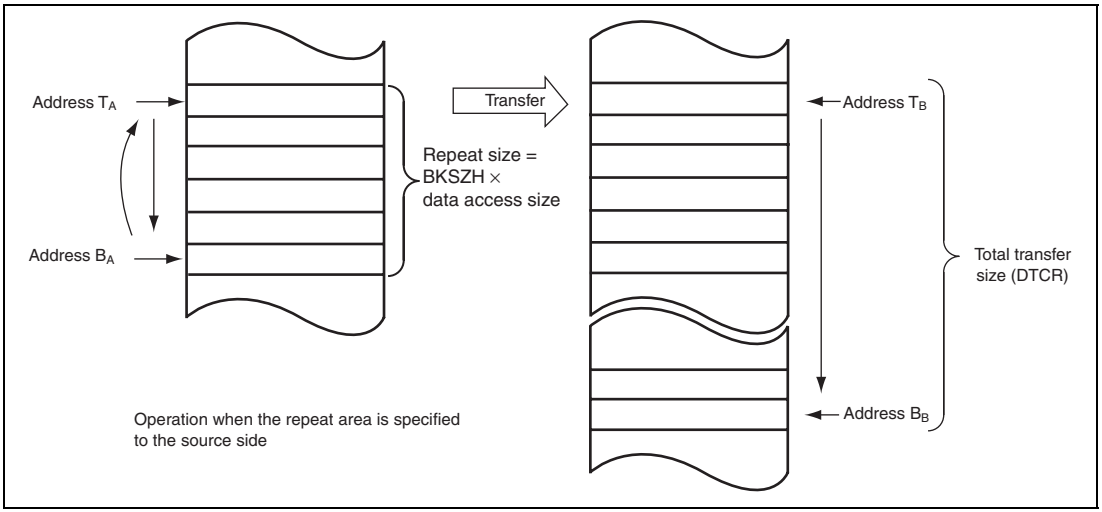


Figure 7.9 Operations in Repeat Transfer Mode

(3) Block Transfer Mode

In block transfer mode, one block size of data is transferred at a single transfer request. Up to 4 Gbytes can be specified as total transfer size by DTCR. The block size can be specified in DBSR up to 64K × data access size.

While one block of data is being transferred, transfer requests from other channels are suspended. When the transfer is completed, the bus is released to the other bus master.

The block area can be specified for the source or destination address side by bits ARS1 and ARS0 in DACR. The address specified as the block area returns to the transfer start address when the block size of data is completed. When the block area is specified as neither source nor destination address side, the operation continues without returning the address to the transfer start address. A repeat size end interrupt can be requested.

The $\overline{\text{TEND}}$ signal is output every time 1-block data is transferred in the last DMA transfer cycle. When the external request is selected as an activation source, the low level detection of the $\overline{\text{DREQ}}$ signal (DREQS = 0) should be selected.

When an interrupt request by an extended repeat area overflow is used in block transfer mode, settings should be selected carefully. For details, see section 7.4.5, Extended Repeat Area Function.

Figure 7.10 shows an example of the DMA transfer timing in block transfer mode. The transfer conditions are as follows:

- Address mode: single address mode
- Data access size: byte
- 1-block size: three bytes

The block transfer mode operations in single address mode and in dual address mode are shown in figures 7.11 and 7.12, respectively.

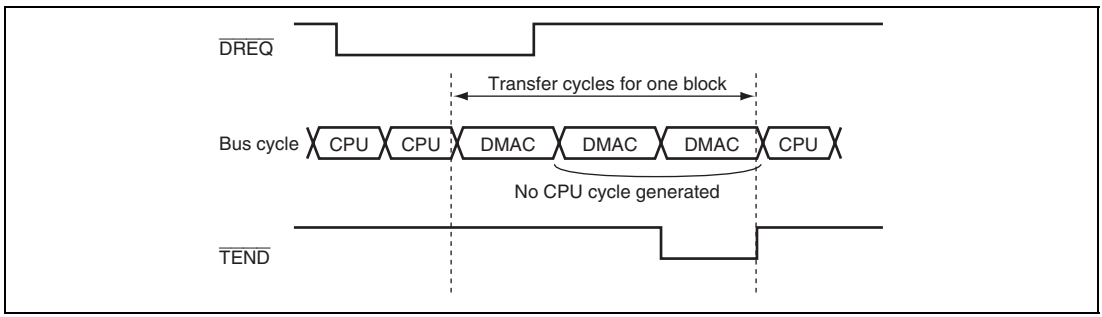


Figure 7.10 Operations in Block Transfer Mode

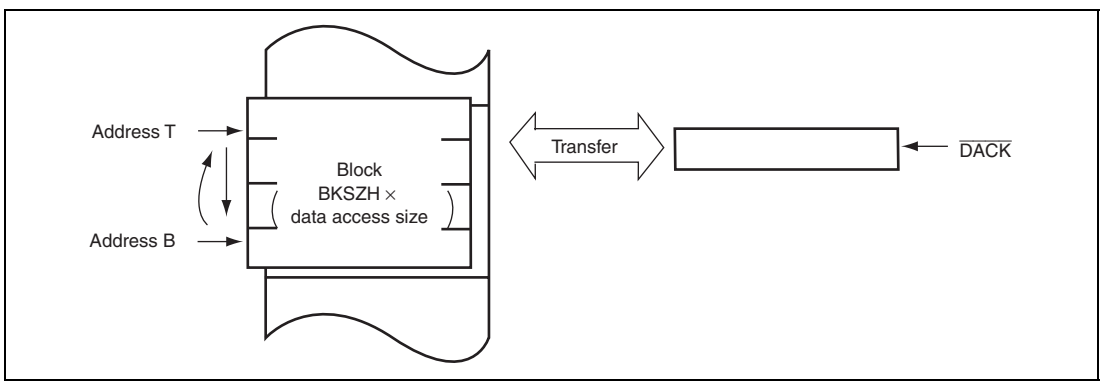


Figure 7.11 Operation in Single Address Mode in Block Transfer Mode (Block Area Specified)

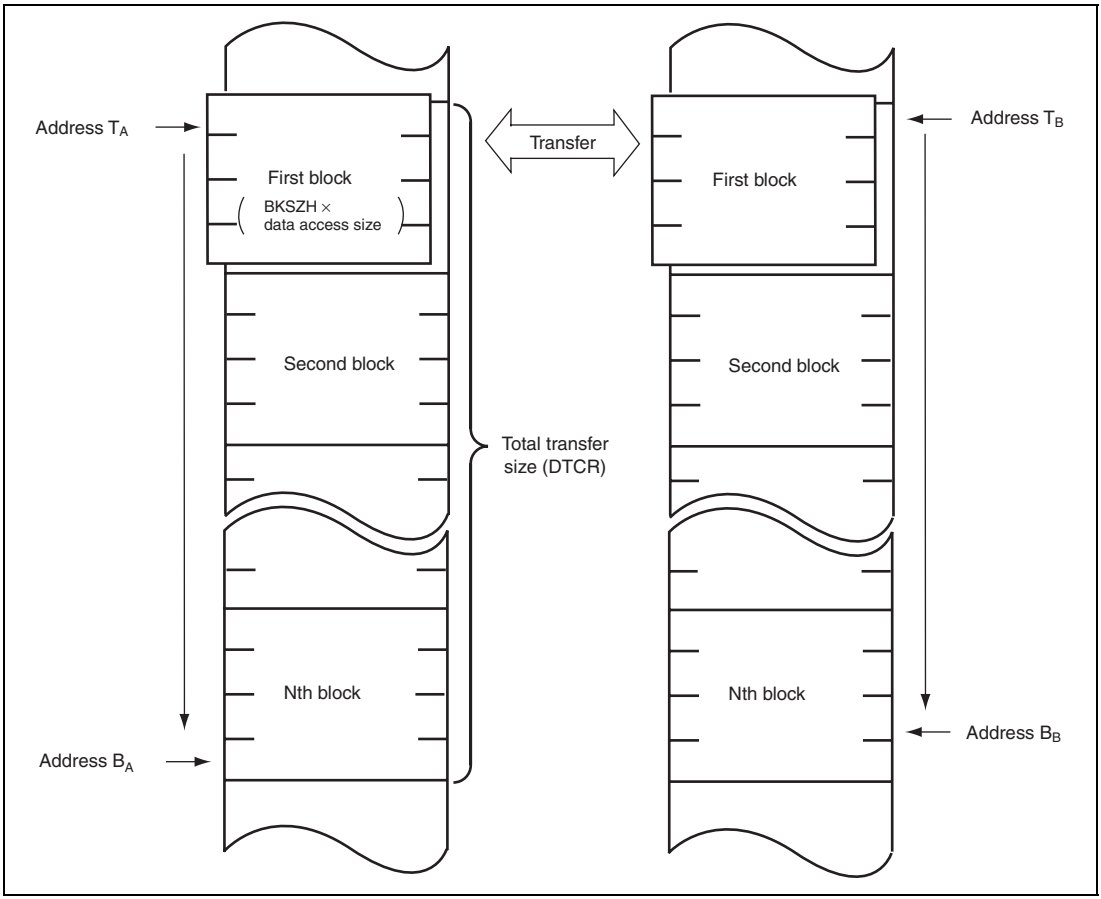


Figure 7.12 Operation in Dual Address Mode in Block Transfer Mode (Block Area Not Specified)

7.4.3 Activation Sources

The DMAC is activated by an auto request, an on-chip module interrupt, and an external request. The activation source is specified by bits DTF1 and DTF0 in DMDR.

(1) Activation by Auto Request

The auto request activation is used when a transfer request from an external device or an on-chip peripheral module is not generated such as a transfer between memory and memory or between memory and an on-chip peripheral module which does not request a transfer. A transfer request is automatically generated inside the DMAC. In auto request activation, setting the DTE bit in DMDR starts a transfer. The bus mode can be selected from cycle stealing and burst modes.

(2) Activation by On-Chip Module Interrupt

An interrupt request from an on-chip peripheral module (on-chip peripheral module interrupt) is used as a transfer request. When a DMA transfer is enabled (DTE = 1), the DMA transfer is started by an on-chip module interrupt.

The activation source of the on-chip module interrupt is selected by the DMA module request select register (DMRSR). The activation sources are specified to the individual channels. Table 7.4 is a list of on-chip module interrupts for the DMAC. The interrupt request selected as the activation source can generate an interrupt request simultaneously to the CPU or DTC. For details, refer to section 5, Interrupt Controller.

The DMAC receives interrupt requests by on-chip peripheral modules independent of the interrupt controller. Therefore, the DMAC is not affected by priority given in the interrupt controller.

When the DMAC is activated while DTA = 1, the interrupt request flag is automatically cleared by a DMA transfer. If multiple channels use a single transfer request as an activation source, when the channel having priority is activated, the interrupt request flag is cleared. In this case, other channels may not be activated because the transfer request is not held in the DMAC.

When the DMAC is activated while DTA = 0, the interrupt request flag is not cleared by the DMAC and should be cleared by the CPU or DTC transfer.

When an activation source is selected while DTE = 0, the activation source does not request a transfer to the DMAC. It requests an interrupt to the CPU or DTC.

In addition, make sure that an interrupt request flag as an on-chip module interrupt source is cleared to 0 before writing 1 to the DTE bit.

Table 7.4 List of On-Chip Module Interrupts to DMAC

On-Chip Module Interrupt Source	On-Chip Module	DMRSR (Vector Number)
ADI0 (A/D conversion end interrupt)	A/D_0	86
ADI1 (A/D conversion end interrupt)	A/D_1	87
TGI0A (TGR0A input capture/compare match)	TPU_0	88
TGI1A (TGR1A input capture/compare match)	TPU_1	93
TGI2A (TGR2A input capture/compare match)	TPU_2	97
TGI3A (TGR3A input capture/compare match)	TPU_3	101
TGI4A (TGR4A input capture/compare match)	TPU_4	106
TGI5A (TGR5A input capture/compare match)	TPU_5	110
RM0_0 (mailbox 0 message receive interrupt of RCAN-TL1 channel 0)	RCAN_0	132
RM0_1 (mailbox 1 message receive interrupt of RCAN-TL1 channel 1)	RCAN_1	134
RXI3 (receive data full interrupt of SCI channel 3)	SCI_3	157
TXI3 (transmit data empty interrupt of SCI channel 3)	SCI_3	158
RXI4 (receive data full interrupt of SCI channel 4)	SCI_4	161
TXI4 (transmit data empty interrupt of SCI channel 4)	SCI_4	162
TGI6A (TGR6A input capture/compare match)	TPU_6	164
TGI7A (TGR7A input capture/compare match)	TPU_7	169
TGI8A (TGR8A input capture/compare match)	TPU_8	173
TGI9A (TGR9A input capture/compare match)	TPU_9	177
TGI10A (TGR10A input capture/compare match)	TPU_10	182
TGI11A (TGR11A input capture/compare match)	TPU_11	188
SPRI_0 (receive data full interrupt of RSPI channel 0)	RSPI_0	197
SPTI_0 (transmit data empty interrupt of RSPI channel 0)	RSPI_0	198
SPRI_1 (receive data full interrupt of RSPI channel 1)	RSPI_1	200
SPTI_1 (transmit data empty interrupt of RSPI channel 1)	RSPI_1	201
SPRI_2 (receive data full interrupt of RSPI channel 2)	RSPI_2	203
SPTI_2 (transmit data empty interrupt of RSPI channel 2)	RSPI_2	204
SPRI_3 (receive data full interrupt of RSPI channel 3)	RSPI_3	206
SPTI_3 (transmit data empty interrupt of RSPI channel 3)	RSPI_3	207

(3) Activation by External Request

A transfer is started by a transfer request signal ($\overline{\text{DREQ}}$) from an external device. When a DMA transfer is enabled (DTE = 1), the DMA transfer is started by the $\overline{\text{DREQ}}$ assertion. When a DMA transfer between on-chip peripheral modules is performed, select an activation source from the auto request and on-chip module interrupt (the external request cannot be used).

A transfer request signal is input to the $\overline{\text{DREQ}}$ pin. The $\overline{\text{DREQ}}$ signal is detected on the falling edge or low level. Whether the falling edge or low-level detection is used is selected by the DREQS bit in DMDR.

When an external request is selected as an activation source, clear the DDR bit to 0 and set the ICR bit to 1 for the corresponding pin. For details, see section 9, I/O Ports.

7.4.4 Bus Access Modes

There are two types of bus access modes: cycle stealing and burst.

When an activation source is the auto request, the cycle stealing or burst mode is selected by bit DTF0 in DMDR. When an activation source is the on-chip module interrupt or external request, the cycle stealing mode is selected.

(1) Cycle Stealing Mode

In cycle stealing mode, the DMAC releases the bus every time one unit of transfers (byte, word, longword, or 1-block size) is completed. After that, when a transfer is requested, the DMAC obtains the bus to transfer 1-unit data and then releases the bus on completion of the transfer. This operation is continued until the transfer end condition is satisfied.

When a transfer is requested to another channel during a DMA transfer, the DMAC releases the bus and then transfers data for the requested channel. For details on operations when a transfer is requested to multiple channels, see section 7.4.8, Priority of Channels.

Figure 7.13 shows an example of timing in cycle stealing mode. The transfer conditions are as follows:

- Address mode: Single address mode
- Sampling method of the $\overline{\text{DREQ}}$ signal: Low level detection

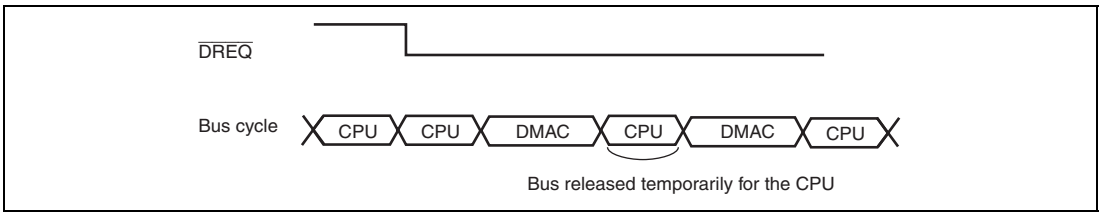


Figure 7.13 Example of Timing in Cycle Stealing Mode

(2) Burst Access Mode

In burst mode, once it takes the bus, the DMAC continues a transfer without releasing the bus until the transfer end condition is satisfied. Even if a transfer is requested from another channel having priority, the transfer is not stopped once it is started. The DMAC releases the bus in the next cycle after the transfer for the channel in burst mode is completed. This is similar to operation in cycle stealing mode. However, setting the IBCCS bit in BCR2 of the bus controller to 1 makes the DMAC release the bus to pass the bus to another bus master.

In block transfer mode, the burst mode setting is ignored (operation is the same as that in burst mode during one block of transfers). The DMAC is always operated in cycle stealing mode.

Clearing the DTE bit in DMDR to 0 stops a DMA transfer. A transfer requested before the DTE bit is cleared to 0 by the DMAC is executed. When an interrupt by a transfer size error, a repeat size end, or an extended repeat area overflow occurs, the DTE bit is cleared to 0 and the transfer ends.

Figure 7.14 shows an example of timing in burst mode.

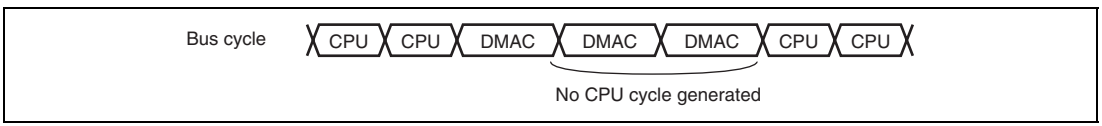


Figure 7.14 Example of Timing in Burst Mode

7.4.5 Extended Repeat Area Function

The source and destination address sides can be specified as the extended repeat area. The contents of the address register repeat addresses within the area specified as the extended repeat area. For example, to use a ring buffer as the transfer target, the contents of the address register should return to the start address of the buffer every time the contents reach the end address of the buffer (overflow on the ring buffer address). This operation can automatically be performed using the extended repeat area function of the DMAC.

The extended repeat areas can be specified independently to the source address register (DSAR) and destination address register (DDAR).

The extended repeat area on the source address is specified by bits SARA4 to SARA0 in DACR. The extended repeat area on the destination address is specified by bits DARA4 to DARA0 in DACR. The extended repeat area sizes for each side can be specified independently.

A DMA transfer is stopped and an interrupt by an extended repeat area overflow can be requested to the CPU when the contents of the address register reach the end address of the extended repeat area. When an overflow on the extended repeat area set in DSAR occurs while the SARIE bit in DACR is set to 1, the ESIF bit in DMDR is set to 1 and the DTE bit in DMDR is cleared to 0 to stop the transfer. At this time, if the ESIE bit in DMDR is set to 1, an interrupt by an extended repeat area overflow is requested to the CPU. When the DARIE bit in DACR is set to 1, an overflow on the extended repeat area set in DDAR occurs, meaning that the destination side is a target. During the interrupt handling, setting the DTE bit in DMDR to 1 resumes the transfer.

Figure 7.15 shows an example of the extended repeat area operation.

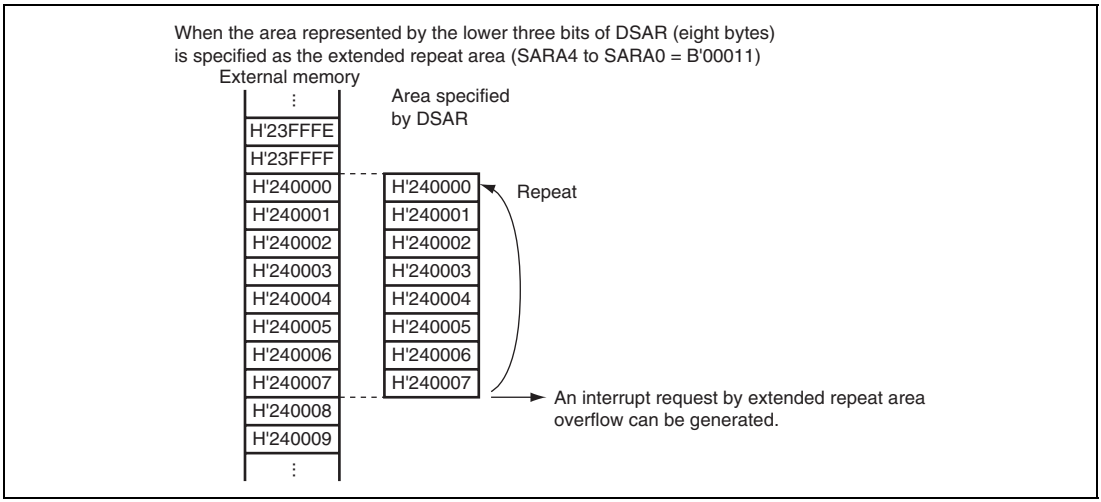


Figure 7.15 Example of Extended Repeat Area Operation

When an interrupt by an extended repeat area overflow is used in block transfer mode, the following should be taken into consideration.

When a transfer is stopped by an interrupt by an extended repeat area overflow, the address register must be set so that the block size is a power of 2 or the block size boundary is aligned with the extended repeat area boundary. When an overflow on the extended repeat area occurs during a transfer of one block, the interrupt by the overflow is suspended and the transfer overruns.

Figure 7.16 shows examples when the extended repeat area function is used in block transfer mode.

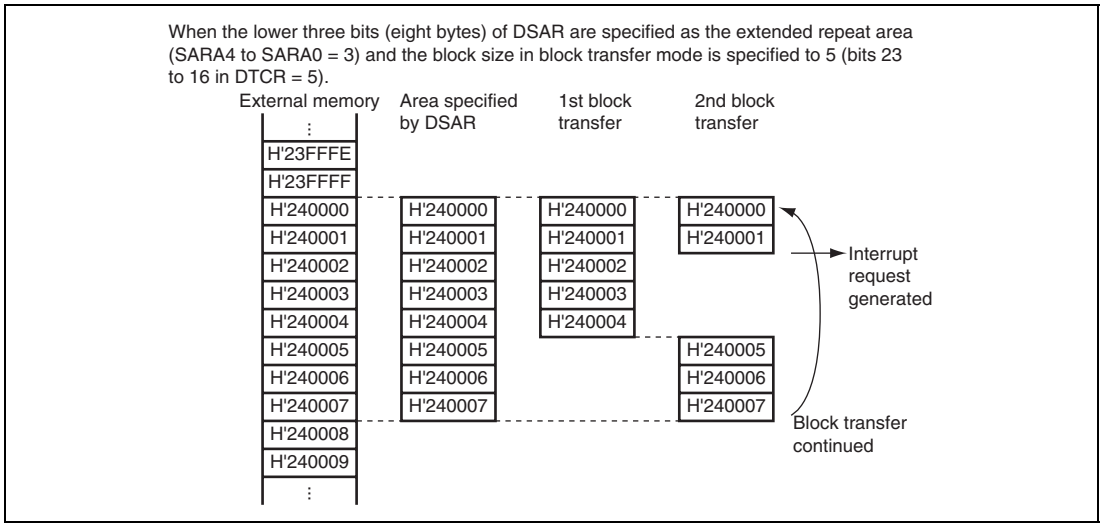


Figure 7.16 Example of Extended Repeat Area Function in Block Transfer Mode

7.4.6 Address Update Function Using Offset

The source and destination addresses are updated by fixing, increment/decrement by 1, 2, or 4, or offset addition. When the offset addition is selected, the offset specified by the offset register (DOFR) is added to the address every time the DMAC transfers the data access size of data. This function realizes a data transfer where addresses are allocated to separated areas.

Figure 7.17 shows the address update method.

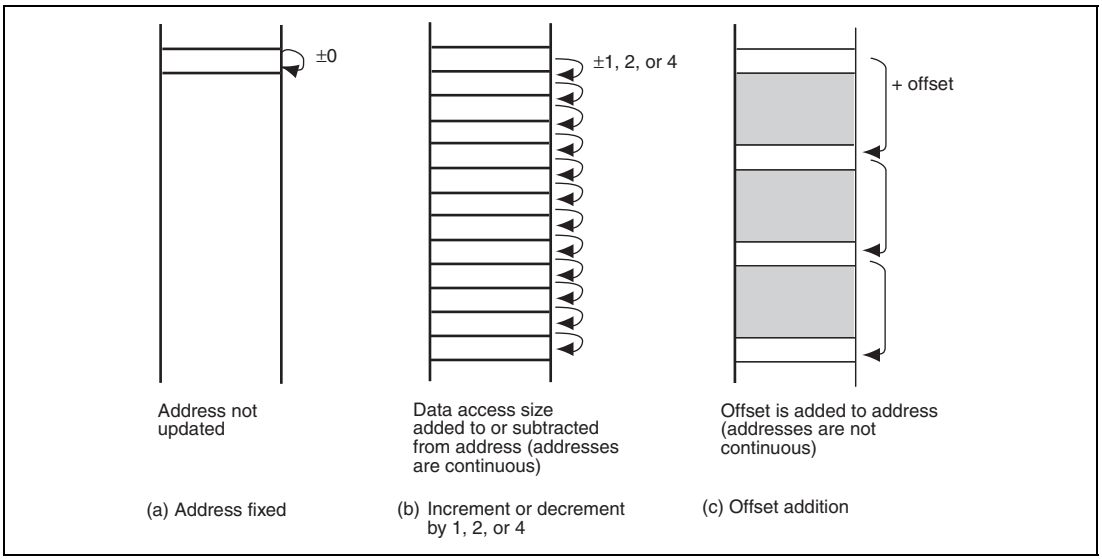


Figure 7.17 Address Update Method

In item (a), Address fixed, the transfer source or destination address is not updated indicating the same address.

In item (b), Increment or decrement by 1, 2, or 4, the transfer source or destination address is incremented or decremented by the value according to the data access size at each transfer. Byte, word, or longword can be specified as the data access size. The value of 1 for byte, 2 for word, and 4 for longword is used for updating the address. This operation realizes the data transfer placed in consecutive areas.

In item (c), Offset addition, the address update does not depend on the data access size. The offset specified by DOFR is added to the address every time the DMAC transfers data of the data access size.

The address is calculated by the offset set in DOFR and the contents of DSAR and DDAR. Although the DMAC calculates only addition, an offset subtraction can be realized by setting the negative value in DOFR. In this case, the negative value must be 2's complement.

(1) Basic Transfer Using Offset

Figure 7.18 shows a basic operation of a transfer using the offset addition.

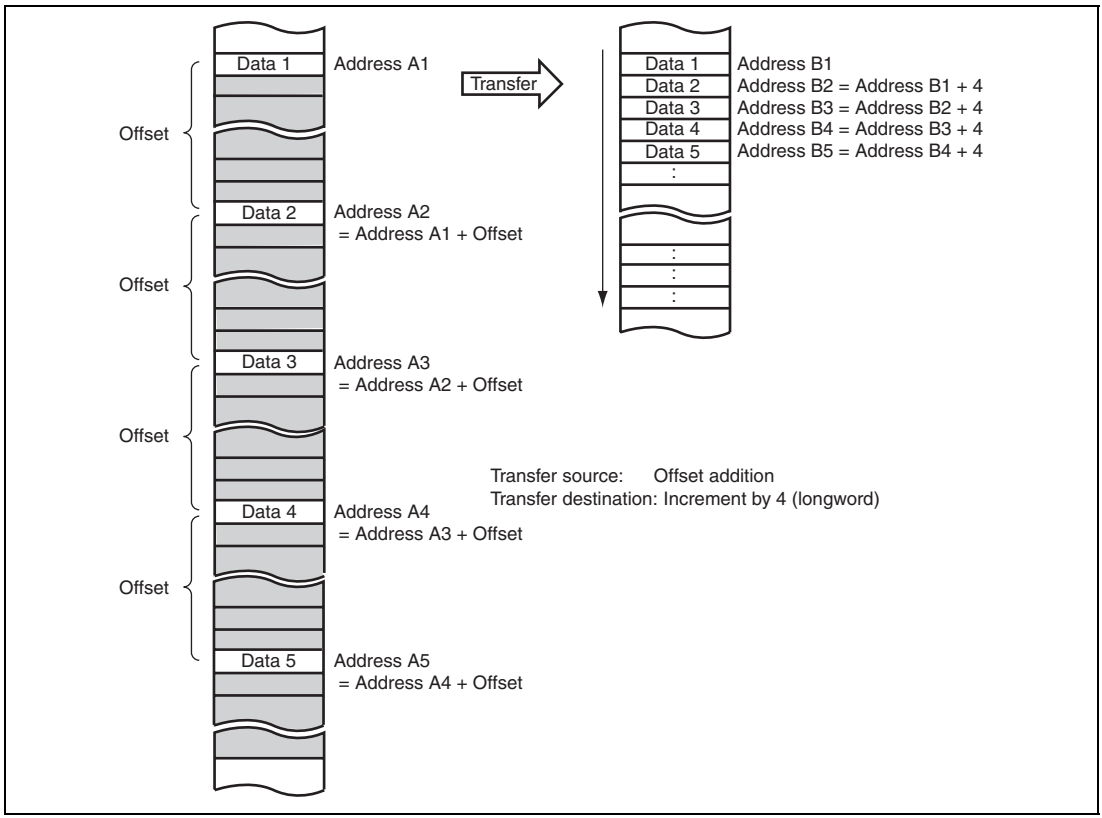


Figure 7.18 Operation of Offset Addition

In figure 7.18, the offset addition is selected as the transfer source address update and increment or decrement by 1, 2, or 4 is selected as the transfer destination address. The address update means that data at the address which is away from the previous transfer source address by the offset is read from. The data read from the address away from the previous address is written to the consecutive area in the destination side.

(2) XY Conversion Using Offset

Figure 7.19 shows the XY conversion using the offset addition in repeat transfer mode.

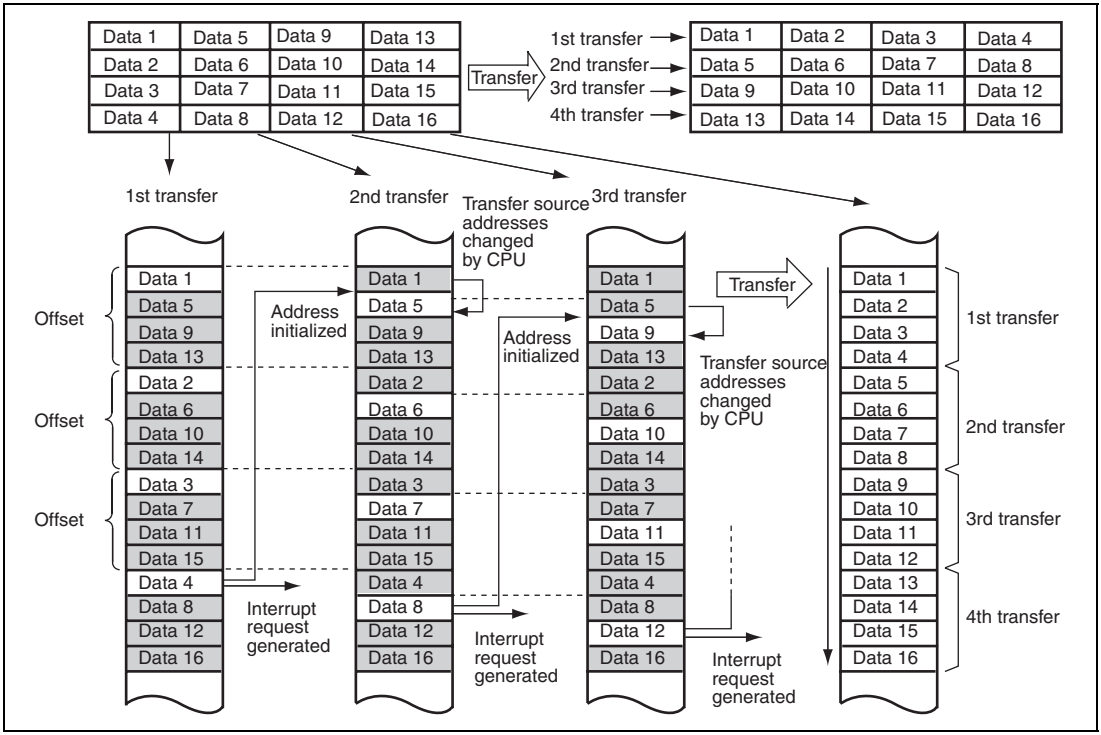


Figure 7.19 XY Conversion Operation Using Offset Addition in Repeat Transfer Mode

In figure 7.19, the source address side is specified to the repeat area by DACR and the offset addition is selected. The offset value is set to 4 × data access size (when the data access size is longword, H'00000010 is set in DOFR, as an example). The repeat size is set to 4 × data access size (when the data access size is longword, the repeat size is set to 4 × 4 = 16 bytes, as an example). The increment or decrement by 1, 2, or 4 is specified as the transfer destination address. A repeat size end interrupt is requested when the repeat size of transfers is completed.

When a transfer starts, the transfer source address is added to the offset every time data is transferred. The transfer data is written to the destination continuous addresses. When data 4 is transferred meaning that the repeat size of transfers is completed, the transfer source address returns to the transfer start address (address of data 1 on the transfer source) and a repeat size end interrupt is requested. While this interrupt stops the transfer temporarily, the contents of DSAR are written to the address of data 5 by the CPU (when the data access size is longword, write the data 1 address + 4). When the DTE bit in DMDR is set to 1, the transfer is resumed from the state when the transfer is stopped. Accordingly, operations are repeated and the transfer source data is transposed to the destination area (XY conversion).

Figure 7.20 shows a flowchart of the XY conversion.

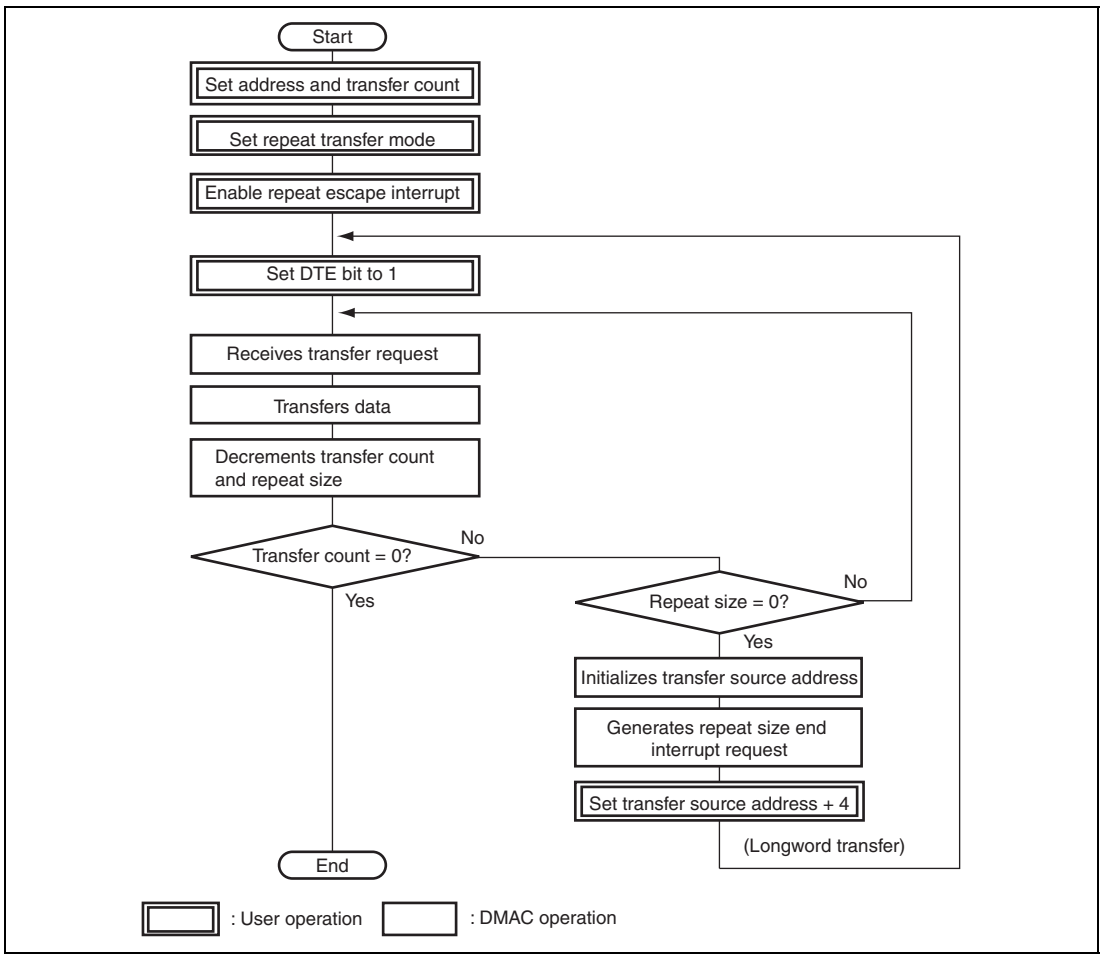


Figure 7.20 XY Conversion Flowchart Using Offset Addition in Repeat Transfer Mode

(3) Offset Subtraction

When setting the negative value in DOFR, the offset value must be 2's complement. The 2's complement is obtained by the following formula.

2's complement of offset = 1 + ~offset (~: bit inversion)

Example: 2's complement of H'0001FFFF
= H'FFFE0000 + H'00000001
= H'FFFE0001

The value of 2's complement can be obtained by the NEG.L instruction.

7.4.7 Register during DMA Transfer

The DMAC registers are updated by a DMA transfer. The value to be updated differs according to the other settings and transfer state. The registers to be updated are bits BKSZH and BKSZ in DSAR, DDAR, DDCR and DBSR, and the DTE, ACT, ERRF, ESIF, and DTIF bits in DMDR.

(1) DMA Source Address Register (DSAR)

When the transfer source address set in DSAR is accessed, the contents of DSAR are output and then are updated to the next address.

The increment or decrement can be specified by bits SAT1 and SAT0 in DACR. When SAT1 and SAT0 = B'00, the address is fixed. When SAT1 and SAT0 = B'01, the address is added with the offset. When SAT1 and SAT0 = B'10, the address is incremented. When SAT1 and SAT0 = B'11, the address is decremented. The size of increment or decrement depends on the data access size.

The data access size is specified by bits DTSZ1 and DTSZ0 in DMDR. When DTSZ1 and DTSZ0 = B'00, the data access size is byte and the address is incremented or decremented by 1. When DTSZ1 and DTSZ0 = B'01, the data access size is word and the address is incremented or decremented by 2. When DTSZ1 and DTSZ0 = B'10, the data access size is longword and the address is incremented or decremented by 4. Even if the access data size of the source address is word or longword, when the source address is not aligned with the word or longword boundary, the read bus cycle is divided into byte or word cycles. While data of one word or one longword is being read, the size of increment or decrement is changing according to the actual data access size, for example, +1 or +2 for byte or word data. After one word or one longword of data is read, the address when the read cycle is started is incremented or decremented by the value according to bits SAT1 and SAT0.

In block or repeat transfer mode, when the block or repeat size of data transfers is completed while the block or repeat area is specified to the source address side, the source address returns to the transfer start address and is not affected by the address update.

When the extended repeat area is specified to the source address side, operation follows the setting. The upper address bits are fixed and is not affected by the address update.

While data is being transferred, DSAR must be accessed in longwords. If the upper word and lower word are read separately, incorrect data may be read from since the contents of DSAR during the transfer may be updated regardless of the access by the CPU. Moreover, DSAR for the channel being transferred must not be written to.

(2) DMA Destination Address Register (DDAR)

When the transfer destination address set in DDAR is accessed, the contents of DDAR are output and then are updated to the next address.

The increment or decrement can be specified by bits DAT1 and DAT0 in DACR. When DAT1 and DAT0 = B'00, the address is fixed. When DAT1 and DAT0 = B'01, the address is added with the offset. When DAT1 and DAT0 = B'10, the address is incremented. When DAT1 and DAT0 = B'11, the address is decremented. The incrementing or decrementing size depends on the data access size.

The data access size is specified by bits DTSZ1 and DTSZ0 in DMDR. When DTSZ1 and DTSZ0 = B'00, the data access size is byte and the address is incremented or decremented by 1. When DTSZ1 and DTSZ0 = B'01, the data access size is word and the address is incremented or decremented by 2. When DTSZ1 and DTSZ0 = B'10, the data access size is longword and the address is incremented or decremented by 4. Even if the access data size of the destination address is word or longword, when the destination address is not aligned with the word or longword boundary, the write bus cycle is divided into byte and word cycles. While one word or one longword of data is being written, the incrementing or decrementing size is changing according to the actual data access size, for example, +1 or +2 for byte or word data. After the one word or one longword of data is written, the address when the write cycle is started is incremented or decremented by the value according to bits SAT1 and SAT0.

In block or repeat transfer mode, when the block or repeat size of data transfers is completed while the block or repeat area is specified to the destination address side, the destination address returns to the transfer start address and is not affected by the address update.

When the extended repeat area is specified to the destination address side, operation follows the setting. The upper address bits are fixed and is not affected by the address update.

While data is being transferred, DDAR must be accessed in longwords. If the upper word and lower word are read separately, incorrect data may be read from since the contents of DDAR during the transfer may be updated regardless of the access by the CPU. Moreover, DDAR for the channel being transferred must not be written to.

(3) DMA Transfer Count Register (DTCR)

A DMA transfer decrements the contents of DTCR by the transferred bytes. When byte data is transferred, DTCR is decremented by 1. When word data is transferred, DTCR is decremented by 2. When longword data is transferred, DTCR is decremented by 4. However, when DTCR = 0, the contents of DTCR are not changed since the number of transfers is not counted.

While data is being transferred, all the bits of DTCR may be changed. DTCR must be accessed in longwords. If the upper word and lower word are read separately, incorrect data may be read from since the contents of DTCR during the transfer may be updated regardless of the access by the CPU. Moreover, DTCR for the channel being transferred must not be written to.

When a conflict occurs between the address update by DMA transfer and write access by the CPU, the CPU has priority. When a conflict occurs between change from 1, 2, or 4 to 0 in DTCR and write access by the CPU (other than 0), the CPU has priority in writing to DTCR. However, the transfer is stopped.

(4) DMA Block Size Register (DBSR)

DBSR is enabled in block or repeat transfer mode. Bits 31 to 16 in DBSR function as BKSZH and bits 15 to 0 in DBSR function as BKSZ. The BKSZH bits (16 bits) store the block size and repeat size and its value is not changed. The BKSZ bits (16 bits) function as a counter for the block size and repeat size and its value is decremented by 1 in every transfer. When the BKSZ value is to change from 1 to 0 by a DMA transfer, 0 is not stored but the BKSZH value is loaded into the BKSZ bits.

Since the upper 16 bits of DBSR are not updated, DBSR can be accessed in words.

DBSR for the channel being transferred must not be written to.

(5) DTE Bit in DMDR

Although the DTE bit in DMDR enables or disables data transfer by the CPU write access, it is automatically cleared to 0 according to the DMA transfer state by the DMAC.

The conditions for clearing the DTE bit by the DMAC are as follows:

- When the total size of transfers is completed
- When a transfer is completed by a transfer size error interrupt
- When a transfer is completed by a repeat size end interrupt
- When a transfer is completed by an extended repeat area overflow interrupt
- When a transfer is stopped by an NMI interrupt
- When a transfer is stopped by an address error
- Reset state
- Hardware standby mode*
- When a transfer is stopped by writing 0 to the DTE bit

Writing to the registers for the channels when the corresponding DTE bit is set to 1 is prohibited (except for the DTE bit). When changing the register settings after writing 0 to the DTE bit, confirm that the DTE bit has been cleared to 0.

Note: * This LSI does not have hardware standby mode.

Figure 7.21 show the procedure for changing the register settings for the channel being transferred.

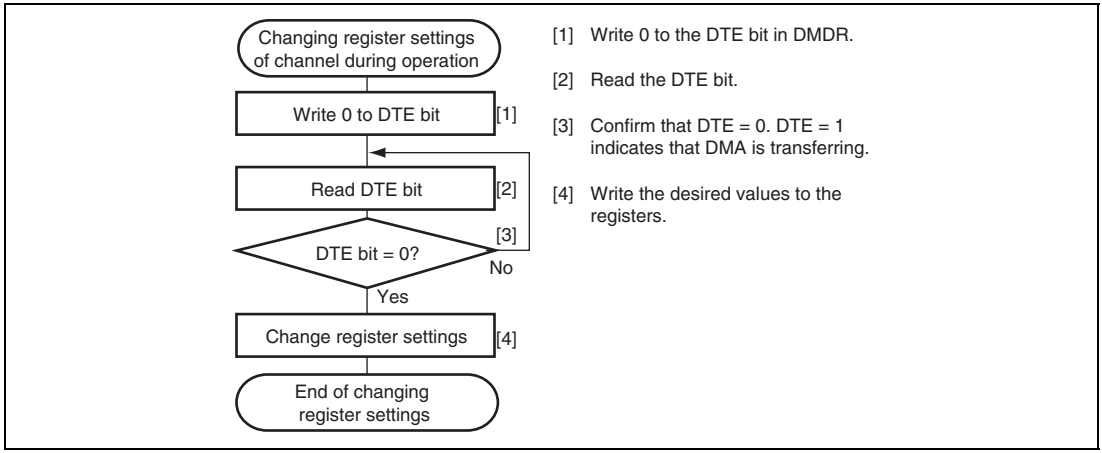


Figure 7.21 Procedure for Changing Register Setting For Channel being Transferred

(6) ACT Bit in DMDR

The ACT bit in DMDR indicates whether the DMAC is in the idle or active state. When DTE = 0 or DTE = 1 and the DMAC is waiting for a transfer request, the ACT bit is 0. Otherwise (the DMAC is in the active state), the ACT bit is 1. When individual transfers are stopped by writing 0 and the transfer is not completed, the ACT bit retains 1.

In block transfer mode, even if individual transfers are stopped by writing 0 to the DTE bit, the 1-block size of transfers is not stopped. The ACT bit retains 1 from writing 0 to the DTE bit to completion of a 1-block size transfer.

In burst mode, up to three times of DMA transfer are performed from the cycle in which the DTE bit is written to 0. The ACT bit retains 1 from writing 0 to the DTE bit to completion of DMA transfer.

(7) ERRF Bit in DMDR

When an address error or an NMI interrupt occurs, the DMAC clears the DTE bits for all the channels to stop a transfer. In addition, it sets the ERRF bit in DMDR_0 to 1 to indicate that an address error or an NMI interrupt has occurred regardless of whether or not the DMAC is in operation.

(8) ESIF Bit in DMDR

When an interrupt by an transfer size error, a repeat size end, or an extended repeat area overflow is requested, the ESIF bit in DMDR is set to 1. When both the ESIF and ESIE bits are set to 1, a transfer escape interrupt is requested to the CPU or DTC.

The ESIF bit is set to 1 when the ACT bit in DMDR is cleared to 0 to stop a transfer after the bus cycle of the interrupt source is completed.

The ESIF bit is automatically cleared to 0 and a transfer request is cleared if the transfer is resumed by setting the DTE bit to 1 during interrupt handling.

For details on interrupts, see section 7.7, Interrupt Sources.

(9) DTIF Bit in DMDR

The DTIF bit in DMDR is set to 1 after the total transfer size of transfers is completed. When both the DTIF and DTIE bits in DMDR are set to 1, a transfer end interrupt by the transfer counter is requested to the CPU or DTC.

The DTIF bit is set to 1 when the ACT bit in DMDR is cleared to 0 to stop a transfer after the bus cycle is completed.

The DTIF bit is automatically cleared to 0 and a transfer request is cleared if the transfer is resumed by setting the DTE bit to 1 during interrupt handling.

For details on interrupts, see section 7.7, Interrupt Sources.

7.4.8 Priority of Channels

The channels of the DMAC are given following priority levels: channel 0 > channel 1 > channel 2 > channel 3. Table 7.5 shows the priority levels among the DMAC channels.

Table 7.5 Priority between DMAC Channels

Channel	Priority
Channel 0	High
Channel 1	↑
Channel 2	
Channel 3	Low

The channel having highest priority other than the channel being transferred is selected when a transfer is requested from other channels. The selected channel starts the transfer after the channel being transferred releases the bus. At this time, when a bus master other than the DMAC requests the bus, the cycle for the bus master is inserted.

In a burst transfer or a block transfer, channels are not switched.

Figure 7.22 shows a transfer example when multiple transfer requests from channels 0 to 2.

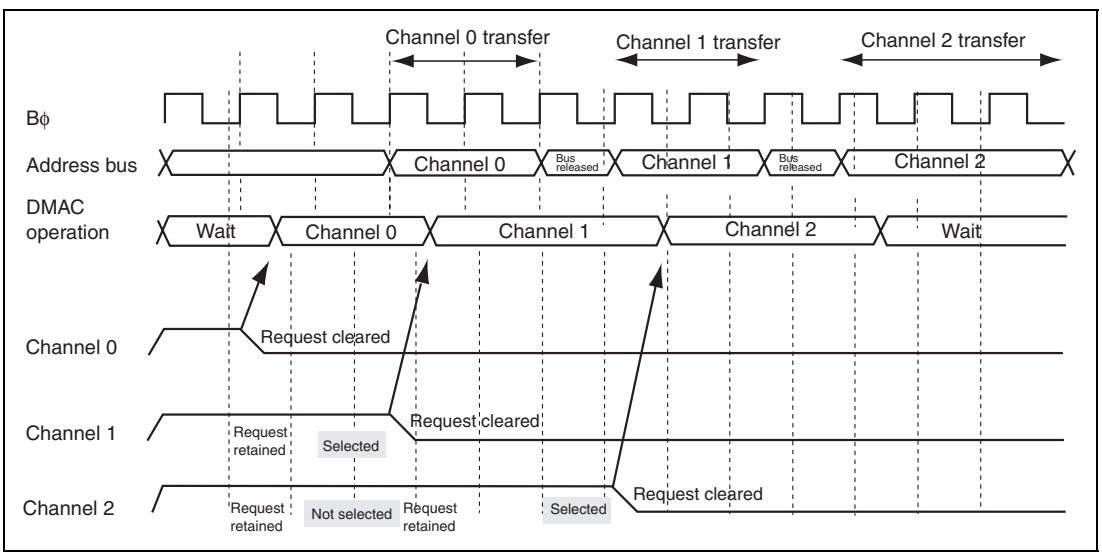


Figure 7.22 Example of Timing for Channel Priority

7.4.9 DMA Basic Bus Cycle

Figure 7.23 shows an example of signal timing of a basic bus cycle. In figure 7.23, data is transferred in words from the 16-bit 2-state access space to the 8-bit 3-state access space. When the bus mastership is passed from the CPU to the DMAC, data is read from the source address and it is written to the destination address. The bus is not released between the read and write cycles by other bus requests. DMAC bus cycles follows the bus controller settings.

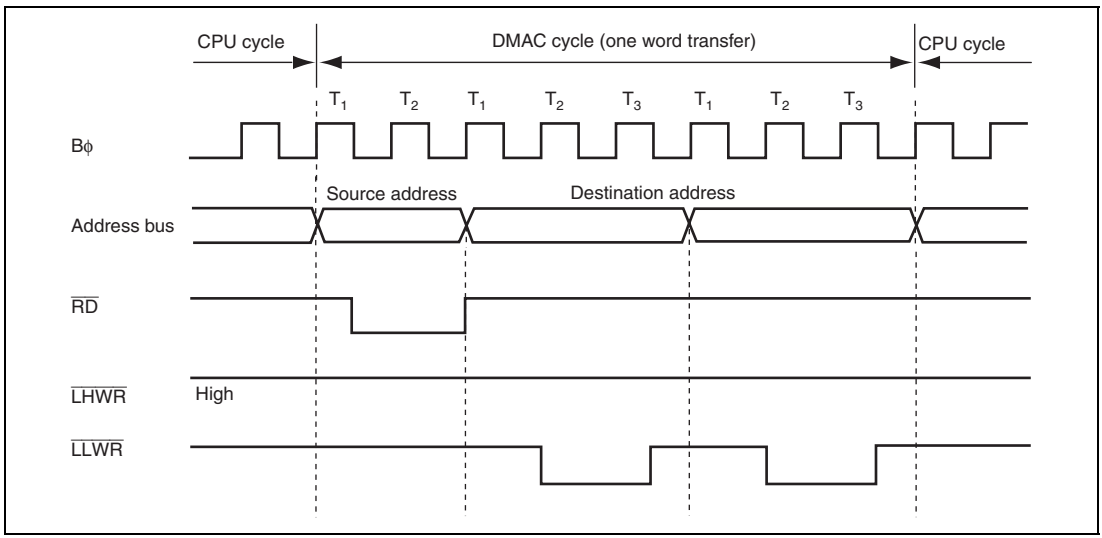


Figure 7.23 Example of Bus Timing of DMA Transfer

7.4.10 Bus Cycles in Dual Address Mode

(1) Normal Transfer Mode (Cycle Stealing Mode)

In cycle stealing mode, the bus is released every time one transfer size of data (one byte, one word, or one longword) is completed. One bus cycle or more by the CPU or DTC are executed in the bus released cycles.

In figure 7.24, the $\overline{\text{TEND}}$ signal output is enabled and data is transferred in words from the external 16-bit 2-state access space to the external 16-bit 2-state access space in normal transfer mode by cycle stealing.

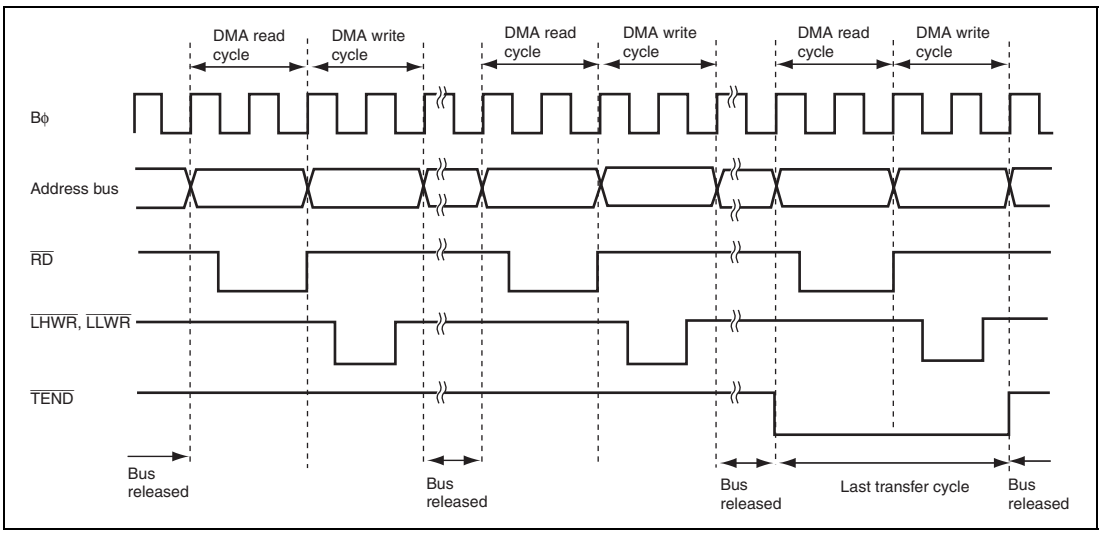


Figure 7.24 Example of Transfer in Normal Transfer Mode by Cycle Stealing

In figures 7.25 and 7.26, the $\overline{\text{TEND}}$ signal output is enabled and data is transferred in longwords from the external 16-bit 2-state access space to the 16-bit 2-state access space in normal transfer mode by cycle stealing.

In figure 7.25, the transfer source (DSAR) is not aligned with a longword boundary and the transfer destination (DDAR) is aligned with a longword boundary.

In figure 7.26, the transfer source (DSAR) is aligned with a longword boundary and the transfer destination (DDAR) is not aligned with a longword boundary.

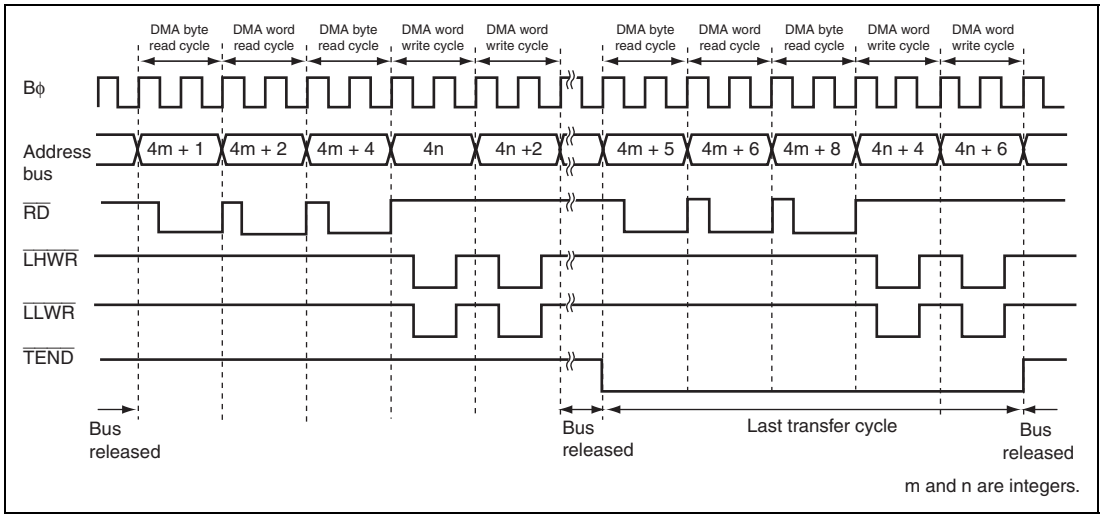


Figure 7.25 Example of Transfer in Normal Transfer Mode by Cycle Stealing (Transfer Source DSAR = Odd Address and Source Address Increment)

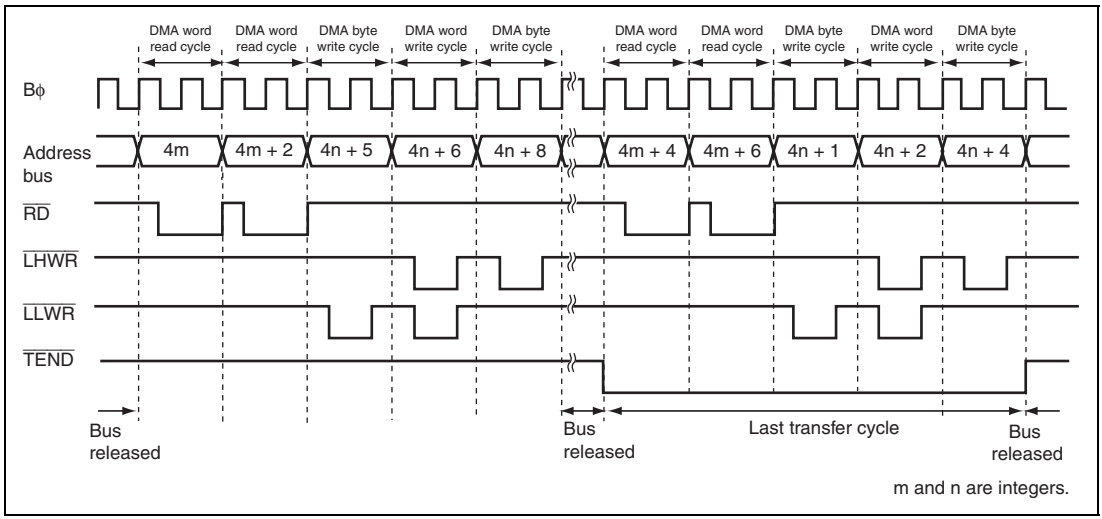


Figure 7.26 Example of Transfer in Normal Transfer Mode by Cycle Stealing (Transfer Destination DDAR = Odd Address and Destination Address Decrement)

(2) Normal Transfer Mode (Burst Mode)

In burst mode, one byte, one word, or one longword of data continues to be transferred until the transfer end condition is satisfied.

When a burst transfer starts, a transfer request from a channel having priority is suspended until the burst transfer is completed.

In figure 7.27, the $\overline{\text{TEND}}$ signal output is enabled and data is transferred in words from the external 16-bit 2-state access space to the external 16-bit 2-state access space in normal transfer mode by burst access.

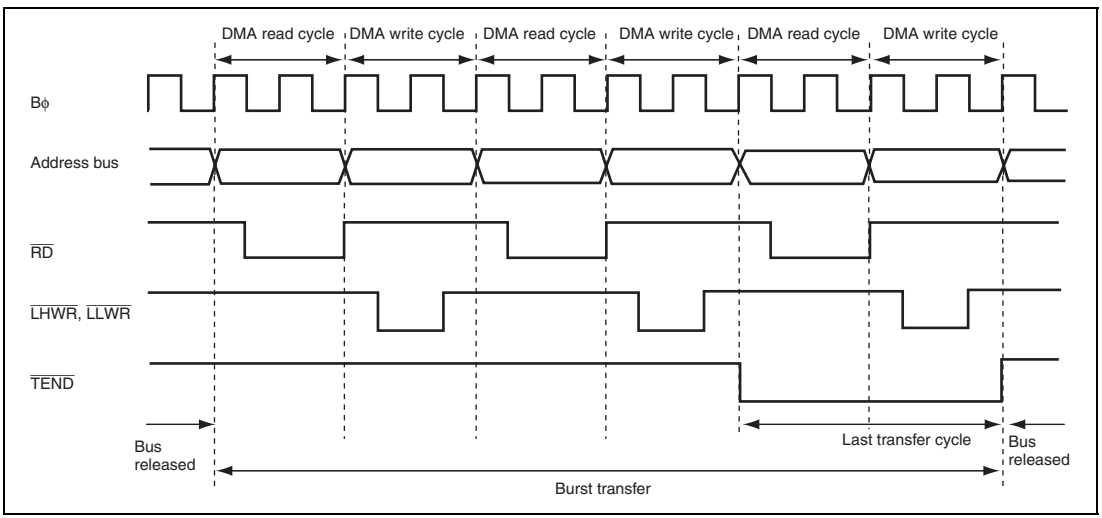


Figure 7.27 Example of Transfer in Normal Transfer Mode by Burst Access

(3) Block Transfer Mode

In block transfer mode, the bus is released every time a 1-block size of transfers at a single transfer request is completed.

In figure 7.28, the $\overline{\text{TEND}}$ signal output is enabled and data is transferred in words from the external 16-bit 2-state access space to the external 16-bit 2-state access space in block transfer mode.

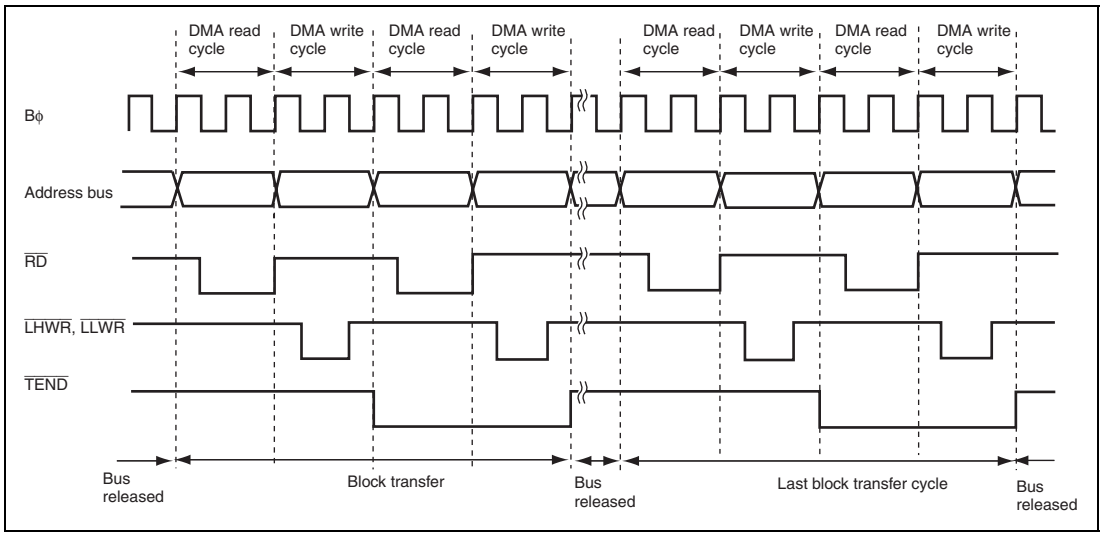


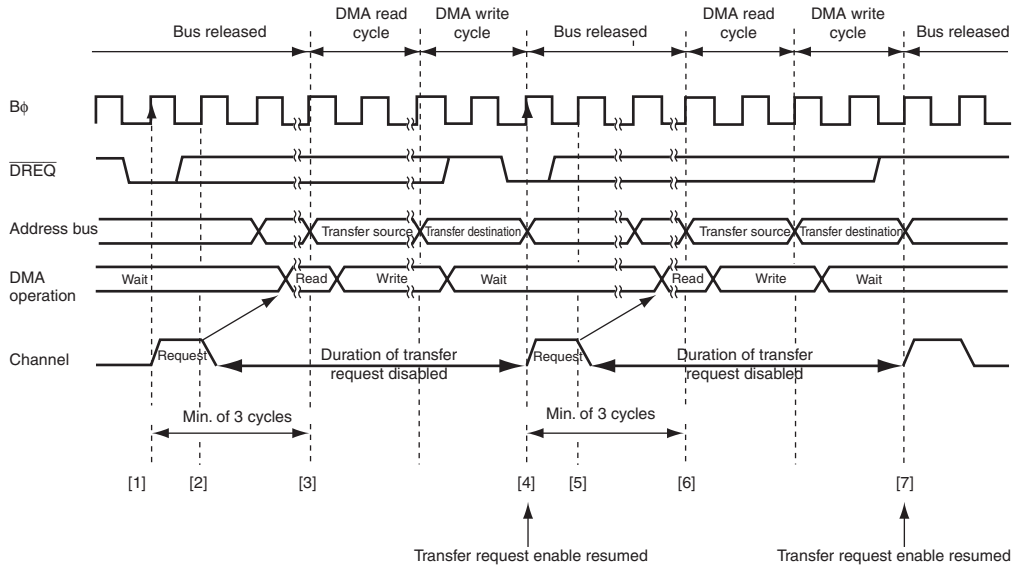
Figure 7.28 Example of Transfer in Block Transfer Mode

(4) Activation Timing by $\overline{\text{DREQ}}$ Falling Edge

Figure 7.29 shows an example of normal transfer mode activated by the $\overline{\text{DREQ}}$ signal falling edge.

The $\overline{\text{DREQ}}$ signal is sampled every cycle from the next rising edge of the Bϕ signal immediately after the DTE bit write cycle.

When a low level of the $\overline{\text{DREQ}}$ signal is detected while a transfer request by the $\overline{\text{DREQ}}$ signal is enabled, a transfer request is held in the DMAC. When the DMAC is activated, the transfer request is cleared and starts detecting a high level of the $\overline{\text{DREQ}}$ signal for falling edge detection. If a high level of the $\overline{\text{DREQ}}$ signal has been detected until completion of the DMA write cycle, receiving the next transfer request resumes and then a low level of the $\overline{\text{DREQ}}$ signal is detected. This operation is repeated until the transfer is completed.



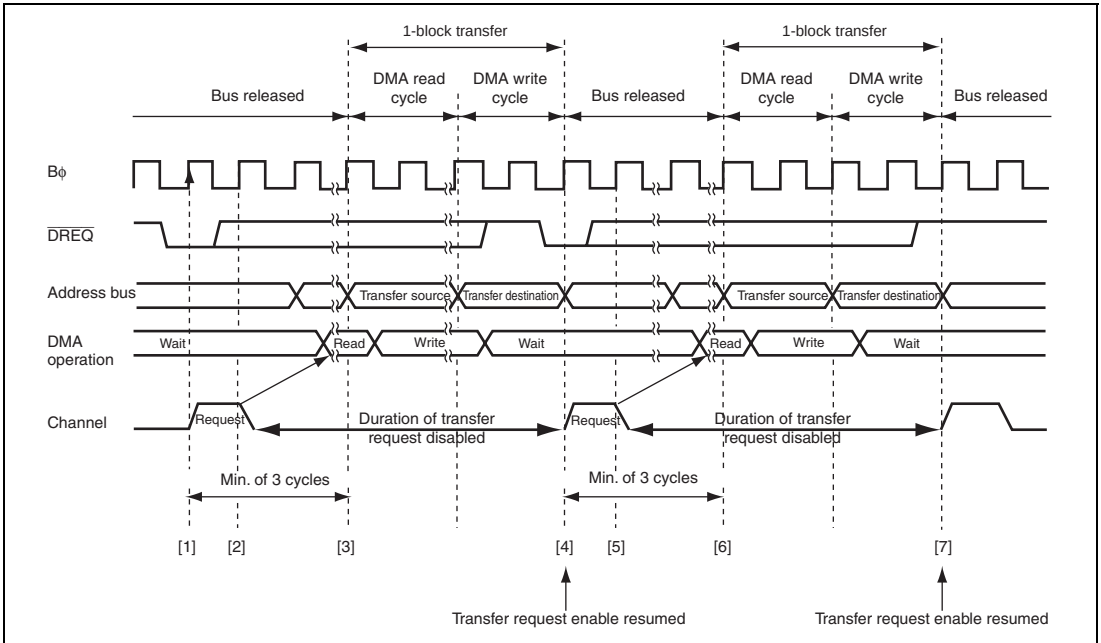
[1] After DMA transfer request is enabled, a low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held.
[2][5] The DMAC is activated and the transfer request is cleared.
[3][6] A DMA cycle is started and sampling the $\overline{\text{DREQ}}$ signal at the rising edge of the Bϕ signal is started to detect a high level of the $\overline{\text{DREQ}}$ signal.
[4][7] When a high level of the $\overline{\text{DREQ}}$ signal has been detected, transfer request enable is resumed after completion of the write cycle. (A low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held. This is the same as [1].)

Figure 7.29 Example of Transfer in Normal Transfer Mode Activated by $\overline{\text{DREQ}}$ Falling Edge

Figure 7.30 shows an example of block transfer mode activated by the $\overline{\text{DREQ}}$ signal falling edge.

The $\overline{\text{DREQ}}$ signal is sampled every cycle from the next rising edge of the Bϕ signal immediately after the DTE bit write cycle.

When a low level of the $\overline{\text{DREQ}}$ signal is detected while a transfer request by the $\overline{\text{DREQ}}$ signal is enabled, a transfer request is held in the DMAC. When the DMAC is activated, the transfer request is cleared and starts detecting a high level of the $\overline{\text{DREQ}}$ signal for falling edge detection. If a high level of the $\overline{\text{DREQ}}$ signal has been detected until completion of the DMA write cycle, receiving the next transfer request resumes and then a low level of the $\overline{\text{DREQ}}$ signal is detected. This operation is repeated until the transfer is completed.



[1] After DMA transfer request is enabled, a low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held.
[2][5] The DMAC is activated and the transfer request is cleared.
[3][6] A DMA cycle is started and sampling the $\overline{\text{DREQ}}$ signal at the rising edge of the Bϕ signal is started to detect a high level of the $\overline{\text{DREQ}}$ signal.
[4][7] When a high level of the $\overline{\text{DREQ}}$ signal has been detected, transfer request enable is resumed after completion of the write cycle. (A low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held. This is the same as [1].)

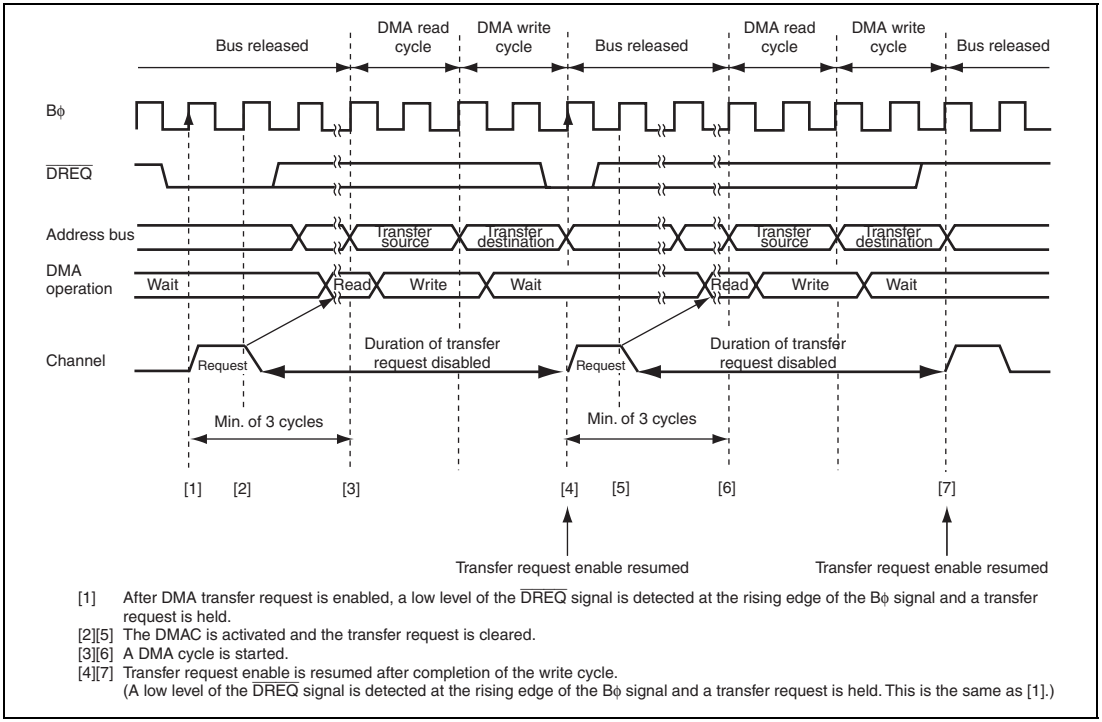
Figure 7.30 Example of Transfer in Block Transfer Mode Activated by $\overline{\text{DREQ}}$ Falling Edge

(5) Activation Timing by $\overline{\text{DREQ}}$ Low Level

Figure 7.31 shows an example of normal transfer mode activated by the $\overline{\text{DREQ}}$ signal low level.

The $\overline{\text{DREQ}}$ signal is sampled every cycle from the next rising edge of the Bϕ signal immediately after the DTE bit write cycle.

When a low level of the $\overline{\text{DREQ}}$ signal is detected while a transfer request by the $\overline{\text{DREQ}}$ signal is enabled, a transfer request is held in the DMAC. When the DMAC is activated, the transfer request is cleared. Receiving the next transfer request resumes after completion of the write cycle and then a low level of the $\overline{\text{DREQ}}$ signal is detected. This operation is repeated until the transfer is completed.



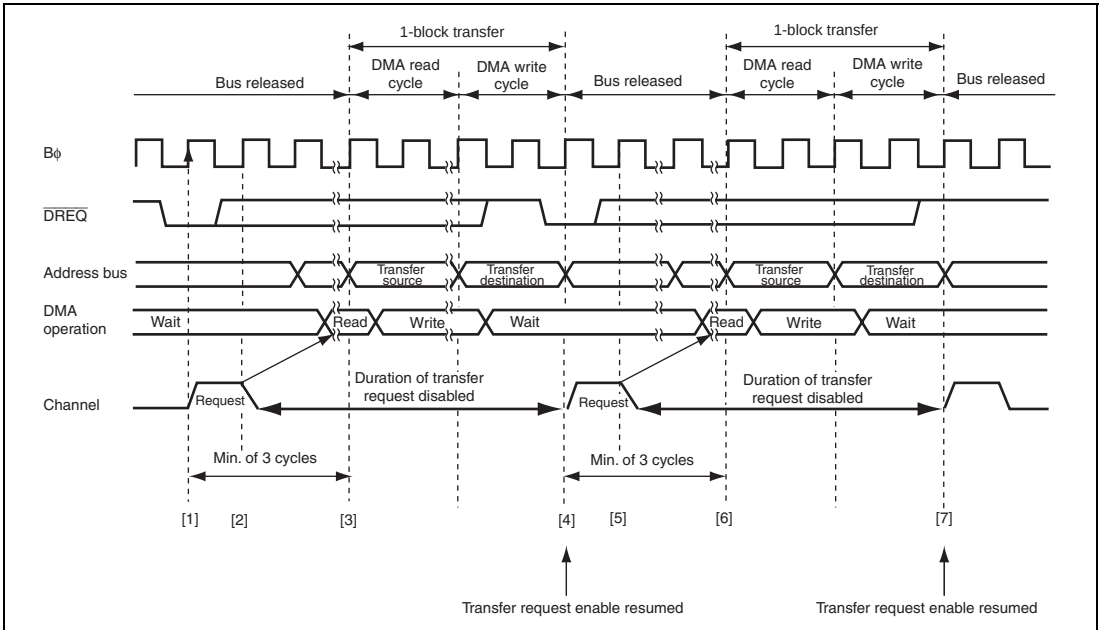
[1] After DMA transfer request is enabled, a low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held.
[2][5] The DMAC is activated and the transfer request is cleared.
[3][6] A DMA cycle is started.
[4][7] Transfer request enable is resumed after completion of the write cycle.
(A low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held. This is the same as [1].)

Figure 7.31 Example of Transfer in Normal Transfer Mode Activated by $\overline{\text{DREQ}}$ Low Level

Figure 7.32 shows an example of block transfer mode activated by the $\overline{\text{DREQ}}$ signal low level.

The $\overline{\text{DREQ}}$ signal is sampled every cycle from the next rising edge of the Bϕ signal immediately after the DTE bit write cycle.

When a low level of the $\overline{\text{DREQ}}$ signal is detected while a transfer request by the $\overline{\text{DREQ}}$ signal is enabled, a transfer request is held in the DMAC. When the DMAC is activated, the transfer request is cleared. Receiving the next transfer request resumes after completion of the write cycle and then a low level of the $\overline{\text{DREQ}}$ signal is detected. This operation is repeated until the transfer is completed.



[1] After DMA transfer request is enabled, a low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held.
[2][5] The DMAC is activated and the transfer request is cleared.
[3][6] A DMA cycle is started.
[4][7] Transfer request enable is resumed after completion of the write cycle.
(A low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held. This is the same as [1].)

Figure 7.32 Example of Transfer in Block Transfer Mode Activated by $\overline{\text{DREQ}}$ Low Level

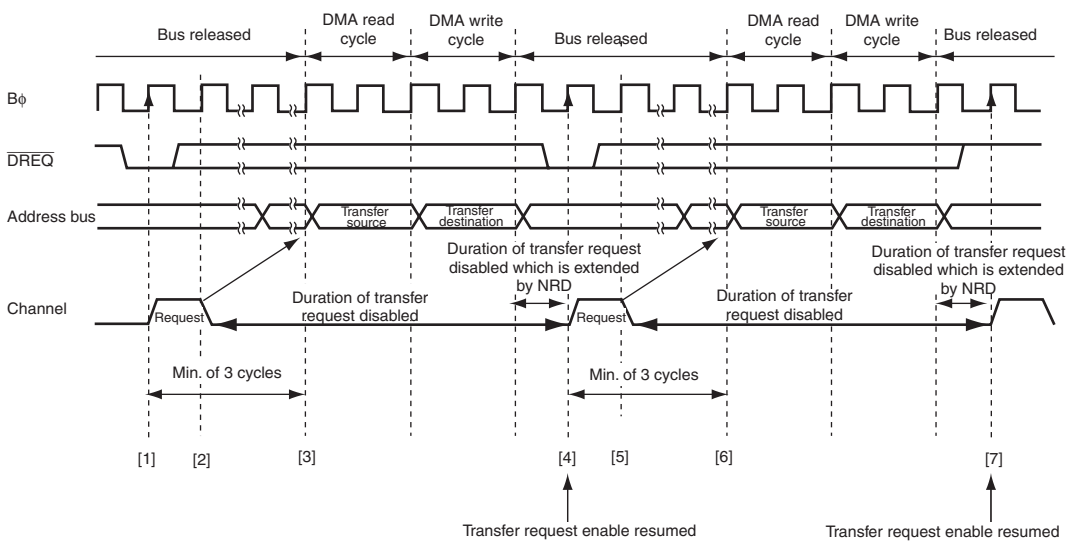
(6) Activation Timing by $\overline{\text{DREQ}}$ Low Level with NRD = 1

When the NRD bit in DMDR is set to 1, the timing of receiving the next transfer request is delayed for one cycle.

Figure 7.33 shows an example of normal transfer mode activated by the $\overline{\text{DREQ}}$ signal low level with NRD = 1.

The $\overline{\text{DREQ}}$ signal is sampled every cycle from the next rising edge of the Bϕ signal immediately after the DTE bit write cycle.

When a low level of the $\overline{\text{DREQ}}$ signal is detected while a transfer request by the $\overline{\text{DREQ}}$ signal is enabled, a transfer request is held in the DMAC. When the DMAC is activated, the transfer request is cleared. Receiving the next transfer request resumes after completion of the write cycle and then a low level of the $\overline{\text{DREQ}}$ signal is detected. This operation is repeated until the transfer is completed.



[1] After DMA transfer request is enabled, a low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held.
[2][5] The DMAC is activated and the transfer request is cleared.
[3][6] A DMA cycle is started.
[4][7] Transfer request enable is resumed one cycle after completion of the write cycle.
(A low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held. This is the same as [1].)

Figure 7.33 Example of Transfer in Normal Transfer Mode Activated by $\overline{\text{DREQ}}$ Low Level with NRD = 1

7.4.11 Bus Cycles in Single Address Mode

(1) Single Address Mode (Read and Cycle Stealing)

In single address mode, one byte, one word, or one longword of data is transferred at a single transfer request and after the transfer the bus is released temporarily. One bus cycle or more by the CPU or DTC are executed in the bus released cycles.

In figure 7.34, the $\overline{\text{TEND}}$ signal output is enabled and data is transferred in bytes from the external 8-bit 2-state access space to the external device in single address mode (read).

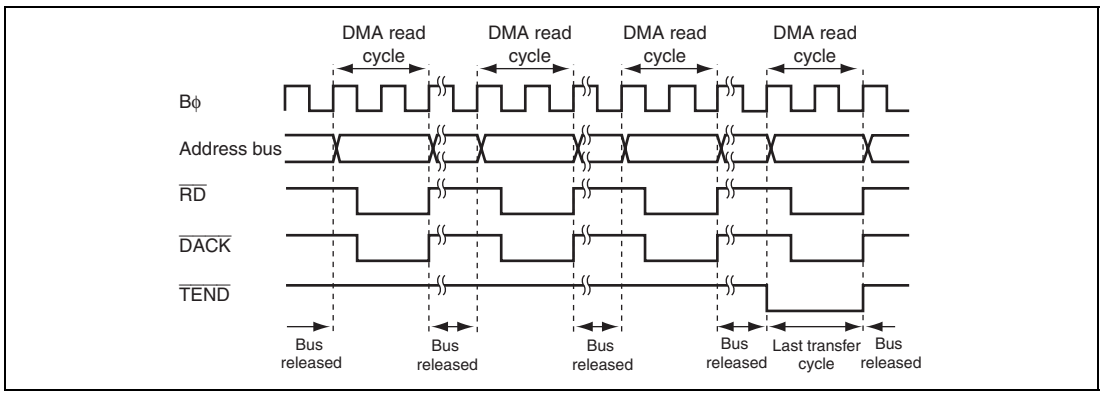


Figure 7.34 Example of Transfer in Single Address Mode (Byte Read)

(2) Single Address Mode (Write and Cycle Stealing)

In single address mode, data of one byte, one word, or one longword is transferred at a single transfer request and after the transfer the bus is released temporarily. One bus cycle or more by the CPU or DTC are executed in the bus released cycles.

In figure 7.35, the $\overline{\text{TEND}}$ signal output is enabled and data is transferred in bytes from the external 8-bit 2-state access space to the external device in single address mode (write).

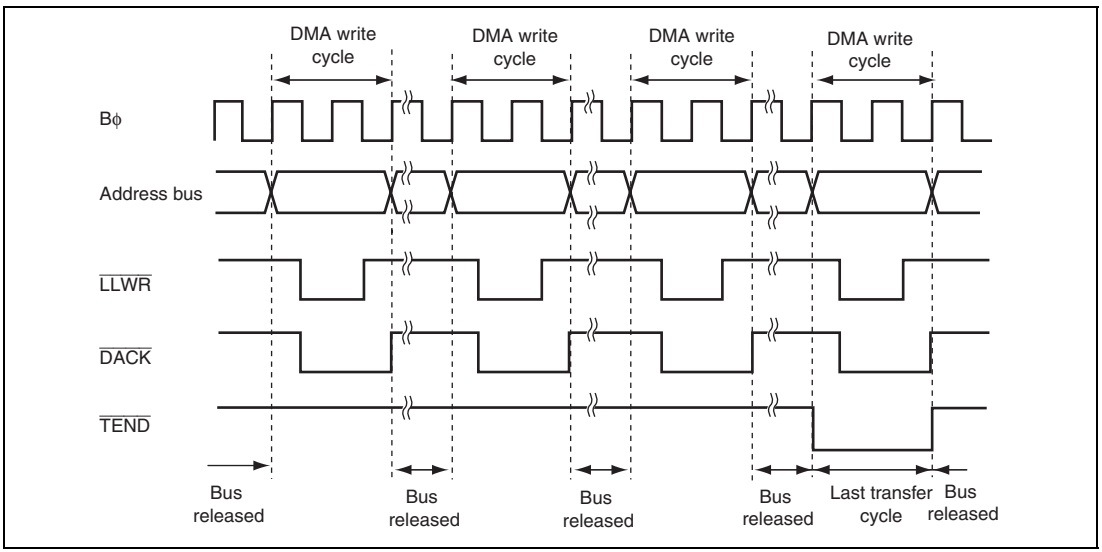


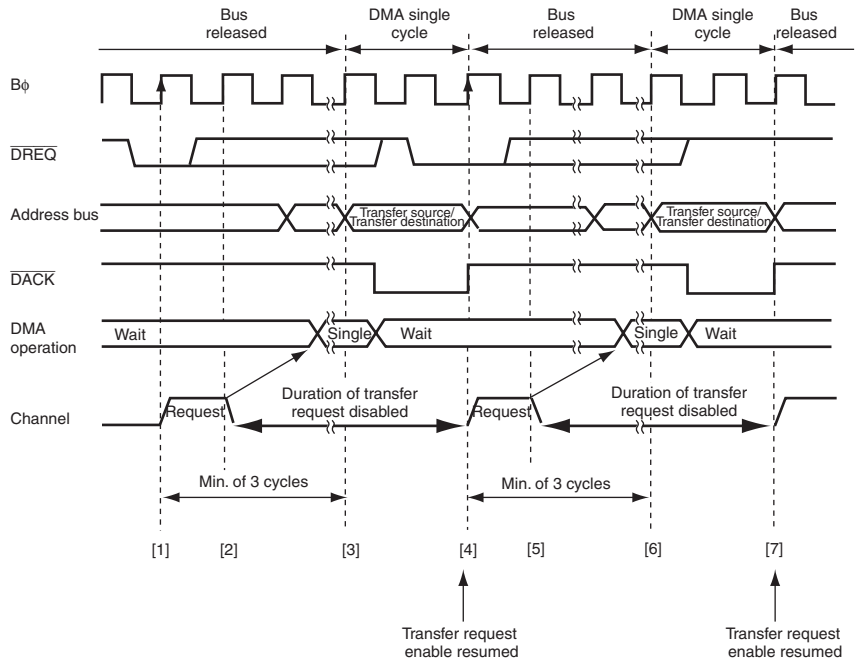
Figure 7.35 Example of Transfer in Single Address Mode (Byte Write)

(3) Activation Timing by $\overline{\text{DREQ}}$ Falling Edge

Figure 7.36 shows an example of single address mode activated by the $\overline{\text{DREQ}}$ signal falling edge.

The $\overline{\text{DREQ}}$ signal is sampled every cycle from the next rising edge of the Bϕ signal immediately after the DTE bit write cycle.

When a low level of the $\overline{\text{DREQ}}$ signal is detected while a transfer request by the $\overline{\text{DREQ}}$ signal is enabled, a transfer request is held in the DMAC. When the DMAC is activated, the transfer request is cleared and starts detecting a high level of the $\overline{\text{DREQ}}$ signal for falling edge detection. If a high level of the $\overline{\text{DREQ}}$ signal has been detected until completion of the single cycle, receiving the next transfer request resumes and then a low level of the $\overline{\text{DREQ}}$ signal is detected. This operation is repeated until the transfer is completed.



[1] After DMA transfer request is enabled, a low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held.
[2][5] The DMAC is activated and the transfer request is cleared.
[3][6] A DMA cycle is started and sampling the $\overline{\text{DREQ}}$ signal at the rising edge of the Bϕ signal is started to detect a high level of the $\overline{\text{DREQ}}$ signal.
[4][7] When a high level of the $\overline{\text{DREQ}}$ signal has been detected, transfer enable is resumed after completion of the write cycle. (A low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held. This is the same as [1].)

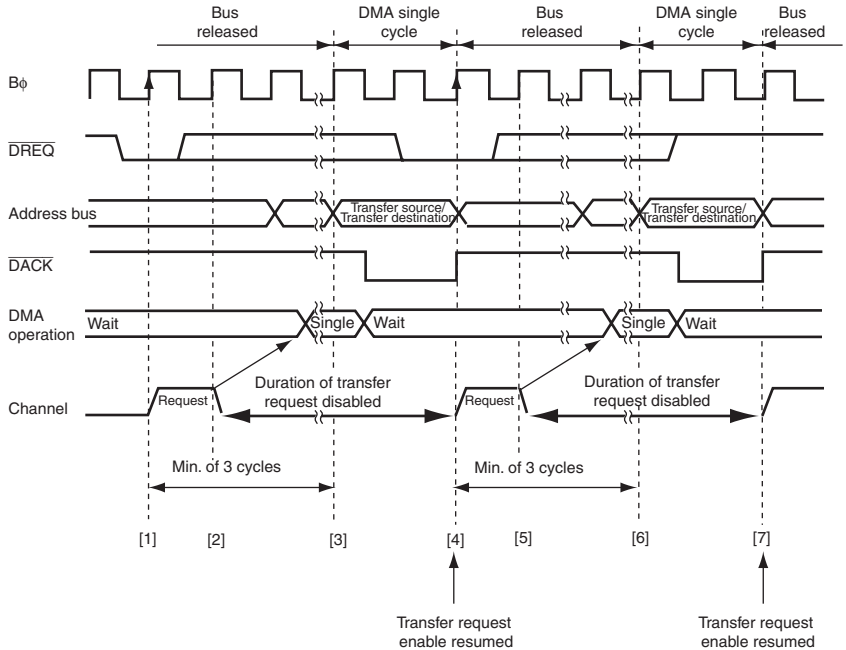
Figure 7.36 Example of Transfer in Single Address Mode Activated by $\overline{\text{DREQ}}$ Falling Edge

(4) Activation Timing by $\overline{\text{DREQ}}$ Low Level

Figure 7.37 shows an example of normal transfer mode activated by the $\overline{\text{DREQ}}$ signal low level.

The $\overline{\text{DREQ}}$ signal is sampled every cycle from the next rising edge of the Bϕ signal immediately after the DTE bit write cycle.

When a low level of the $\overline{\text{DREQ}}$ signal is detected while a transfer request by the $\overline{\text{DREQ}}$ signal is enabled, a transfer request is held in the DMAC. When the DMAC is activated, the transfer request is cleared. Receiving the next transfer request resumes after completion of the single cycle and then a low level of the $\overline{\text{DREQ}}$ signal is detected. This operation is repeated until the transfer is completed.



[1] After DMA transfer request is enabled, a low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held.
[2][5] The DMAC is activated and the transfer request is cleared.
[3][6] A DMA cycle is started.
[4][7] Transfer request enable is resumed after completion of the single cycle.
(A low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held. This is the same as [1].)

Figure 7.37 Example of Transfer in Single Address Mode Activated by $\overline{\text{DREQ}}$ Low Level

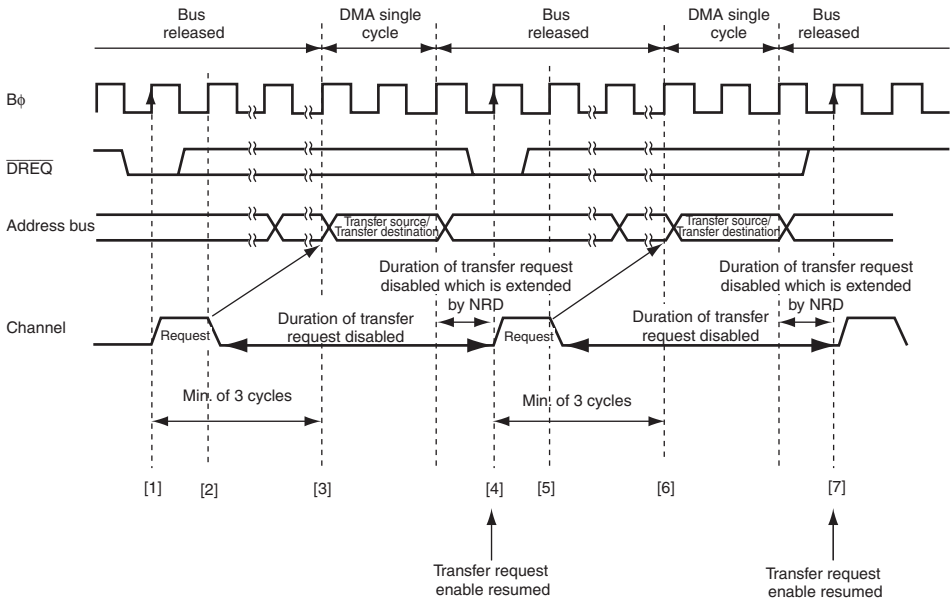
(5) Activation Timing by $\overline{\text{DREQ}}$ Low Level with NRD = 1

When the NRD bit in DMDR is set to 1, the timing of receiving the next transfer request is delayed for one cycle.

Figure 7.38 shows an example of single address mode activated by the $\overline{\text{DREQ}}$ signal low level with NRD = 1.

The $\overline{\text{DREQ}}$ signal is sampled every cycle from the next rising edge of the Bϕ signal immediately after the DTE bit write cycle.

When a low level of the $\overline{\text{DREQ}}$ signal is detected while a transfer request by the $\overline{\text{DREQ}}$ signal is enabled, a transfer request is held in the DMAC. When the DMAC is activated, the transfer request is cleared. Receiving the next transfer request resumes after one cycle of the transfer request duration inserted by NRD = 1 on completion of the single cycle and then a low level of the $\overline{\text{DREQ}}$ signal is detected. This operation is repeated until the transfer is completed.



[1] After DMA transfer request is enabled, a low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held.
[2][5] The DMAC is activated and the transfer request is cleared.
[3][6] A DMA cycle is started.
[4][7] Transfer request enable is resumed one cycle after completion of the single cycle.
(A low level of the $\overline{\text{DREQ}}$ signal is detected at the rising edge of the Bϕ signal and a transfer request is held. This is the same as [1].)

Figure 7.38 Example of Transfer in Single Address Mode Activated by $\overline{\text{DREQ}}$ Low Level with NRD = 1

7.5 DMA Transfer End

Operations on completion of a transfer differ according to the transfer end condition. DMA transfer completion indicates that the DTE and ACT bits in DMDR are changed from 1 to 0.

(1) Transfer End by DTCR Change from 1, 2, or 4, to 0

When DTCR is changed from 1, 2, or 4 to 0, a DMA transfer for the channel is completed. The DTE bit in DMDR is cleared to 0 and the DTIF bit in DMDR is set to 1. At this time, when the DTIE bit in DMDR is set to 1, a transfer end interrupt by the transfer counter is requested. When the DTCR value is 0 before the transfer, the transfer is not stopped.

(2) Transfer End by Transfer Size Error Interrupt

When the following conditions are satisfied while the TSEIE bit in DMDR is set to 1, a transfer size error occurs and a DMA transfer is terminated. At this time, the DTE bit in DMDR is cleared to 0 and the ESIF bit in DMDR is set to 1.

- In normal transfer mode and repeat transfer mode, when the next transfer is requested while a transfer is disabled due to the DTCR value less than the data access size.
- In block transfer mode, when the next transfer is requested while a transfer is disabled due to the DTCR value less than the block size.

When the TSEIE bit in DMDR is cleared to 0, data is transferred until the DTCR value reaches 0. A transfer size error is not generated. Operation in each transfer mode is shown below.

- In normal transfer mode and repeat transfer mode, when the DTCR value is less than the data access size, data is transferred in bytes.
- In block transfer mode, when the DTCR value is less than the block size, the specified size of data in DTCR is transferred instead of transferring the block size of data. The transfer is performed in bytes.

(3) Transfer End by Repeat Size End Interrupt

In repeat transfer mode, when the next transfer is requested after completion of a 1-repeat size data transfer while the RPTIE bit in DACR is set to 1, a repeat size end interrupt is requested. When the interrupt is requested to complete DMA transfer, the DTE bit in DMDR is cleared to 0 and the ESIF bit in DMDR is set to 1. Under this condition, setting the DTE bit to 1 resumes the transfer.

In block transfer mode, when the next transfer is requested after completion of a 1-block size data transfer, a repeat size end interrupt can be requested.

(4) Transfer End by Interrupt on Extended Repeat Area Overflow

When an overflow on the extended repeat area occurs while the extended repeat area is specified and the SARIE or DARIE bit in DACR is set to 1, an interrupt by an extended repeat area overflow is requested. When the interrupt is requested, the DMA transfer is terminated, the DTE bit in DMDR is cleared to 0, and the ESIF bit in DMDR is set to 1.

In dual address mode, even if an interrupt by an extended repeat area overflow occurs during a read cycle, the following write cycle is performed.

In block transfer mode, even if an interrupt by an extended repeat area overflow occurs during a 1-block transfer, the remaining data is transferred. The transfer is not terminated by an extended repeat area overflow interrupt unless the current transfer is complete.

(5) Transfer End by Clearing DTE Bit in DMDR

When the DTE bit in DMDR is cleared to 0 by the CPU, a transfer is completed after the current DMA cycle and a DMA cycle in which the transfer request is accepted are completed.

In block transfer mode, a DMA transfer is completed after 1-block data is transferred.

(6) Transfer End by NMI Interrupt

When an NMI interrupt is requested, the DTE bits for all the channels are cleared to 0 and the ERRF bit in DMDR_0 is set to 1. When an NMI interrupt is requested during a DMA transfer, the transfer is forced to stop. To perform DMA transfer after an NMI interrupt is requested, clear the ERRF bit to 0 and then set the DTE bits for the channels to 1.

The transfer end timings after an NMI interrupt is requested are shown below.

(a) Normal Transfer Mode and Repeat Transfer Mode

In dual address mode, a DMA transfer is completed after completion of the write cycle for one transfer unit.

In single address mode, a DMA transfer is completed after completion of the bus cycle for one transfer unit.

(b) Block Transfer Mode

A DMA transfer is forced to stop. Since a 1-block size of transfers is not completed, operation is not guaranteed.

In dual address mode, the write cycle corresponding to the read cycle is performed. This is similar to (a) in normal transfer mode.

(7) Transfer End by Address Error

When an address error occurs, the DTE bits for all the channels are cleared to 0 and the ERRF bit in DMDR_0 is set to 1. When an address error occurs during a DMA transfer, the transfer is forced to stop. To perform a DMA transfer after an address error occurs, clear the ERRF bit to 0 and then set the DTE bits to 1 for the channels.

The transfer end timing after an address error is the same as that after an NMI interrupt.

(8) Transfer End by Hardware Standby Mode* or Reset

The DMAC is initialized by a reset and a transition to the hardware standby mode*. A DMA transfer is not guaranteed.

Note: * This LSI does not have hardware standby mode.

7.6 Relationship among DMAC and Other Bus Masters

7.6.1 DMAC Priority Control Function Over CPU

The DMAC priority control function over CPU can be used according to the CPU priority control register (CPUPCR) setting. For details, see section 5.7, Priority Control Function of DTC and DMAC Over CPU.

The priority level of the DMAC is specified by bits DMAP2 to DMAP0 in DMDR and can be specified for each channel.

The priority level of the CPU is specified by bits CPUP2 to CPUP0. The value of bits CPUP2 to CPUP0 is updated according to the exception handling priority.

If the CPU priority control is enabled by setting the CPUPCE bit in CPUPCR to 1, when the CPU has priority over the DMAC, a transfer request for the corresponding channel is masked and the transfer is not activated. When another channel has priority over or the same as the CPU, a transfer request is received regardless of the priority between channels and the transfer is activated.

The transfer request masked by the CPU priority control function is suspended. When the transfer channel is given priority over the CPU by changing priority levels of the CPU or channel, the transfer request is received and the transfer is resumed. Writing 0 to the DTE bit clears the suspended transfer request.

When the CPUPCE bit is cleared to 0, it is regarded as the lowest priority.

7.6.2 Bus Arbitration among DMAC and Other Bus Masters

When DMA transfer cycles are consecutively performed, bus cycles of other bus masters may be inserted between the transfer cycles. The DMAC can release the bus temporarily to pass the bus to other bus masters.

The consecutive DMA transfer cycles may not be divided according to the transfer mode settings to achieve high-speed access.

The read and write cycles of a DMA transfer are not separated. Refreshing, external bus released, and on-chip bus master (CPU or DTC) cycles are not inserted between the read and write cycles of a DMA transfer.

In block transfer mode and an auto request transfer by burst access, bus cycles of the DMA transfer are consecutively performed. For this duration, since the DMAC has priority over the CPU and DTC, accesses to the external space is suspended (the IBCCS bit in the bus control register 2 (BCR2) is cleared to 0).

When the bus is passed to another channel or an auto request transfer by cycle stealing, bus cycles of the DMA and on-chip bus master are performed alternatively.

When the arbitration function among the DMAC and on-chip bus masters is enabled by setting the IBCCS bit in BCR2 to 1, the bus is used alternatively except the bus cycles which are not separated. For details, see section 6, Bus Controller (BSC).

A conflict may occur between external space access of the DMAC and an external bus released cycle. Even if a burst or block transfer is performed by the DMAC, the transfer is stopped temporarily and a cycle of external bus release is inserted by the BSC according to the external bus priority (when the CPU external access and the DTC external access do not have priority over a DMAC transfer, the transfers are not operated until the DMAC releases the bus).

In dual address mode, the DMAC releases the external bus after the external space write cycle. Since the external read cycle and the external write cycle are not separated, the bus is not released.

An internal space (on-chip memory and internal I/O registers) access of the DMAC and an external bus released cycle may be performed at the same time.

7.7 Interrupt Sources

The DMAC interrupt sources are a transfer end interrupt by the transfer counter and a transfer escape end interrupt which is generated when a transfer is terminated before the transfer counter reaches 0. Table 7.6 shows interrupt sources and priority.

Table 7.6 Interrupt Sources and Priority

Abbr.	Interrupt Sources	Priority
DMTEND0	Transfer end interrupt by channel 0 transfer counter	High
DMTEND1	Transfer end interrupt by channel 1 transfer counter	
DMTEND2	Transfer end interrupt by channel 2 transfer counter	
DMTEND3	Transfer end interrupt by channel 3 transfer counter	
DMEEND0	Interrupt by channel 0 transfer size error	
	Interrupt by channel 0 repeat size end	
	Interrupt by channel 0 extended repeat area overflow on source address	
	Interrupt by channel 0 extended repeat area overflow on destination address	
DMEEND1	Interrupt by channel 1 transfer size error	
	Interrupt by channel 1 repeat size end	
	Interrupt by channel 1 extended repeat area overflow on source address	
	Interrupt by channel 1 extended repeat area overflow on destination address	
DMEEND2	Interrupt by channel 2 transfer size error	
	Interrupt by channel 2 repeat size end	
	Interrupt by channel 2 extended repeat area overflow on source address	
	Interrupt by channel 2 extended repeat area overflow on destination address	
DMEEND3	Interrupt by channel 3 transfer size error	
	Interrupt by channel 3 repeat size end	
	Interrupt by channel 3 extended repeat area overflow on source address	
	Interrupt by channel 3 extended repeat area overflow on destination address	Low

Each interrupt is enabled or disabled by the DTIE and ESIE bits in DMDR for the corresponding channel. A DMTEND interrupt is generated by the combination of the DTIF and DTIE bits in DMDR. A DMEEND interrupt is generated by the combination of the ESIF and ESIE bits in

DMDR. The DMEEND interrupt sources are not distinguished. The priority among channels are decided by the interrupt controller and it is shown in table 7.6. For details, see section 5, Interrupt Controller.

Each interrupt source is specified by the interrupt enable bit in the register for the corresponding channel. A transfer end interrupt by the transfer counter, a transfer size error interrupt, a repeat size end interrupt, an interrupt by an extended repeat area overflow on the source address, and an interrupt by an extended repeat area overflow on the destination address are enabled or disabled by the DTIE bit in DMDR, the TSEIE bit in DMDR, the RPTIE bit in DACR, the SARIE bit in DACR, and the DARIE bit in DACR, respectively.

A transfer end interrupt by the transfer counter is generated when the DTIF bit in DMDR is set to 1. The DTIF bit is set to 1 when DTCR becomes 0 by a transfer while the DTIE bit in DMDR is set to 1.

An interrupt other than the transfer end interrupt by the transfer counter is generated when the ESIF bit in DMDR is set to 1. The ESIF bit is set to 1 when the conditions are satisfied by a transfer while the enable bit is set to 1.

A transfer size error interrupt is generated when the next transfer cannot be performed because the DTCR value is less than the data access size, meaning that the data access size of transfers cannot be performed. In block transfer mode, the block size is compared with the DTCR value for transfer error decision.

A repeat size end interrupt is generated when the next transfer is requested after completion of the repeat size of transfers in repeat transfer mode. Even when the repeat area is not specified in the address register, the transfer can be stopped periodically according to the repeat size. At this time, when a transfer end interrupt by the transfer counter is generated, the ESIF bit is set to 1.

An interrupt by an extended repeat area overflow on the source and destination addresses is generated when the address exceeds the extended repeat area (overflow). At this time, when a transfer end interrupt by the transfer counter, the ESIF bit is set to 1.

Figure 7.39 is a block diagram of interrupts and interrupt flags. To clear an interrupt, clear the DTIF or ESIF bit in DMDR to 0 in the interrupt handling routine or continue the transfer by setting the DTE bit in DMDR to 1 after resetting the register. Figure 7.40 shows the procedure to resume the transfer by clearing an interrupt.

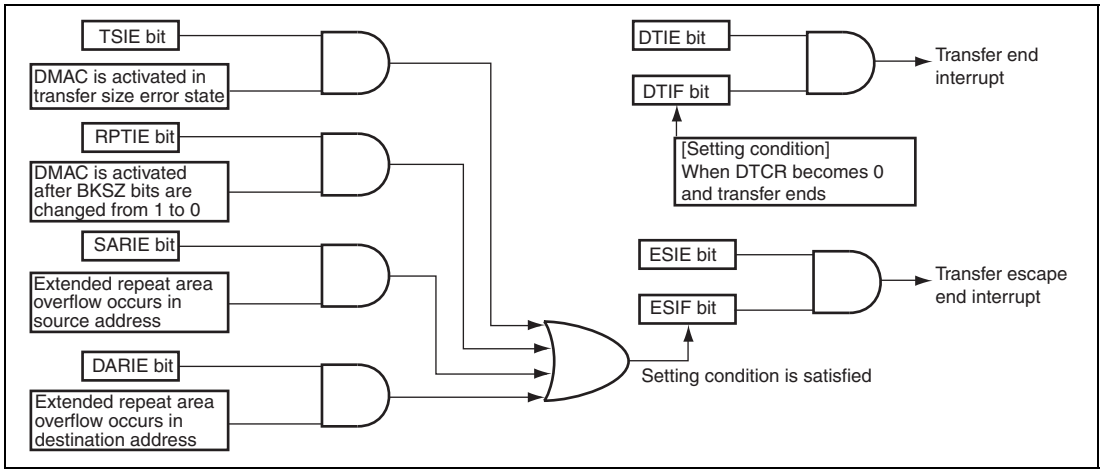
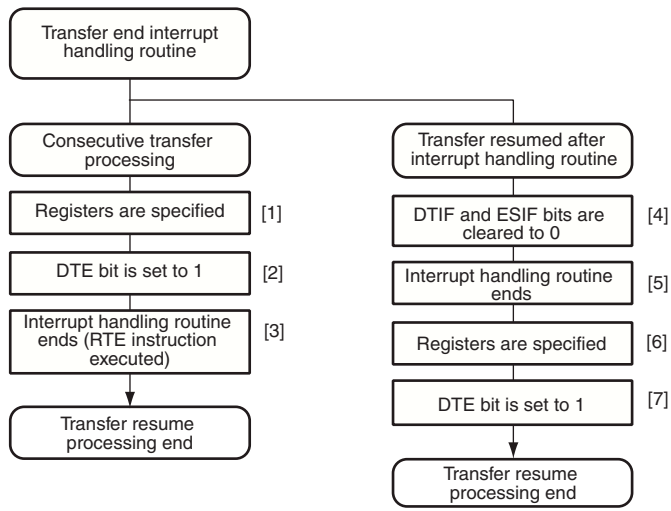


Figure 7.39 Interrupt and Interrupt Sources



[1] Specify the values in the registers such as transfer counter and address register.
[2] Set the DTE bit in DMDR to 1 to resume DMA operation. Setting the DTE bit to 1 automatically clears the DTIF or ESIF bit in DMDR to 0 and an interrupt source is cleared.
[3] End the interrupt handling routine by the RTE instruction.
[4] Read that the DTIF or the ESIF bit in DMDR = 1 and then write 0 to the bit.
[5] Complete the interrupt handling routine and clear the interrupt mask.
[6] Specify the values in the registers such as transfer counter and address register.
[7] Set the DTE bit to 1 to resume DMA operation.

Figure 7.40 Procedure Example of Resuming Transfer by Clearing Interrupt Source

7.8 Notes on Usage

1. DMAC Register Access During Operation

   Except for clearing the DTE bit in DMDR, the settings for channels being transferred (including waiting state) must not be changed. The register settings must be changed during the transfer prohibited state.

2. Settings of Module Stop Function

   The DMAC operation can be enabled or disabled by the module stop control register. The DMAC is enabled by the initial value.

   Setting bit MSTPA13 in MSTPCRA stops the clock supplied to the DMAC and the DMAC enters the module stop state. However, when a transfer for a channel is enabled or when an interrupt is being requested, bit MSTPA13 cannot be set to 1. Clear the DTE bit in DMDR to 0, clear the DTIF or DTIE bit in DMDR to 0, and then set bit MSTPA13.

   When the clock is stopped, the DMAC registers cannot be accessed. However, the following register settings are valid in the module stop state. Disable them before entering the module stop state, if necessary.

   — TENDE bit in DMDR is 1 (the $\overline{\text{TEND}}$ signal output enabled)

   — DACKE bit in DMDR is 1 (the $\overline{\text{DACK}}$ signal output enabled)

3. Activation by $\overline{\text{DREQ}}$ Falling Edge

   The $\overline{\text{DREQ}}$ falling edge detection is synchronized with the DMAC internal operation.

   A. Activation request waiting state: Waiting for detecting the $\overline{\text{DREQ}}$ low level. A transition to 2. is made.

   B. Transfer waiting state: Waiting for a DMAC transfer. A transition to 3. is made.

   C. Transfer prohibited state: Waiting for detecting the $\overline{\text{DREQ}}$ high level. A transition to 1. is made.

   After a DMAC transfer enabled, a transition to 1. is made. Therefore, the $\overline{\text{DREQ}}$ signal is sampled by low level detection at the first activation after a DMAC transfer enabled.

4. Acceptation of Activation Source

   At the beginning of an activation source reception, a low level is detected regardless of the setting of $\overline{\text{DREQ}}$ falling edge or low level detection. Therefore, if the $\overline{\text{DREQ}}$ signal is driven low before setting DMDR, the low level is received as a transfer request.

   When the DMAC is activated, clear the $\overline{\text{DREQ}}$ signal of the previous transfer.

Section 8 Data Transfer Controller (DTC)

This LSI has a data transfer controller (DTC). The DTC can be activated to transfer data by an interrupt request.

8.1 Features

- Transfer possible over any number of channels:
  Multiple data transfer enabled for one activation source (chain transfer)
  Chain transfer specifiable after data transfer (when the counter is 0)
- Three transfer modes
  Normal/repeat/block transfer modes selectable
  Transfer source and destination addresses can be selected from increment/decrement/fixed
- Short address mode or full address mode selectable
  - Short address mode
    Transfer information is located on a 3-longword boundary
    The transfer source and destination addresses can be specified by 24 bits to select a 16-Mbyte address space directly
  - Full address mode
    Transfer information is located on a 4-longword boundary
    The transfer source and destination addresses can be specified by 32 bits to select a 4-Gbyte address space directly
- Size of data for data transfer can be specified as byte, word, or longword
  The bus cycle is divided if an odd address is specified for a word or longword transfer.
  The bus cycle is divided if address 4n + 2 is specified for a longword transfer.
- A CPU interrupt can be requested for the interrupt that activated the DTC
  A CPU interrupt can be requested after one data transfer completion
  A CPU interrupt can be requested after the specified data transfer completion
- Read skip of the transfer information specifiable
- Writeback skip executed for the fixed transfer source and destination addresses
- Module stop mode specifiable

Note: Only short address mode is supported since this LSI has an address space of 16 Mbytes. Full address mode cannot be selected.

Figure 8.1 shows a block diagram of the DTC. The DTC transfer information can be allocated to the data area*. When the transfer information is allocated to the on-chip RAM, a 32-bit bus connects the DTC to the on-chip RAM, enabling 32-bit/1-state reading and writing (1 or 2 states for writing) of the DTC transfer information.

Note: * Configure the LSI such that reading/writing from/to RAM is enabled. See section 19, RAM for details.

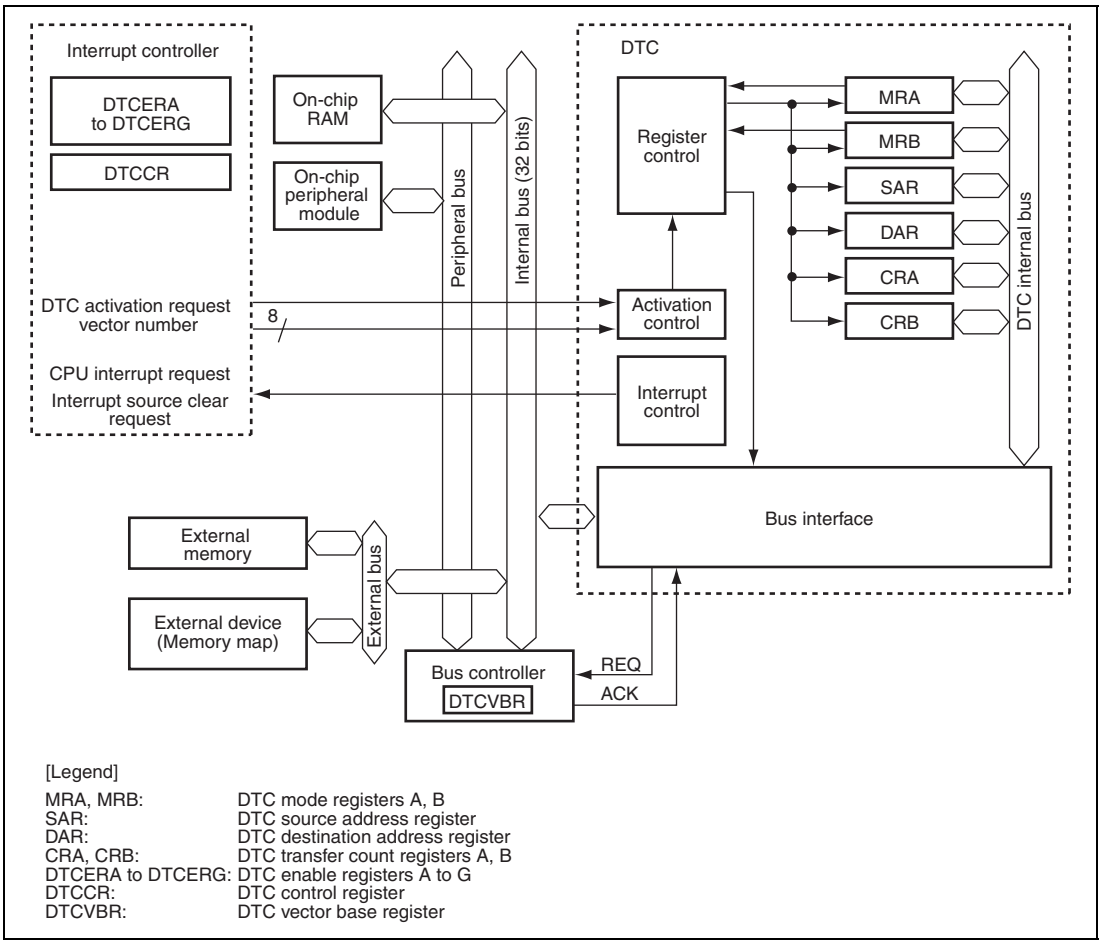


Figure 8.1 Block Diagram of DTC

8.2 Register Descriptions

DTC has the following registers.

- DTC mode register A (MRA)
- DTC mode register B (MRB)
- DTC source address register (SAR)
- DTC destination address register (DAR)
- DTC transfer count register A (CRA)
- DTC transfer count register B (CRB)

These six registers MRA, MRB, SAR, DAR, CRA, and CRB cannot be directly accessed by the CPU. The contents of these registers are stored in the data area as transfer information. When a DTC activation request occurs, the DTC reads a start address of transfer information that is stored in the data area according to the vector address, reads the transfer information, and transfers data. After the data transfer, it writes a set of updated transfer information back to the data area.

- DTC enable registers A to G (DTCERA to DTCERG)
- DTC control register (DTCCR)
- DTC vector base register (DTCVBR)

8.2.1 DTC Mode Register A (MRA)

MRA selects DTC operating mode. MRA cannot be accessed directly by the CPU.

Bit	7	6	5	4	3	2	1	0
Bit Name	MD1	MD0	Sz1	Sz0	SM1	SM0	—	—
Initial Value	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
R/W	—	—	—	—	—	—	—	—

Bit	Bit Name	Initial Value	R/W	Description
7	MD1	Undefined	—	DTC Mode 1 and 0
6	MD0	Undefined	—	Specify DTC transfer mode. 00: Normal transfer mode 01: Repeat transfer mode 10: Block transfer mode 11: Setting prohibited
5	Sz1	Undefined	—	DTC Data Transfer Size 1 and 0
4	Sz0	Undefined	—	Specify the size of data to be transferred. 00: Byte-size transfer 01: Word-size transfer 10: Longword-size transfer 11: Setting prohibited
3	SM1	Undefined	—	Source Address Mode 1 and 0
2	SM0	Undefined	—	Specify an SAR operation after a data transfer. 0X: SAR is fixed (SAR writeback is skipped) 10: SAR is incremented after a transfer (by 1 when Sz1 and Sz0 = B'00; by 2 when Sz1 and Sz0 = B'01; by 4 when Sz1 and Sz0 = B'10) 11: SAR is decremented after a transfer (by 1 when Sz1 and Sz0 = B'00; by 2 when Sz1 and Sz0 = B'01; by 4 when Sz1 and Sz0 = B'10)
1, 0	—	Undefined	—	Reserved The write value should always be 0.

X: Don't care

8.2.2 DTC Mode Register B (MRB)

MRB selects DTC operating mode. MRB cannot be accessed directly by the CPU.

Bit	7	6	5	4	3	2	1	0
Bit Name	CHNE	CHNS	DISEL	DTS	DM1	DM0	—	—
Initial Value	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
R/W	—	—	—	—	—	—	—	—

Bit	Bit Name	Initial Value	R/W	Description
7	CHNE	Undefined	—	DTC Chain Transfer Enable Specifies the chain transfer. For details, see section 8.5.7, Chain Transfer. The chain transfer condition is selected by the CHNS bit. 0: Disables the chain transfer 1: Enables the chain transfer
6	CHNS	Undefined	—	DTC Chain Transfer Select Specifies the chain transfer condition. If the following transfer is a chain transfer, the completion check of the specified transfer count is not performed and activation source flag or DTCER is not cleared. 0: Chain transfer every time 1: Chain transfer only when transfer counter = 0
5	DISEL	Undefined	—	DTC Interrupt Select When this bit is set to 1, a CPU interrupt request is generated every time after a data transfer ends. When this bit is set to 0, a CPU interrupt request is only generated when the specified number of data transfer ends.
4	DTS	Undefined	—	DTC Transfer Mode Select Specifies either the source or destination as repeat or block area during repeat or block transfer mode. 0: Specifies the destination as repeat or block area 1: Specifies the source as repeat or block area

Bit	Bit Name	Initial Value	R/W	Description
3	DM1	Undefined	—	Destination Address Mode 1 and 0 Specify a DAR operation after a data transfer. 0X: DAR is fixed (DAR writeback is skipped) 10: DAR is incremented after a transfer (by 1 when Sz1 and Sz0 = B'00; by 2 when Sz1 and Sz0 = B'01; by 4 when Sz1 and Sz0 = B'10) 11: SAR is decremented after a transfer (by 1 when Sz1 and Sz0 = B'00; by 2 when Sz1 and Sz0 = B'01; by 4 when Sz1 and Sz0 = B'10)
2	DM0	Undefined	—	
1, 0	—	Undefined	—	Reserved The write value should always be 0.

X: Don't care

8.2.3 DTC Source Address Register (SAR)

SAR is a 32-bit register that designates the source address of data to be transferred by the DTC.

In full address mode, 32 bits of SAR are valid. In short address mode, the lower 24 bits of SAR are valid and bits 31 to 24 are ignored. At this time, the upper eight bits are filled with the value specified by bit 23.

If a word or longword access is performed while an odd address is specified in SAR or if a longword access is performed while address 4n + 2 is specified in SAR, the bus cycle is divided into multiple cycles to transfer data. For details, see section 8.5.1, Bus Cycle Division.

SAR cannot be accessed directly from the CPU.

8.2.4 DTC Destination Address Register (DAR)

DAR is a 32-bit register that designates the destination address of data to be transferred by the DTC.

In full address mode, 32 bits of DAR are valid. In short address mode, the lower 24 bits of DAR are valid and bits 31 to 24 are ignored. At this time, the upper eight bits are filled with the value specified by bit 23.

If a word or longword access is performed while an odd address is specified in DAR or if a longword access is performed while address 4n + 2 is specified in DAR, the bus cycle is divided into multiple cycles to transfer data. For details, see section 8.5.1, Bus Cycle Division.

DAR cannot be accessed directly from the CPU.

8.2.5 DTC Transfer Count Register A (CRA)

CRA is a 16-bit register that designates the number of times data to be transferred by the DTC.

In normal transfer mode, CRA functions as a 16-bit transfer counter (1 to 65,536). It is decremented by 1 every time data is transferred, and bit DTCEn (n = 15 to 0) corresponding to the activation source is cleared and then an interrupt is requested to the CPU when the count reaches H'0000. The transfer count is 1 when CRA = H'0001, 65,535 when CRA = H'FFFF, and 65,536 when CRA = H'0000.

In repeat transfer mode, CRA is divided into two parts: the upper eight bits (CRAH) and the lower eight bits (CRAL). CRAH holds the number of transfers while CRAL functions as an 8-bit transfer counter (1 to 256). CRAL is decremented by 1 every time data is transferred, and the contents of CRAH are sent to CRAL when the count reaches H'00. The transfer count is 1 when CRAH = CRAL = H'01, 255 when CRAH = CRAL = H'FF, and 256 when CRAH = CRAL = H'00.

In block transfer mode, CRA is divided into two parts: the upper eight bits (CRAH) and the lower eight bits (CRAL). CRAH holds the block size while CRAL functions as an 8-bit block-size counter (1 to 256 for byte, word, or longword). CRAL is decremented by 1 every time a byte (word or longword) data is transferred, and the contents of CRAH are sent to CRAL when the count reaches H'00. The block size is 1 byte (word or longword) when CRAH = CRAL =H'01, 255 bytes (words or longwords) when CRAH = CRAL = H'FF, and 256 bytes (words or longwords) when CRAH = CRAL =H'00.

CRA cannot be accessed directly from the CPU.

8.2.6 DTC Transfer Count Register B (CRB)

CRB is a 16-bit register that designates the number of times data to be transferred by the DTC in block transfer mode. It functions as a 16-bit transfer counter (1 to 65,536) that is decremented by 1 every time data is transferred, and bit DTCEn (n = 15 to 0) corresponding to the activation source is cleared and then an interrupt is requested to the CPU when the count reaches H'0000. The transfer count is 1 when CRB = H'0001, 65,535 when CRB = H'FFFF, and 65,536 when CRB = H'0000.

CRB is not available in normal and repeat transfer modes and cannot be accessed directly by the CPU.

8.2.7 DTC Enable Registers A to G (DTCERA to DTCERG)

DTCER, which is comprised of DTCERA to DTCERG, is a register that specifies DTC activation interrupt sources. The correspondence between interrupt sources and DTCE bits is shown in table 8.1. Use bit manipulation instructions such as BSET and BCLR to read or write a DTCE bit. If all interrupts are masked, multiple activation sources can be set at one time (only at the initial setting) by writing data after executing a dummy read on the relevant register.

Bit	15	14	13	12	11	10	9	8
Bit Name	DTCE15	DTCE14	DTCE13	DTCE12	DTCE11	DTCE10	DTCE9	DTCE8
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	DTCE7	DTCE6	DTCE5	DTCE4	DTCE3	DTCE2	DTCE1	DTCE0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
15	DTCE15	0	R/W	DTC Activation Enable 15 to 0
14	DTCE14	0	R/W	Setting this bit to 1 specifies a relevant interrupt source to a DTC activation source.
13	DTCE13	0	R/W	
12	DTCE12	0	R/W	[Clearing conditions]
11	DTCE11	0	R/W	• When writing 0 to the bit to be cleared after reading 1
10	DTCE10	0	R/W	• When the DISEL bit is 1 and the data transfer has ended
9	DTCE9	0	R/W	• When the specified number of transfers has ended
8	DTCE8	0	R/W	These bits are not cleared when the DISEL bit is 0 and the specified number of transfers has not ended.
7	DTCE7	0	R/W	
6	DTCE6	0	R/W	
5	DTCE5	0	R/W	
4	DTCE4	0	R/W	
3	DTCE3	0	R/W	
2	DTCE2	0	R/W	
1	DTCE1	0	R/W	
0	DTCE0	0	R/W	

8.2.8 DTC Control Register (DTCCR)

DTCCR specifies transfer information read skip.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	RRS	RCHNE	—	—	ERR
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R	R	R/(W)*

Note: * Only 0 can be written to clear the flag.

Bit	Bit Name	Initial Value	R/W	Description
7 to 5	—	All 0	R/W	Reserved These bits are always read as 0. The write value should always be 0.

Bit	Bit Name	Initial Value	R/W	Description
4	RRS	0	R/W	DTC Transfer Information Read Skip Enable Controls the vector address read and transfer information read. A DTC vector number is always compared with the vector number for the previous activation. If the vector numbers match and this bit is set to 1, the DTC data transfer is started without reading a vector address and transfer information. If the previous DTC activation is a chain transfer, the vector address read and transfer information read are always performed. 0: Transfer read skip is not performed. 1: Transfer read skip is performed when the vector numbers match.
3	RCHNE	0	R/W	Chain Transfer Enable After DTC Repeat Transfer Enables/disables the chain transfer while transfer counter (CRAL) is 0 in repeat transfer mode. In repeat transfer mode, the CRAH value is written to CRAL when CRAL is 0. Accordingly, chain transfer may not occur when CRAL is 0. If this bit is set to 1, the chain transfer is enabled when CRAH is written to CRAL. 0: Disables the chain transfer after repeat transfer 1: Enables the chain transfer after repeat transfer
2, 1	—	All 0	R	Reserved These are read-only bits and cannot be modified.
0	ERR	0	R/(W)*	Transfer Stop Flag Indicates that an address error or an NMI interrupt occurs. If an address error or an NMI interrupt occurs, the DTC stops. 0: No interrupt occurs 1: An interrupt occurs [Clearing condition] • When writing 0 after reading 1

Note: * Only 0 can be written to clear this flag.

8.2.9 DTC Vector Base Register (DTCVBR)

DTCVBR is a 32-bit register that specifies the base address for vector table address calculation. Bits 31 to 28 and bits 11 to 0 are fixed 0 and cannot be written to. The initial value of DTCVBR is H'00000000.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Bit Name																
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Name																
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R	R	R	R	R

8.3 Activation Sources

The DTC is activated by an interrupt request. The interrupt source is selected by DTCER. A DTC activation source can be selected by setting the corresponding bit in DTCER; the CPU interrupt source can be selected by clearing the corresponding bit in DTCER to 0. At the end of a data transfer (or the last consecutive transfer in the case of chain transfer), the activation source interrupt flag or corresponding DTCER bit is cleared.

8.4 Location of Transfer Information and DTC Vector Table

The transfer information is located in the data area. The start address of transfer information should be located at the address that is a multiple of four (4n). Otherwise, the lower two bits are ignored during access ([1:0] = B'00.) Transfer information can be located in either short address mode (three longwords) or full address mode (four longwords). Transfer information located in the data area is shown in figure 8.2.

The DTC reads the start address of transfer information from the vector table according to the activation source, and then reads the transfer information from the start address. Figure 8.3 shows correspondences between the DTC vector address and transfer information.

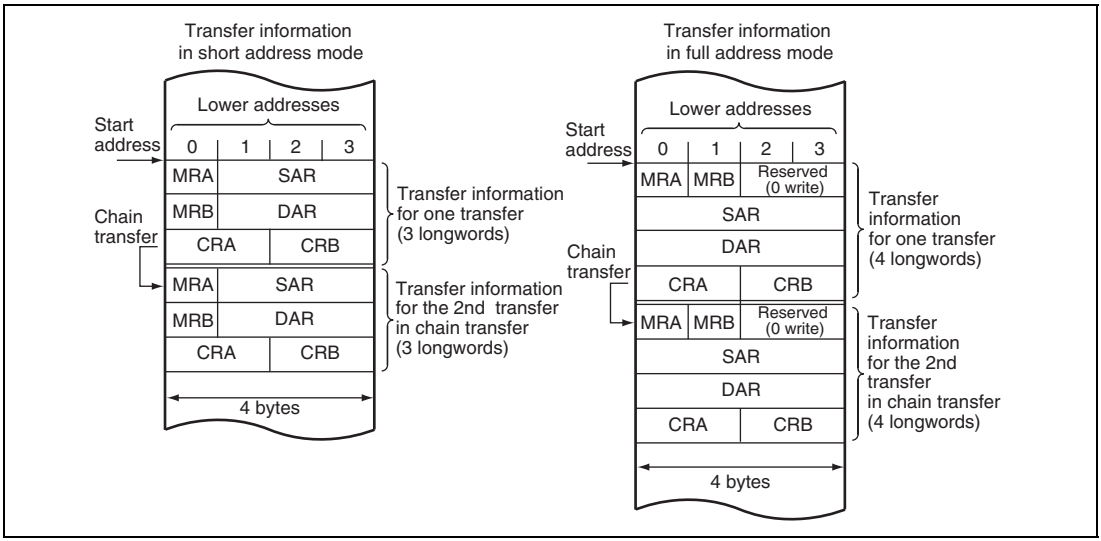


Figure 8.2 Transfer Information on Data Area

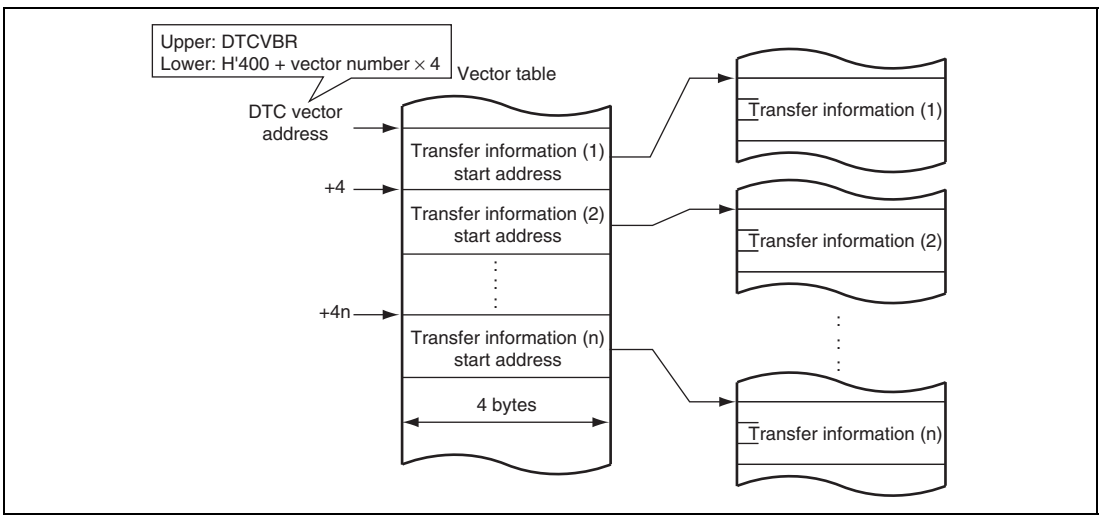


Figure 8.3 Correspondence between DTC Vector Address and Transfer Information

Table 8.1 shows correspondence between the DTC activation source and vector address.

Table 8.1 Interrupt Sources, DTC Vector Addresses, and Corresponding DTCEs

Origin of Activation Source	Activation Source	Vector Number	DTC Vector Address Offset	DTCE*[1]	Priority
External pin	IRQ0	64	H'500	DTCEA15	High
	IRQ1	65	H'504	DTCEA14	
	IRQ2	66	H'508	DTCEA13	
	IRQ3	67	H'50C	DTCEA12	
	IRQ4	68	H'510	DTCEA11	
	IRQ5	69	H'514	DTCEA10	
	IRQ6	70	H'518	DTCEA9	
	IRQ7	71	H'51C	DTCEA8	
	IRQ8	72	H'520	DTCEA7	
	IRQ9	73	H'524	DTCEA6	
	IRQ10	74	H'528	DTCEA5	
	IRQ11	75	H'52C	DTCEA4	
	IRQ12	76	H'530	DTCEA3	
	IRQ13	77	H'534	DTCEA2	
	IRQ14	78	H'538	DTCEA1	
A/D_0	ADI0 (A/D_0 conversion end)	86	H'558	DTCEB15	
A/D_1	ADI1 (A/D_1 conversion end)	87	H'55C	DTCEB14	
TPU_0	TGI0A	88	H'560	DTCEB13	
	TGI0B	89	H'564	DTCEB12	
	TGI0C	90	H'568	DTCEB11	
	TGI0D	91	H'56C	DTCEB10	
TPU_1	TGI1A	93	H'574	DTCEB9	
	TGI1B	94	H'578	DTCEB8	
TPU_2	TGI2A	97	H'584	DTCEB7	
	TGI2B	98	H'588	DTCEB6	Low

Origin of Activation Source	Activation Source	Vector Number	DTC Vector Address Offset	DTCE*1	Priority
TPU_3	TGI3A	101	H'594	DTCEB5	High
	TGI3B	102	H'598	DTCEB4	
	TGI3C	103	H'59C	DTCEB3	
	TGI3D	104	H'5A0	DTCEB2	
TPU_4	TGI4A	106	H'5A8	DTCEB1	
	TGI4B	107	H'5AC	DTCEB0	
TPU_5	TGI5A	110	H'5B8	DTCEC15	
	TGI5B	111	H'5BC	DTCEC14	
TMR_0	CMI0A	116	H'5D0	DTCEC13	
	CMI0B	117	H'5D4	DTCEC12	
TMR_1	CMI1A	119	H'5DC	DTCEC11	
	CMI1B	120	H'5E0	DTCEC10	
DMAC	DMTEND0	128	H'600	DTCEC3	
	DMTEND1	129	H'604	DTCEC2	
	DMTEND2	130	H'608	DTCEC1	
	DMTEND3	131	H'60C	DTCEC0	
RCAN_0	RM0_0	132	H'610	DTCED15	
	ERS0_0/OVR0_0/RM1_0/SLE0_0	133	H'614	DTCED14	
RCAN_1	RM0_1	134	H'618	DTCED13	
	ERS0_1/OVR0_1/RM1_1/SLE0_1	135	H'61C	DTCED12	
DMAC	DMEEND0	136	H'620	DTCED11	
	DMEEND1	137	H'624	DTCED10	
	DMEEND2	138	H'628	DTCED9	
	DMEEND3	139	H'62C	DTCED8	
SCI_3	RXI3	157	H'674	DTCEE15	
	TXI3	158	H'678	DTCEE14	
SCI_4	RXI4	161	H'684	DTCEE13	
	TXI4	162	H'688	DTCEE12	Low

Origin of Activation Source	Activation Source	Vector Number	DTC Vector Address Offset	DTCE*1	Priority
TPU_6	TGI6A	164	H'690	DTCEE11	High
	TGI6B	165	H'694	DTCEE10	
	TGI6C	166	H'698	DTCEE9	
	TGI6D	167	H'69C	DTCEE8	
TPU_7	TGI7A	169	H'6A4	DTCEE7	
	TGI7B	170	H'6A8	DTCEE6	
TPU_8	TGI8A	173	H'6B4	DTCEE5	
	TGI8B	174	H'6B8	DTCEE4	
TPU_9	TGI9A	177	H'6C4	DTCEE3	
	TGI9B	178	H'6C8	DTCEE2	
	TGI9C	179	H'6CC	DTCEE1	
	TGI9D	180	H'6D0	DTCEE0	
TPU_10	TGI10A	182	H'6D8	DTCEF15	
	TGI10B	183	H'6DC	DTCEF14	
	Reserved for system use	184	H'6E0	DTCEF13	
		185	H'6E4	DTCEF12	
	TCI10V*2	186	H'6E8	DTCEF11	
TPU_11	TGI11A	188	H'6F0	DTCEF10	
	TGI11B	189	H'6F4	DTCEF9	
RSPI_0	SPRI_0	197	H'714	DTCEF5	
	SPTI_0	198	H'718	DTCEF4	
RSPI_1	SPRI_1	200	H'720	DTCEF3	
	SPTI_1	201	H'724	DTCEF2	
RSPI_2	SPRI_2	203	H'72C	DTCEF1	
	SPTI_2	204	H'730	DTCEF0	
RSPI_3	SPRI_3	206	H'738	DTCEG15	
	SPTI_3	207	H'73C	DTCEG14	Low

Notes: 1. The DTCE bits with no corresponding interrupt are reserved, and the write value should always be 0. To leave software standby mode or all-module-clock-stop mode with an interrupt, write 0 to the corresponding DTCE bit.

2. TCI10V does not activate the DTC.

8.5 Operation

The DTC stores transfer information in the data area. When activated, the DTC reads transfer information that is stored in the data area and transfers data on the basis of that transfer information. After the data transfer, it writes updated transfer information back to the data area. Since transfer information is in the data area, it is possible to transfer data over any required number of channels. There are three transfer modes: normal, repeat, and block.

The DTC specifies the source address and destination address in SAR and DAR, respectively. After a transfer, SAR and DAR are incremented, decremented, or fixed independently.

Table 8.2 shows the DTC transfer modes.

Table 8.2 DTC Transfer Modes

Transfer Mode	Size of Data Transferred at One Transfer Request	Memory Address Increment or Decrement	Transfer Count
Normal	1 byte/word/longword	Incremented/decremented by 1, 2, or 4, or fixed	1 to 65536
Repeat*1	1 byte/word/longword	Incremented/decremented by 1, 2, or 4, or fixed	1 to 256*3
Block*2	Block size specified by CRAH (1 to 256 bytes/words/longwords)	Incremented/decremented by 1, 2, or 4, or fixed	1 to 65536

Notes: 1. Either source or destination is specified to repeat area.
2. Either source or destination is specified to block area.
3. After transfer of the specified transfer count, initial state is recovered to continue the operation.

Setting the CHNE bit in MRB to 1 makes it possible to perform a number of transfers with a single activation (chain transfer). Setting the CHNS bit in MRB to 1 can also be made to have chain transfer performed only when the transfer counter value is 0.

Figure 8.4 shows a flowchart of DTC operation, and table 8.3 summarizes the chain transfer conditions (combinations for performing the second and third transfers are omitted).

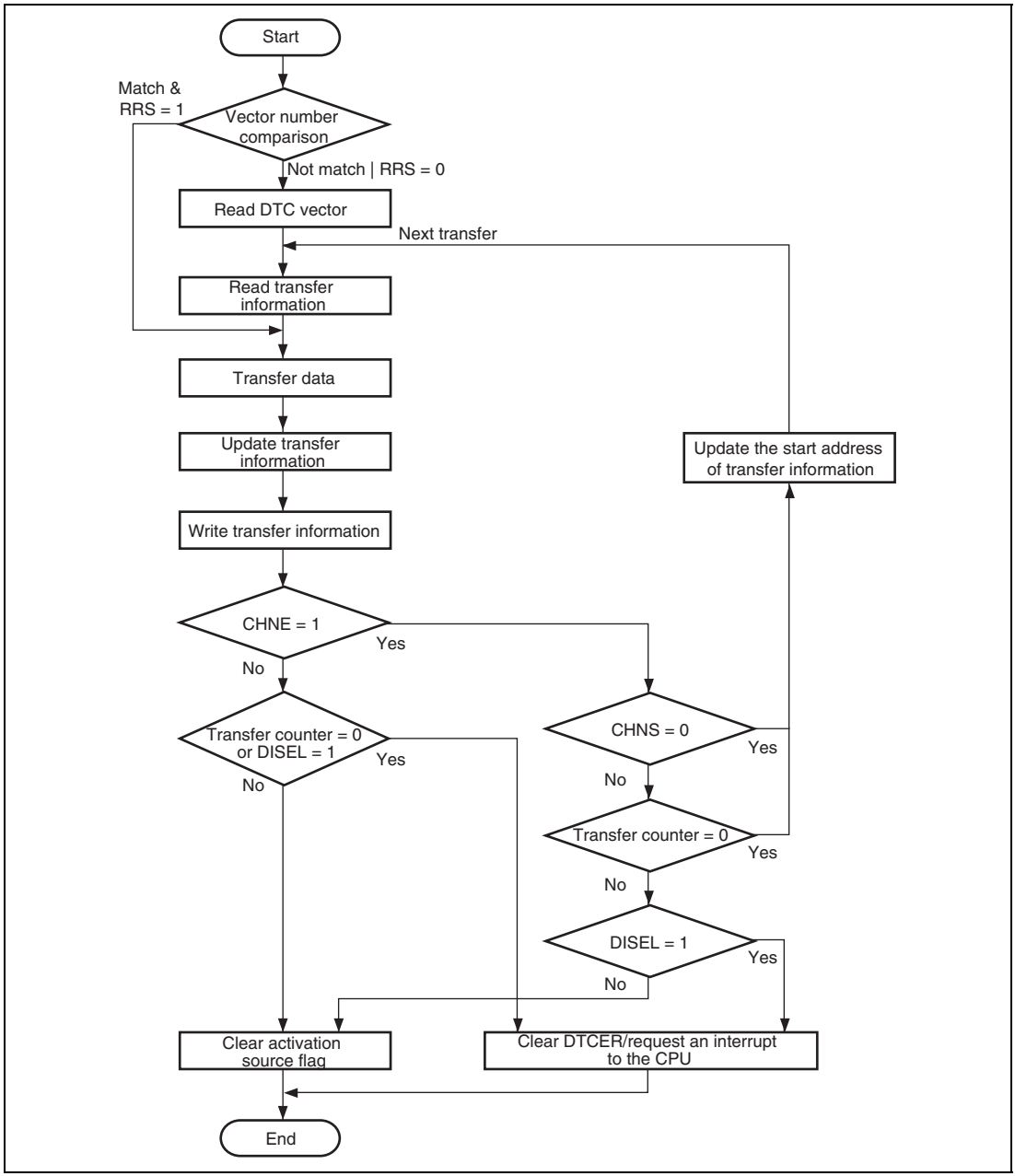


Figure 8.4 Flowchart of DTC Operation

Table 8.3 Chain Transfer Conditions

1st Transfer				2nd Transfer				
CHNE	CHNS	DISEL	Transfer Counter*1	CHNE	CHNS	DISEL	Transfer Counter*1	DTC Transfer
0	—	0	Not 0	—	—	—	—	Ends at 1st transfer
0	—	0	0*2	—	—	—	—	Ends at 1st transfer
0	—	1		—	—	—	—	Interrupt request to CPU
1	0	—	—	0	—	0	Not 0	Ends at 2nd transfer
				0	—	0	0*2	Ends at 2nd transfer
				0	—	1	—	Interrupt request to CPU
1	1	0	Not 0	—	—		—	Ends at 1st transfer
1	1	—	0*2	0	—	0	Not 0	Ends at 2nd transfer
				0	—	0	0*2	Ends at 2nd transfer
				0	—	1		Interrupt request to CPU
1	1	1	Not 0	—	—	—	—	Ends at 1st transfer
								Interrupt request to CPU

Notes: 1. CRA in normal mode transfer, CRAL in repeat transfer mode, or CRB in block transfer mode
2. When the contents of the CRAH is written to the CRAL in repeat transfer mode

8.5.1 Bus Cycle Division

When the transfer data size is word and the SAR and DAR values are not a multiple of 2, the bus cycle is divided and the transfer data is read from or written to in bytes. Similarly, when the transfer data size is longword and the SAR and DAR values are not a multiple of 4, the bus cycle is divided and the transfer data is read from or written to in words.

Table 8.4 shows the relationship among, SAR, DAR, transfer data size, bus cycle divisions, and access data size. Figure 8.5 shows the bus cycle division example.

Table 8.4 Number of Bus Cycle Divisions and Access Size

	Specified Data Size		
SAR and DAR Values	Byte (B)	Word (W)	Longword (LW)
Address 4n	1 (B)	1 (W)	1 (LW)
Address 2n + 1	1 (B)	2 (B-B)	3 (B-W-B)
Address 4n + 2	1 (B)	1 (W)	2 (W-W)

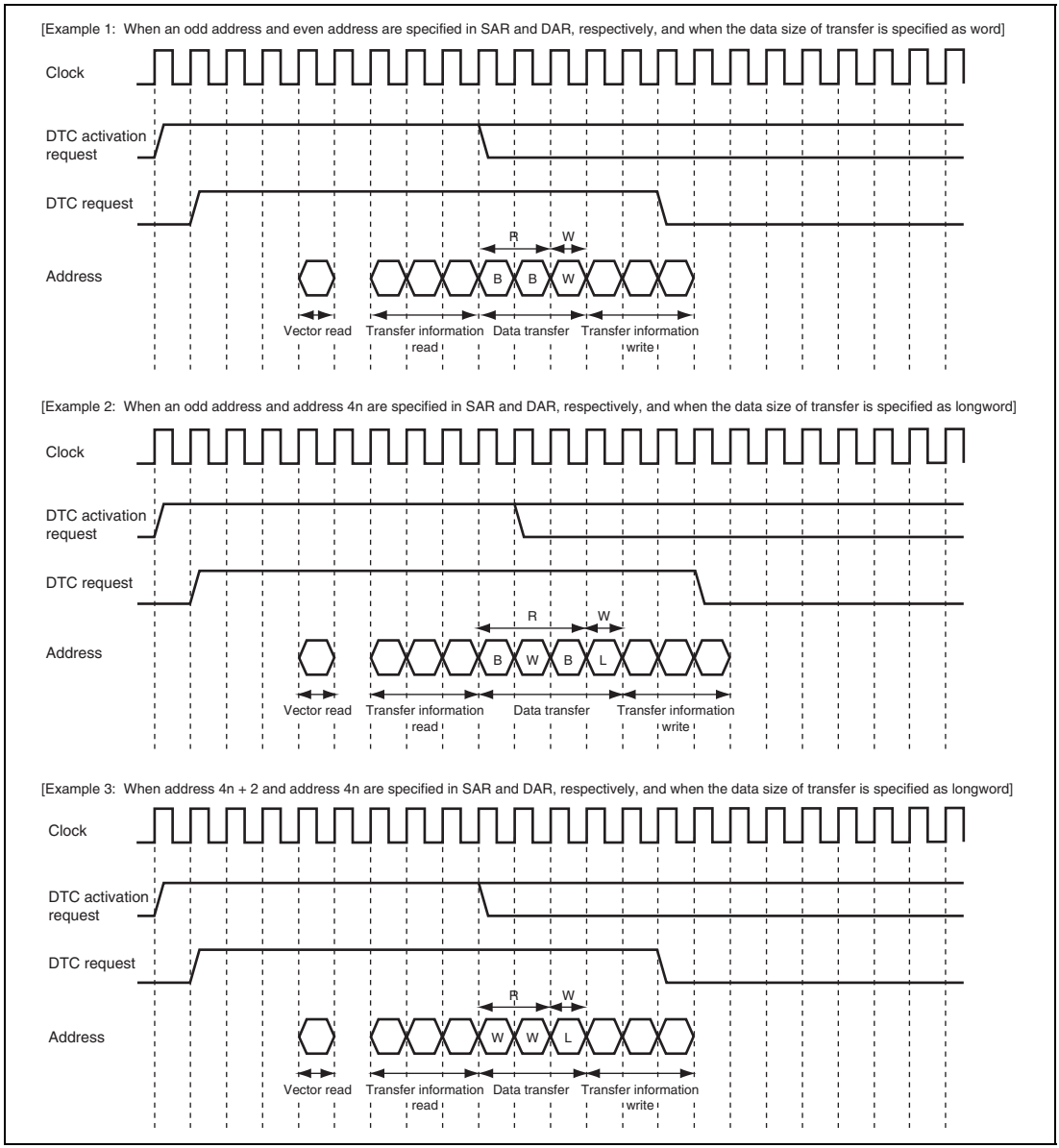


Figure 8.5 Bus Cycle Division Example

8.5.2 Transfer Information Read Skip Function

By setting the RRS bit of DTCCR, the vector address read and transfer information read can be skipped. The current DTC vector number is always compared with the vector number of previous activation. If the vector numbers match when RRS = 1, a DTC data transfer is performed without reading the vector address and transfer information. If the previous activation is a chain transfer, the vector address read and transfer information read are always performed. Figure 8.6 shows the transfer information read skip timing.

To modify the vector table and transfer information, temporarily clear the RRS bit to 0, modify the vector table and transfer information, and then set the RRS bit to 1 again. When the RRS bit is cleared to 0, the stored vector number is deleted, and the updated vector table and transfer information are read at the next activation.

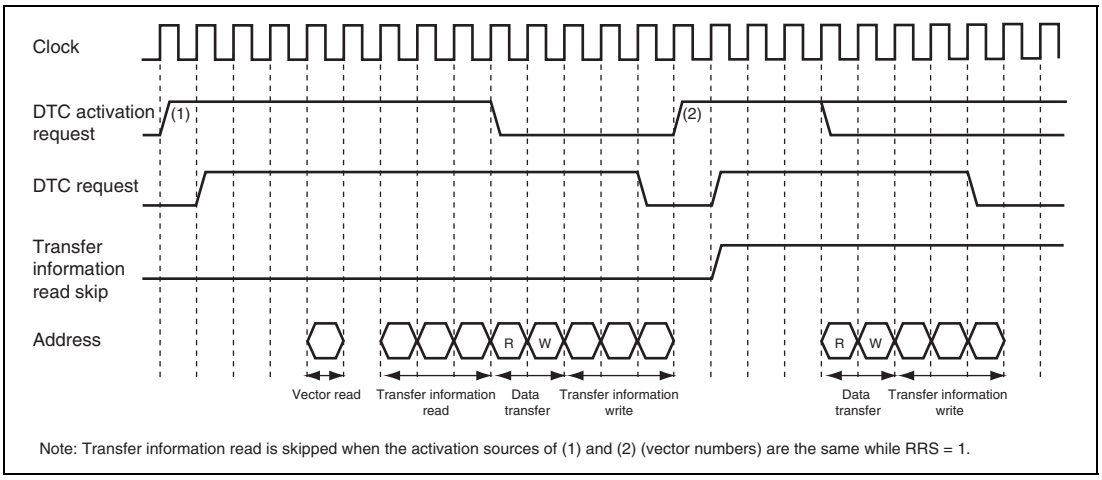


Figure 8.6 Transfer Information Read Skip Timing

8.5.3 Transfer Information Writeback Skip Function

By specifying bit SM1 in MRA and bit DM1 in MRB to the fixed address mode, a part of transfer information will not be written back. This function is performed regardless of short or full address mode. Table 8.5 shows the transfer information writeback skip condition and writeback skipped registers. Note that the CRA and CRB are always written back regardless of the short or full address mode. In addition in full address mode, the writeback of the MRA and MRB are always skipped.

Table 8.5 Transfer Information Writeback Skip Condition and Writeback Skipped Registers

SM1	DM1	SAR	DAR
0	0	Skipped	Skipped
0	1	Skipped	Written back
1	0	Written back	Skipped
1	1	Written back	Written back

8.5.4 Normal Transfer Mode

In normal transfer mode, one operation transfers one byte, one word, or one longword of data. From 1 to 65,536 transfers can be specified. The transfer source and destination addresses can be specified as incremented, decremented, or fixed. When the specified number of transfers ends, an interrupt can be requested to the CPU.

Table 8.6 lists the register function in normal transfer mode. Figure 8.7 shows the memory map in normal transfer mode.

Table 8.6 Register Function in Normal Transfer Mode

Register	Function	Written Back Value
SAR	Source address	Incremented/decremented/fixed*
DAR	Destination address	Incremented/decremented/fixed*
CRA	Transfer count A	CRA – 1
CRB	Transfer count B	Not updated

Note: * Transfer information writeback is skipped.

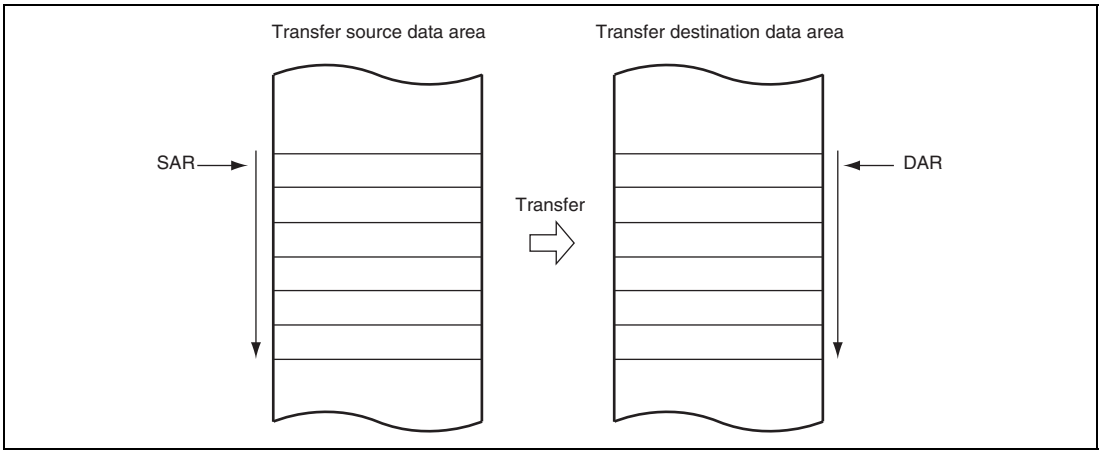


Figure 8.7 Memory Map in Normal Transfer Mode

8.5.5 Repeat Transfer Mode

In repeat transfer mode, one operation transfers one byte, one word, or one longword of data. By the DTS bit in MRB, either the source or destination can be specified as a repeat area. From 1 to 256 transfers can be specified. When the specified number of transfers ends, the transfer counter and address register specified as the repeat area is restored to the initial state, and transfer is repeated. The other address register is then incremented, decremented, or left fixed. In repeat transfer mode, the transfer counter (CRAL) is updated to the value specified in CRAH when CRAL becomes H'00. Thus the transfer counter value does not reach H'00, and therefore a CPU interrupt cannot be requested when DISEL = 0.

Table 8.7 lists the register function in repeat transfer mode. Figure 8.8 shows the memory map in repeat transfer mode.

Table 8.7 Register Function in Repeat Transfer Mode

		Written Back Value	
Register	**Function**	**CRAL is not 1**	**CRAL is 1**
SAR	Source address	Incremented/decremented/fixed*	DTS =0: Incremented/ decremented/fixed* DTS = 1: SAR initial value
DAR	Destination address	Incremented/decremented/fixed*	DTS = 0: DAR initial value DTS =1: Incremented/ decremented/fixed*
CRAH	Transfer count storage	CRAH	CRAH
CRAL	Transfer count A	CRAL – 1	CRAH
CRB	Transfer count B	Not updated	Not updated

Note: * Transfer information writeback is skipped.

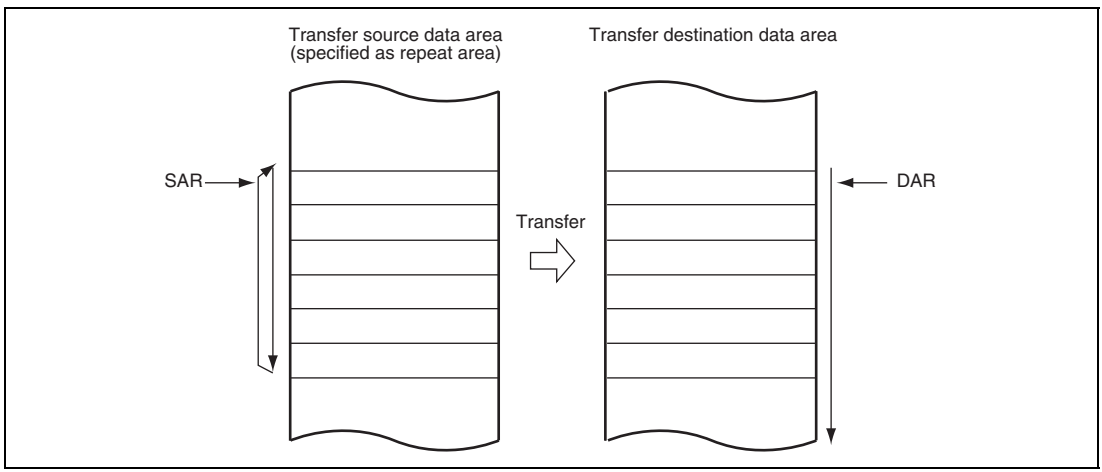


Figure 8.8 Memory Map in Repeat Transfer Mode (When Transfer Source is Specified as Repeat Area)

8.5.6 Block Transfer Mode

In block transfer mode, one operation transfers one block of data. Either the transfer source or the transfer destination is designated as a block area by the DTS bit in MRB.

The block size is 1 to 256 bytes (1 to 256 words, or 1 to 256 longwords). When the transfer of one block ends, the block size counter (CRAL) and address register (SAR when DTS = 1 or DAR when DTS = 0) specified as the block area is restored to the initial state. The other address register is then incremented, decremented, or left fixed. From 1 to 65,536 transfers can be specified. When the specified number of transfers ends, an interrupt is requested to the CPU.

Table 8.8 lists the register function in block transfer mode. Figure 8.9 shows the memory map in block transfer mode.

Table 8.8 Register Function in Block Transfer Mode

Register	Function	Written Back Value
SAR	Source address	DTS = 0: Incremented/decremented/fixed*
		DTS = 1: SAR initial value
DAR	Destination address	DTS = 0: DAR initial value
		DTS = 1: Incremented/decremented/fixed*
CRAH	Block size storage	CRAH
CRAL	Block size counter	CRAH
CRB	Block transfer counter	CRB – 1

Note: * Transfer information writeback is skipped.

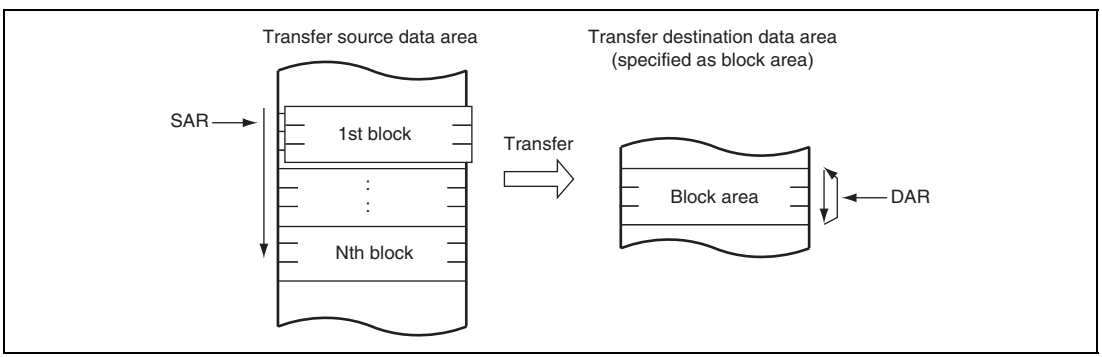


Figure 8.9 Memory Map in Block Transfer Mode (When Transfer Destination is Specified as Block Area)

8.5.7 Chain Transfer

Setting the CHNE bit in MRB to 1 enables a number of data transfers to be performed consecutively in response to a single transfer request. Setting the CHNE and CHNS bits in MRB set to 1 enables a chain transfer only when the transfer counter reaches 0. SAR, DAR, CRA, CRB, MRA, and MRB, which define data transfers, can be set independently. Figure 8.10 shows the chain transfer operation.

In the case of transfer with CHNE set to 1, an interrupt request to the CPU is not generated at the end of the specified number of transfers or by setting the DISEL bit to 1, and the interrupt source flag for the activation source and DTCER are not affected.

In repeat transfer mode, setting the RCHNE bit in DTCCR and the CHNE and CHNS bits in MRB to 1 enables a chain transfer after transfer with transfer counter = 1 has been completed.

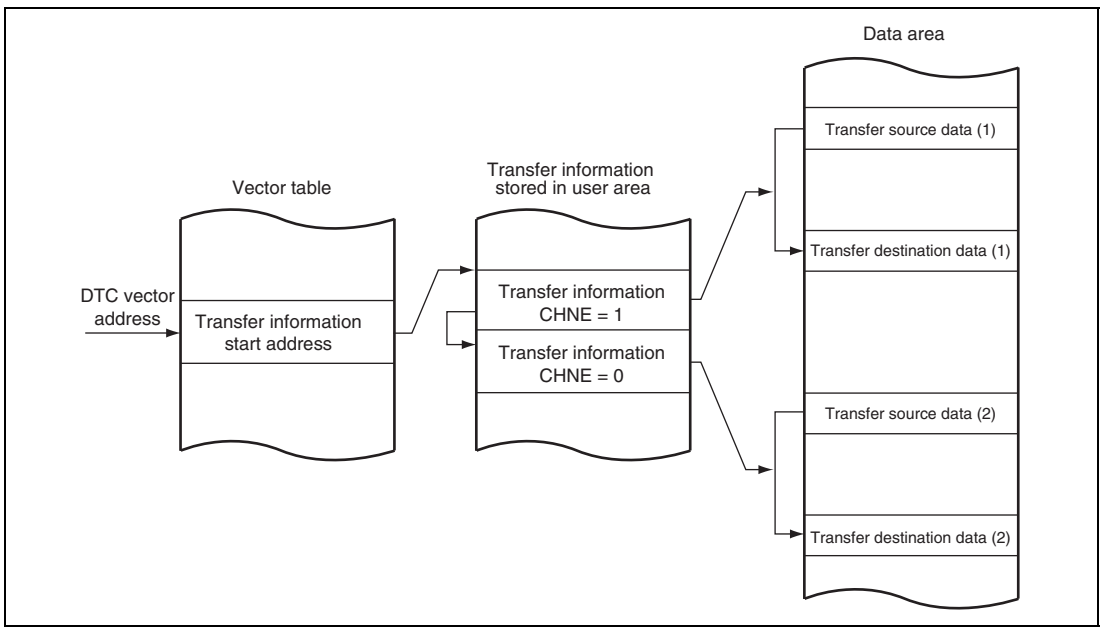


Figure 8.10 Operation of Chain Transfer

8.5.8 Operation Timing

Figures 8.11 to 8.14 show the DTC operation timings.

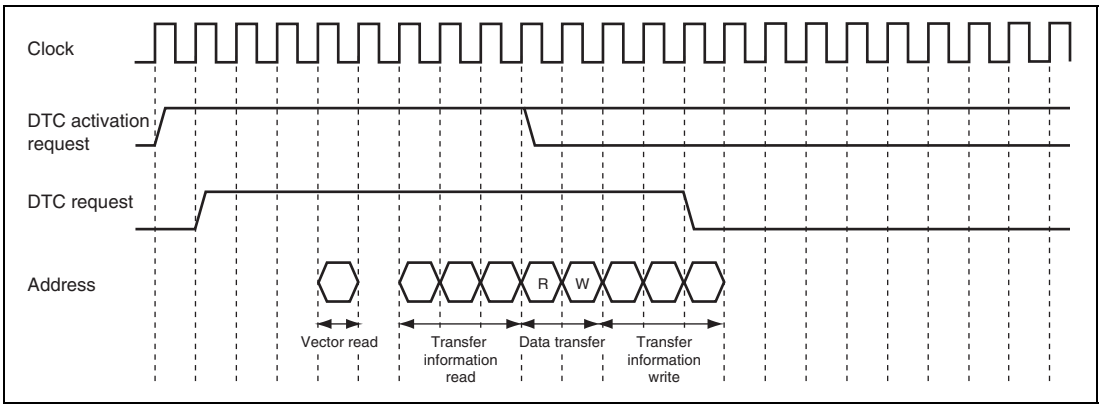


Figure 8.11 DTC Operation Timing
(Example of Short Address Mode in Normal Transfer Mode or Repeat Transfer Mode)

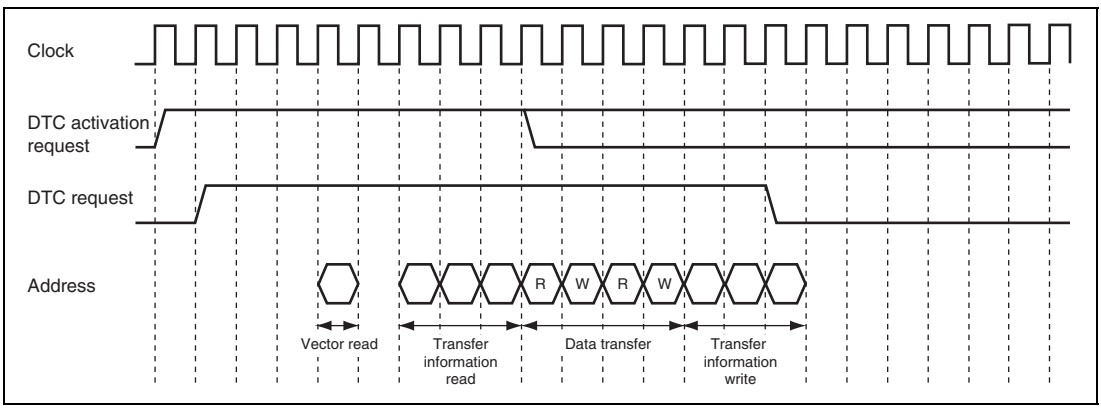


Figure 8.12 DTC Operation Timing
(Example of Short Address Mode in Block Transfer Mode with Block Size of 2)

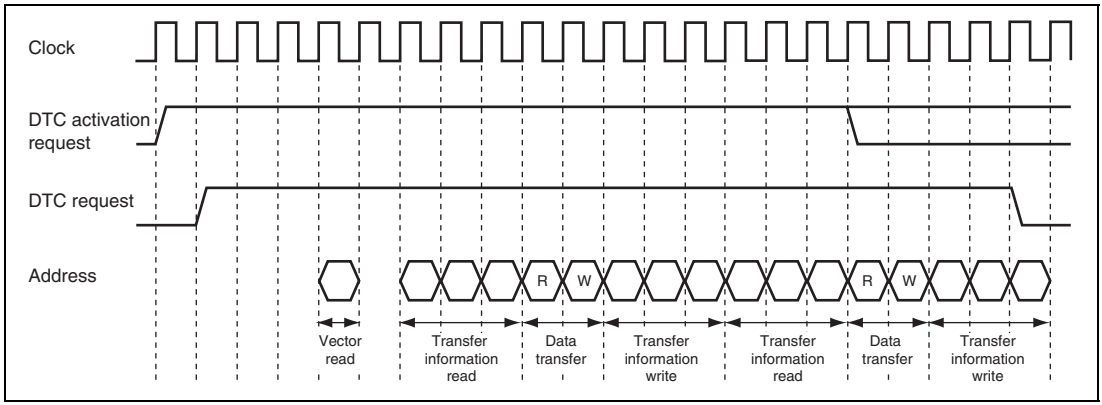


Figure 8.13 DTC Operation Timing (Example of Short Address Mode in Chain Transfer)

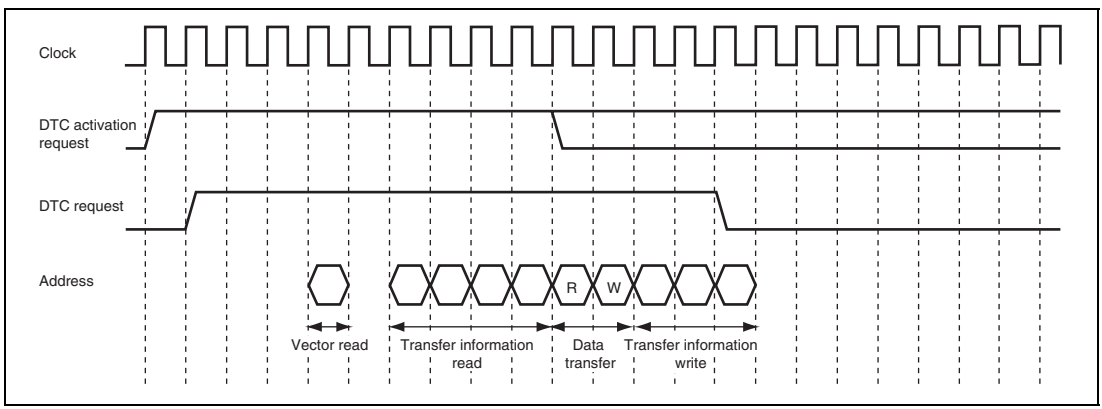


Figure 8.14 DTC Operation Timing
(Example of Full Address Mode in Normal Transfer Mode or Repeat Transfer Mode)

8.5.9 Number of DTC Execution Cycles

Table 8.9 shows the execution status for a single DTC data transfer, and table 8.10 shows the number of cycles required for each execution.

Table 8.9 DTC Execution Status

Mode	Vector Read I		Transfer Information Read J			Transfer Information Write K			Data Read L			Data Write M			Internal Operation N	
Normal	1	0*1	4*2	3*3	0*1	3*2.3	2*4	1*5	3*6	2*7	1	3*6	2*7	1	1	0*1
Repeat	1	0*1	4*2	3*3	0*1	3*2.3	2*4	1*5	3*6	2*7	1	3*6	2*7	1	1	0*1
Block transfer	1	0*1	4*2	3*3	0*1	3*2.3	2*4	1*5	3•P*6	2•P*7	1•P	3•P*6	2•P*7	1•P	1	0*1

[Legend]

P: Block size (CRAH and CRAL value)

Note: 1. When transfer information read is skipped
2. In full address mode operation
3. In short address mode operation
4. When the SAR or DAR is in fixed mode
5. When the SAR and DAR are in fixed mode
6. When a longword is transferred while an odd address is specified in the address register
7. When a word is transferred while an odd address is specified in the address register or when a longword is transferred while address 4n + 2 is specified

Table 8.10 Number of Cycles Required for Each Execution State

Object to be Accessed		On-Chip RAM	On-Chip ROM	On-Chip I/O Registers			External Devices			
Bus width		32	32	8	16	32	8		16	
Access cycles		1	1	2	2	2	2	3	2	3
Execution status	Vector read S_I	1	1	—	—	—	8	12 + 4m	4	6 + 2m
	Transfer information read S_J	1	1	—	—	—	8	12 + 4m	4	6 + 2m
	Transfer information write S_K	1	1	—	—	—	8	12 + 4m	4	6 + 2m
	Byte data read S_L	1	1	2	2	2	2	3 + m	2	3 + m
	Word data read S_L	1	1	4	2	2	4	4 + 2m	2	3 + m
	Longword data read S_L	1	1	8	4	2	8	12 + 4m	4	6 + 2m
	Byte data write S_M	1	1	2	2	2	2	3 + m	2	3 + m
	Word data write S_M	1	1	4	2	2	4	4 + 2m	2	3 + m
	Longword data write S_M	1	1	8	4	2	8	12 + 4m	4	6 + 2m
	Internal operation S_N	1								

[Legend]
m: Number of wait cycles 0 to 7

The number of execution cycles is calculated from the formula below. Note that Σ means the sum of all transfers activated by one activation event (the number in which the CHNE bit is set to 1, plus 1).

$$\text{Number of execution cycles} = I \cdot S_I + \Sigma (J \cdot S_J + K \cdot S_K + L \cdot S_L + M \cdot S_M) + N \cdot S_N$$

8.5.10 DTC Bus Release Timing

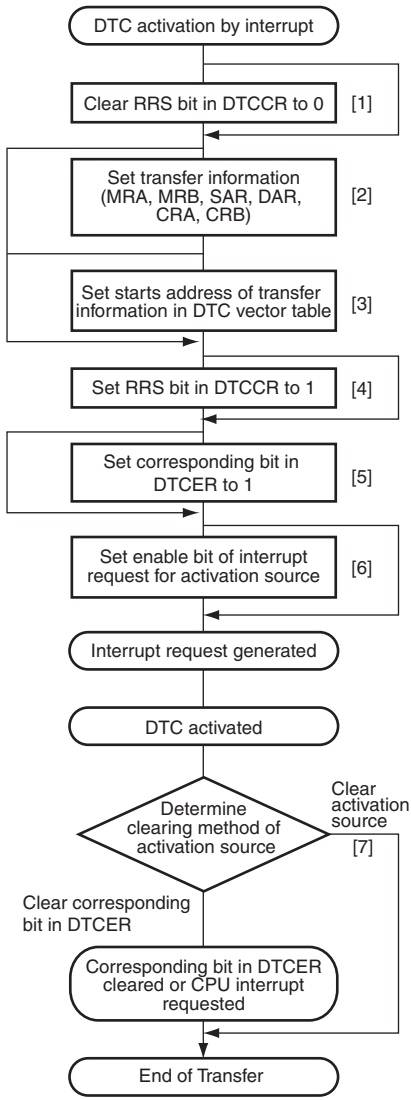
The DTC requests the bus mastership to the bus arbiter when an activation request occurs. The DTC releases the bus after a vector read, transfer information read, a single data transfer, or transfer information writeback. The DTC does not release the bus during transfer information read, single data transfer, or transfer information writeback.

8.5.11 DTC Priority Level Control to the CPU

The priority of the DTC activation sources over the CPU can be controlled by the CPU priority level specified by bits CPUP2 to CPUP0 in CPUPCR and the DTC priority level specified by bits DTCP2 to DTCP0. For details, see section 5, Interrupt Controller.

8.6 DTC Activation by Interrupt

The procedure for using the DTC with interrupt activation is shown in figure 8.15.



[1] Clearing the RRS bit in DTCCR to 0 clears the read skip flag of transfer information. Read skip is not performed when the DTC is activated after clearing the RRS bit. When updating transfer information, the RRS bit must be cleared.

[2] Set the MRA, MRB, SAR, DAR, CRA, and CRB transfer information in the data area. For details on setting transfer information, see section 8.2, Register Descriptions. For details on location of transfer information, see section 8.4, Location of Transfer Information and DTC Vector Table.

[3] Set the start address of the transfer information in the DTC vector table. For details on setting DTC vector table, see section 8.4, Location of Transfer Information and DTC Vector Table.

[4] Setting the RRS bit to 1 performs a read skip of second time or later transfer information when the DTC is activated consecutively by the same interrupt source. Setting the RRS bit to 1 is always allowed. However, the value set during transfer will be valid from the next transfer.

[5] Set the bit in DTCER corresponding to the DTC activation interrupt source to 1. For the correspondence of interrupts and DTCER, refer to table 8.1. The bit in DTCER may be set to 1 on the second or later transfer. In this case, resetting the bit is not needed.

[6] Set the enable bits for the interrupt sources to be used as the activation sources to 1. The DTC is activated when an interrupt used as an activation source is generated. For details on the settings of the interrupt enable bits, see the corresponding descriptions of the corresponding module.

[7] After the end of one data transfer, the DTC clears the activation source flag or clears the corresponding bit in DTCER and requests an interrupt to the CPU. The operation after transfer depends on the transfer information. For details, see section 8.2, Register Descriptions and figure 8.4.

Figure 8.15 DTC with Interrupt Activation

8.7 Examples of Use of the DTC

8.7.1 Normal Transfer Mode

An example is shown in which the DTC is used to receive 128 bytes of data via the SCI.

1. Set MRA to fixed source address (SM1 = SM0 = 0), incrementing destination address (DM1 = 1, DM0 = 0), normal transfer mode (MD1 = MD0 = 0), and byte size (Sz1 = Sz0 = 0). The DTS bit can have any value. Set MRB for one data transfer by one interrupt (CHNE = 0, DISEL = 0). Set the RDR address of the SCI in SAR, the start address of the RAM area where the data will be received in DAR, and 128 (H'0080) in CRA. CRB can be set to any value.
2. Set the start address of the transfer information for an RXI interrupt at the DTC vector address.
3. Set the corresponding bit in DTCER to 1.
4. Set the SCI to the appropriate receive mode. Set the RIE bit in SCR to 1 to enable the receive end (RXI) interrupt. Since the generation of a receive error during the SCI reception operation will disable subsequent reception, the CPU should be enabled to accept receive error interrupts.
5. Each time reception of one byte of data ends on the SCI, the RDRF flag in SSR is set to 1, an RXI interrupt is generated, and the DTC is activated. The receive data is transferred from RDR to RAM by the DTC. DAR is incremented and CRA is decremented. The RDRF flag is automatically cleared to 0.
6. When CRA becomes 0 after the 128 data transfers have ended, the RDRF flag is held at 1, the DTCE bit is cleared to 0, and an RXI interrupt request is sent to the CPU. Termination processing should be performed in the interrupt handling routine.

8.7.2 Chain Transfer

An example of DTC chain transfer is shown in which pulse output is performed using the PPG. Chain transfer can be used to perform pulse output data transfer and PPG output trigger cycle updating. Repeat mode transfer to the PPG's NDR is performed in the first half of the chain transfer, and normal mode transfer to the TPU's TGR in the second half. This is because clearing of the activation source and interrupt generation at the end of the specified number of transfers are restricted to the second half of the chain transfer (transfer when CHNE = 0).

1. Perform settings for transfer to the PPG's NDR. Set MRA to source address incrementing (SM1 = 1, SM0 = 0), fixed destination address (DM1 = DM0 = 0), repeat mode (MD1 = 0, MD0 = 1), and word size (Sz1 = 0, Sz0 = 1). Set the source side as a repeat area (DTS = 1). Set MRB to chain transfer mode (CHNE = 1, CHNS = 0, DISEL = 0). Set the data table start address in SAR, the NDRH address in DAR, and the data table size in CRAH and CRAL. CRB can be set to any value.
2. Perform settings for transfer to the TPU's TGR. Set MRA to source address incrementing (SM1 = 1, SM0 = 0), fixed destination address (DM1 = DM0 = 0), normal mode (MD1 = MD0 = 0), and word size (Sz1 = 0, Sz0 = 1). Set the data table start address in SAR, the TGRA address in DAR, and the data table size in CRA. CRB can be set to any value.
3. Locate the TPU transfer information consecutively after the NDR transfer information.
4. Specify the start address of the NDR transfer information to the DTC vector address.
5. Set the bit corresponding to the TGIA interrupt in DTCER to 1.
6. Set TGRA as an output compare register (output disabled) with TIOR, and enable the TGIA interrupt with TIER.
7. Set the initial output value in PODR, and the next output value in NDR. Set bits in DDR and NDER for which output is to be performed to 1. Using PCR, select the TPU compare match to be used as the output trigger.
8. Set the CST bit in TSTR to 1, and start the TCNT count operation.
9. Each time a TGRA compare match occurs, the next output value is transferred to NDR and the set value of the next output trigger period is transferred to TGRA. The activation source TGFA flag is cleared.
10. When the specified number of transfers is completed (the TPU transfer CRA value is 0), the TGFA flag is held at 1, the DTCE bit is cleared to 0, and a TGIA interrupt request is sent to the CPU. Termination processing should be performed in the interrupt handling routine.

8.7.3 Chain Transfer when Counter = 0

By executing a second data transfer and performing re-setting of the first data transfer only when the counter value is 0, it is possible to perform 256 or more repeat transfers.

An example is shown in which a 128-kbyte input buffer is configured. The input buffer is assumed to have been set to start at lower address H'0000. Figure 8.16 shows the chain transfer when the counter value is 0.

1. For the first transfer, set the normal transfer mode for input data. Set the fixed transfer source address, CRA = H'0000 (65,536 times), CHNE = 1, CHNS = 1, and DISEL = 0.
2. Prepare the upper 8-bit addresses of the start addresses for 65,536-transfer units for the first data transfer in a separate area (in ROM, etc.). For example, if the input buffer is configured at addresses H'200000 to H'21FFFF, prepare H'21 and H'20.
3. For the second transfer, set repeat transfer mode (with the source side as the repeat area) for re-setting the transfer destination address for the first data transfer. Use the upper eight bits of DAR in the first transfer information area as the transfer destination. Set CHNE = DISEL = 0. If the above input buffer is specified as H'200000 to H'21FFFF, set the transfer counter to 2.
4. Execute the first data transfer 65536 times by means of interrupts. When the transfer counter for the first data transfer reaches 0, the second data transfer is started. Set the upper eight bits of the transfer source address for the first data transfer to H'21. The lower 16 bits of the transfer destination address of the first data transfer and the transfer counter are H'0000.
5. Next, execute the first data transfer the 65536 times specified for the first data transfer by means of interrupts. When the transfer counter for the first data transfer reaches 0, the second data transfer is started. Set the upper eight bits of the transfer source address for the first data transfer to H'20. The lower 16 bits of the transfer destination address of the first data transfer and the transfer counter are H'0000.
6. Steps 4 and 5 are repeated endlessly. As repeat mode is specified for the second data transfer, no interrupt request is sent to the CPU.

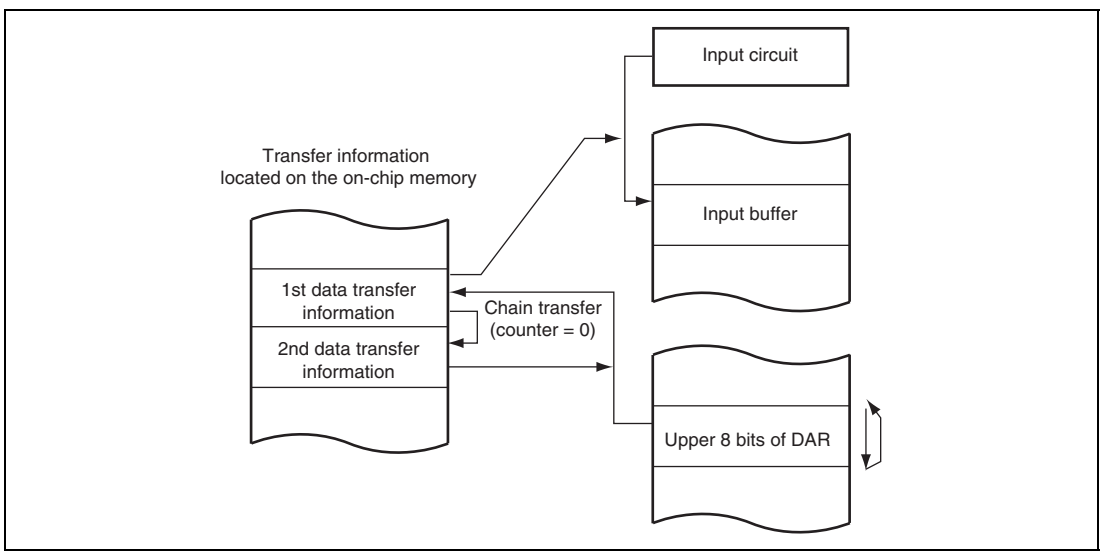


Figure 8.16 Chain Transfer when Counter = 0

8.8 Interrupt Sources

An interrupt request is issued to the CPU when the DTC finishes the specified number of data transfers or a data transfer for which the DISEL bit was set to 1. In the case of interrupt activation, the interrupt set as the activation source is generated. These interrupts to the CPU are subject to CPU mask level and priority level control in the interrupt controller.

8.9 Usage Notes

8.9.1 Module Stop State Setting

Operation of the DTC can be disabled or enabled using the module stop control register. The initial setting is for operation of the DTC to be enabled. Register access is disabled by setting the module stop state. The module stop state cannot be set while the DTC is activated. For details, refer to section 23, Power-Down Modes.

8.9.2 On-Chip RAM

Transfer information can be located in on-chip RAM. In this case, the RAME bit in SYSCR0 must not be cleared to 0.

8.9.3 DMAC Transfer End Interrupt

When the DTC is activated by a DMAC transfer end interrupt, the DTE bit of DMDR is not controlled by the DTC but its value is modified with the write data regardless of the transfer counter value and DISEL bit setting. Accordingly, even if the DTC transfer counter value becomes 0, no interrupt request may be sent to the CPU in some cases.

8.9.4 DTCE Bit Setting

For DTCE bit setting, use bit manipulation instructions such as BSET and BCLR. If all interrupts are disabled, multiple activation sources can be set at one time (only at the initial setting) by writing data after executing a dummy read on the relevant register.

8.9.5 Chain Transfer

When chain transfer is used, clearing of the activation source or DTCER is performed when the last of the chain of data transfers is executed. At this time, SCI and A/D converter interrupt/activation sources are cleared when the DTC reads or writes to the relevant register.

Therefore, when the DTC is activated by an interrupt or activation source, if a read/write of the relevant register is not included in the last chained data transfer, the interrupt or activation source will be retained.

8.9.6 Transfer Information Start Address, Source Address, and Destination Address

The transfer information start address to be specified in the vector table should be address 4n. If an address other than address 4n is specified, the lower 2 bits of the address are regarded as 0s.

The source and destination addresses specified in SAR and DAR, respectively, will be transferred in the divided bus cycles depending on the address and data size.

8.9.7 Transfer Information Modification

When IBCCS = 1 while using the DMAC, clear the IBCCS bit to 0 and then set to 1 again before modifying the DTC transfer information in the CPU exception handling routine initiated by a DTC transfer end interrupt.

8.9.8 Endian Format

Since the DTC only supports big endian, transfer information must be written in big endian format.

Section 9 I/O Ports

Table 9.1 summarizes the port functions. The pins of each port also have other functions such as input/output pins of on-chip peripheral modules or interrupt input pins. Each I/O port includes a data direction register (DDR) that controls input/output, a data register (DR) that stores output data, a port register (PORT) used to read the pin states, and an input buffer control register (ICR) that controls input buffer on/off. Ports 4 and 5 do not have a DR or a DDR register.

Ports D, H, J, and K have internal input pull-up MOSs and a pull-up MOS control register (PCR) that controls the on/off state of the input pull-up MOSs.

Ports 1, 6, A, D, and H have internal pin pull-down MOSs and a pin state setting register (PSR) that controls the on/off state of the pin pull-down MOSs.

Ports 1, 6, A, D, and H have a driving ability switching function and a port driving ability setting register (DSR) that controls the high or low of the output driving ability.

Ports 1, 3, 6, A, D, H, J, and K can drive a single TTL load and capacitive loads up to 30 pF.

All of the I/O ports can drive Darlington transistors when functioning as output ports.

Schmitt-trigger inputs are enabled when a port is used as the $\overline{\text{IRQ}}$ and TPU inputs.

Table 9.1 Port Functions

Port	Description	Bit	Function			Schmitt-Trigger Input*	Input Pull-up MOS Function	Open-Drain Output Function
			I/O	Input	Output			
Port 1	General I/O port also functioning as interrupt inputs, SCI I/Os, A/D converter inputs, TPU inputs, and RSPI I/Os	7	P17	$\overline{\text{IRQ7}}$-A/$\overline{\text{ADTRG1}}$/ TCKLD-B	SSLD1	$\overline{\text{IRQ7}}$-A/TCLKD-B	—	When RSPI function is used
		6	P16/SCK3	$\overline{\text{IRQ6}}$-A/TCLKC-B	—	$\overline{\text{IRQ6}}$-A/TCLKC-B		
		5	P15	$\overline{\text{IRQ5}}$-A/ TCLKB-B/RxD3	—	$\overline{\text{IRQ5}}$-A/TCLKB-B		
		4	P14	$\overline{\text{IRQ4}}$-A/TCLKA-B	TxD3	$\overline{\text{IRQ4}}$-A/TCLKA-B		
		3	P13/SSLD0	$\overline{\text{IRQ3}}$-A/$\overline{\text{ADTRG0}}$	—	$\overline{\text{IRQ3}}$-A		
		2	P12/RSPCKD	$\overline{\text{IRQ2}}$-A	—	$\overline{\text{IRQ2}}$-A		
		1	P11/MISOD	$\overline{\text{IRQ1}}$-A	—	$\overline{\text{IRQ1}}$-A		
		0	P10/MOSID	$\overline{\text{IRQ0}}$	—	$\overline{\text{IRQ0}}$-A		
Port 3	General I/O port also functioning as PPG outputs and TPU I/Os	7	P37/TIOCB2	TIOCA2/TCLKD-A	PO15	All inputs	—	—
		6	P36/TIOCA2	—	PO14	All inputs		
		5	P35/TIOCB1	TIOCA1/TCLKC-A	PO13	All inputs		
		4	P34/TIOCA1	—	PO12	All inputs		
		3	P33/TIOCD0	TIOCC0/TCLKB-A/ —	PO11	P33/TIOCD0/ TIOCC0/TCLKB-A		
		2	P32/TIOCC0	TCLKA-A	PO10	P32/TIOCC0/ TCLKA-A		
		1	P31/TIOCB0	TIOCA0	PO9	P31/TIOCB0/ TIOCA0		
		0	P30/TIOCA0	—	PO8	P30/TIOCA0		
Port 4	General input port also functioning as A/D converter inputs	7	—	P47/AN11	—	—	—	—
		6	—	P46/AN10	—			
		5	—	P45/AN9	—			
		4	—	P44/AN8	—			
		3	—	P43/AN15	—			
		2	—	P42/AN14	—			
		1	—	P41/AN13	—			
		0	—	P40/AN12	—			

Port	Description	Bit	Function			Schmitt-Trigger Input*	Input Pull-up MOS Function	Open-Drain Output Function
			I/O	Input	Output			
Port 5	General input port also functioning as interrupt inputs and A/D converter inputs	7	—	P57/$\overline{\text{IRQ7}}$-B/AN7	—	$\overline{\text{IRQ7}}$-B	—	—
		6	—	P56/$\overline{\text{IRQ6}}$-B/AN6	—	$\overline{\text{IRQ6}}$-B		
		5	—	P55/$\overline{\text{IRQ5}}$-B/AN5	—	$\overline{\text{IRQ5}}$-B		
		4	—	P54/$\overline{\text{IRQ4}}$-B/AN4	—	$\overline{\text{IRQ4}}$-B		
		3	—	P53/IRQ3-B/AN3	—	$\overline{\text{IRQ3}}$-B		
		2	—	P52/$\overline{\text{IRQ2}}$-B/AN2	—	$\overline{\text{IRQ2}}$-B		
		1	—	P51/$\overline{\text{IRQ1}}$-B/AN1	—	$\overline{\text{IRQ1}}$-B		
		0	—	P50/$\overline{\text{IRQ0}}$-B/AN0	—	$\overline{\text{IRQ0}}$-B		
Port 6	General I/O port also functioning as SCI I/Os, RCAN I/Os, and interrupt inputs	6	P66	$\overline{\text{IRQ14}}$/CRx_1	—	$\overline{\text{IRQ14}}$	—	—
		5	P65	$\overline{\text{IRQ13}}$/CRx_0-A/ (CRx_0, CRx_1)-A	CTx_0-B/ (CTx_0 or CTx_1)-B	$\overline{\text{IRQ13}}$		
		4	P64	$\overline{\text{IRQ12}}$/CRx_0-B (CRx_0, CRx_1)-B	CTx_0-A/ (CTx_0 or CTx_1)-A	$\overline{\text{IRQ12}}$		
		3	P63	$\overline{\text{IRQ11}}$	—	$\overline{\text{IRQ11}}$		
		2	P62/SCK4	$\overline{\text{IRQ10}}$	CTx_1	$\overline{\text{IRQ10}}$		
		1	P61	RxD4/$\overline{\text{IRQ9}}$		$\overline{\text{IRQ9}}$		
		0	P60	$\overline{\text{IRQ8}}$	TxD4	$\overline{\text{IRQ8}}$		
Port A	General I/O port also functioning as RSPI I/Os and Bϕ output	7	—	PA7	Bϕ/SSCL3	—	—	—
		6	PA6	—	SSLC2			
		5	PA5	—	SSLC1			
		4	PA4/SSLC0	—	—			
		3	PA3/MOSIC	—	—			
		2	PA2/MISOC	—	—			
		1	PA1/RSPCKC	—	—			

Port	Description	Bit	Function I/O	Function Input	Function Output	Schmitt-Trigger Input*	Input Pull-up MOS Function	Open-Drain Output Function
Port D	General I/O port also functioning as RSPI I/Os	7	PD7/SSLB0	—	—	—	O	Only when RSPI function is used
		6	PD6/RSPCKB	—	—			
		5	PD5/MISOB	—	—			
		4	PD4/MOSIB	—	—			
		3	PD3/SSLA0	—	—			
		2	PD2/RSOCKA	—	—			
		1	PD1/MISOA	—	—			
		0	PD0/MOSIA	—	—			
Port H	General I/O port also functioning as RSPI outputs	7	PH7	—	SSLD3	—	O	Only when RSPI function is used
		6	PH6	—	SSLD2			
		5	PH5	—	SSLB3			
		4	PH4	—	SSLB2			
		3	PH3	—	SSLB1			
		2	PH2	—	SSLA3			
		1	PH1	—	SSLA2			
		0	PH0	—	SSLA1			
Port J	General I/O port also functioning as TPU I/Os	7	PJ7/TIOCB8	TIOCA8/TCLKH	—	PJ7, TIOCA8/ TIOCB8/TCLKH	O	—
		6	PJ6/TIOCA8	—	—	PJ6, TIOCA8		
		5	PJ5/TIOCB7	TIOCA7/TCLKG	—	PJ5, TIOCA7/ TIOCB7-TCLKG		
		4	PJ4/TIOCA7	—	—	PJ4, TIOCA7		
		3	PJ3/TIOCD6	TIOCC6/TCLKF	—	PJ3, TIOCC6/ TIOCD6/TCLKF		
		2	PJ2/TIOCC6	TCLKE	—	PJ2, TIOCC6/ TCLKE		
		1	PJ1/TIOCB6	TIOCA6	—	PJ1, TIOCA6/ TIOCB6		
		0	PJ0/TIOCA6	—	—	PJ0, TIOCA6		

Port	Description	Bit	Function			Schmitt-Trigger Input*	Input Pull-up MOS Function	Open-Drain Output Function
			I/O	Input	Output			
Port K	General I/O port also functioning as TPU I/Os	7	PK7/TIOCB11	TIOCA11	—	PK7, TIOCA11/ TIOCB11	O	—
		6	PK6/TIOCA11	—	—	PK6, TIOCA11		
		5	PK5/TIOCB10	TIOCA10	—	PK5, TIOCA10/ TIOCB10		
		4	PK4/TIOCA10	—	—	PK4, TIOCA10		
		3	PK3/TIOCD9	TIOCC9	—	PK3, TIOCD9/ TIOCC9		
		2	PK2/TIOCC9	—	—	PK2, TIOCC9		
		1	PK1/TIOCB9	TIOCA9	—	PK1, TIOCA9/ TIOCB9		
		0	PK0/TIOCA9	—	—	PK0, TIOCA9		

Note: * Pins other than Schmitt-trigger input pin are CMOS input pins.

9.1 Register Descriptions

Table 9.2 lists each port registers.

Table 9.2 Register Configuration in Each Port

Port	Number of Pins	Registers						
		DDR	DR	PORT	ICR	PCR	DSR	PSR
Port 1	8	O	O	O	O	—	O	O
Port 3	8	O	O	O	O	—	—	—
Port 4	8	—	—	O	O	—	—	—
Port 5	8	—	—	O	O	—	—	—
Port 6*[1]	7	O	O	O	O	—	O	O
Port A*[2]	7	O	O	O	O	—	O	O
Port D	8	O	O	O	O	O	O	O
Port H	8	O	O	O	O	O	O	O
Port J	8	O	O	O	O	O	—	—
Port K	8	O	O	O	O	O	—	—

[Legend]

O: Register exists

—: No register exists

Notes: 1. The lower seven bits are valid and the upper one bit is reserved. The write value should always be the initial value.

2. The upper seven bits are valid and the lower one bit is reserved. The write value should always be the initial value.

Figure 9.1 shows a port block diagram.

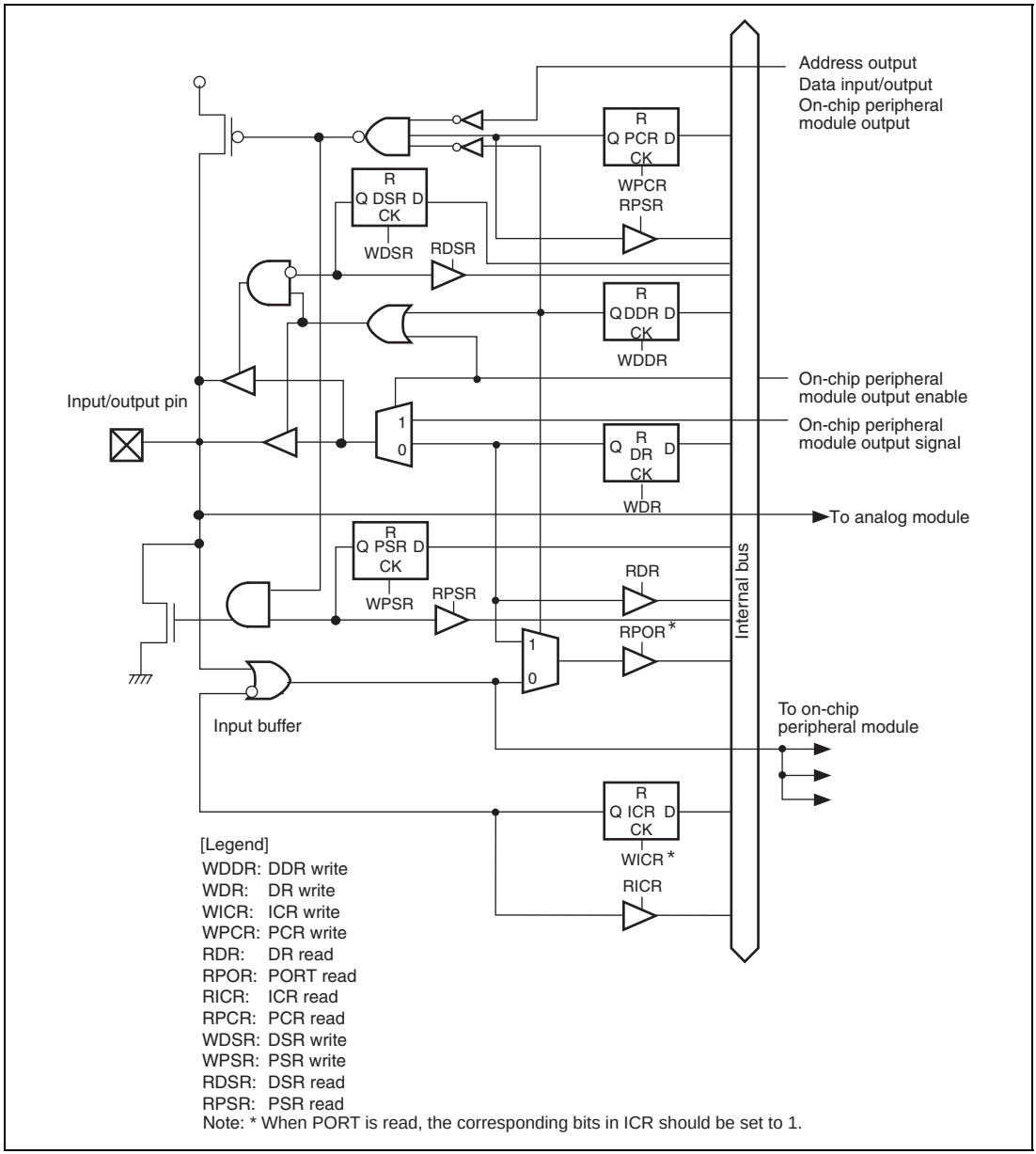


Figure 9.1 I/O Port Block Diagram

9.1.1 Data Direction Register (PnDDR) (n = 1, 3, 6, A, D, H, J, and K)

DDR is an 8-bit write-only register that specifies the port input or output for each bit. A read from DDR is invalid and DDR is always read as an undefined value. As DDR is a write-only register, it should not be written with the bit-manipulation instruction.

When the general I/O port function is selected, the corresponding pin functions as an output port by setting the corresponding DDR bit to 1; the corresponding pin functions as an input port by clearing the corresponding DDR bit to 0.

Bit	7	6	5	4	3	2	1	0
Bit Name	Pn7DDR	Pn6DDR	Pn5DDR	Pn4DDR	Pn3DDR	Pn2DDR	Pn1DDR	Pn0DDR
Initial Value:	0	0	0	0	0	0	0	0
R/W:	W	W	W	W	W	W	W	W

Note: The lower seven bits are valid and the upper one bit is reserved for port 6 data direction register (P6DDR).
The upper seven bits are valid and the lower one bit is reserved for port A data direction register (PADDR).

9.1.2 Data Register (PnDR) (n = 1, 3, 6, A, D, H, J, and K)

DR is an 8-bit readable/writable register that stores the output data of the pins to be used as the general output port.

The initial value of DR is H'00.

Bit	7	6	5	4	3	2	1	0
Bit Name	Pn7DR	Pn6DR	Pn5DR	Pn4DR	Pn3DR	Pn2DR	Pn1DR	Pn0DR
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: The lower seven bits are valid and the upper one bit is reserved for port 6 data register (P6DR).
The upper seven bits are valid and the lower one bit is reserved for port A data register (PADR).

9.1.3 Port Register (PORTn) (n = 1, 3, 4, 5, 6, A, D, H, J, and K)

PORT is an 8-bit read-only register that reflects the port pin status. A write to PORT is invalid. When PORT is read, the DR bits that correspond to the respective DDR bits set to 1.

The initial value of PORT is undefined and is determined based on the port pin status.

Bit	7	6	5	4	3	2	1	0
Bit Name	Pn7	Pn6	Pn5	Pn4	Pn3	Pn2	Pn1	Pn0
Initial Value:	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
R/W:	R	R	R	R	R	R	R	R

Note: The lower seven bits are valid and the upper one bit is reserved for port 6 register (PORT6).
The upper seven bits are valid and the lower one bit is reserved for port A register (PORTA).

9.1.4 Input Buffer Control Register (PnICR) (n = 1, 3, 4, 5, 6, A, D, H, J, and K)

ICR is an 8-bit readable/writable register that controls the port input buffers.

For bits in ICR set to 1, the input buffers of the corresponding pins are valid. For bits in ICR cleared to 0, the input buffers of the corresponding pins are invalid and the input signals are fixed high.

When the pin functions as an input for the peripheral modules or when PORT is read, the corresponding bits should be set to 1. The initial value should be written to a bit whose corresponding pin is not used as an input or is used as an analog input/output pin.

If ICR is modified, an internal edge may occur depending on the pin status. Accordingly, ICR should be modified when the corresponding input pins are not used. For example, in $\overline{\text{IRQ}}$ input, modify ICR while the corresponding interrupt is disabled, clear the IRQF flag in ISR of the interrupt controller to 0, and then enable the corresponding interrupt. If an edge occurs after the ICR setting, the edge should be cancelled.

The initial value of ICR is H'00.

Bit	7	6	5	4	3	2	1	0
Bit Name	Pn7ICR	Pn6ICR	Pn5ICR	Pn4ICR	Pn3ICR	Pn2ICR	Pn1ICR	Pn0ICR
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: The lower seven bits are valid and the upper one bit is reserved for port 6 input buffer control register (P6ICR).
The upper seven bits are valid and the lower one bit is reserved for port A input buffer control register (PAICR).

9.1.5 Pull-Up MOS Control Register (PnPCR) (n = D, H, J, and K)

PCR is an 8-bit readable/writable register that controls on/off of the port input pull-up MOS.

If a bit in PCR is set to 1 while the pin is in input state, the input pull-up MOS corresponding to the bit in PCR is turned on. Table 9.3 shows the input pull-up MOS status.

The initial value of PCR is H'00.

Bit	7	6	5	4	3	2	1	0
Bit Name	Pn7PCR	Pn6PCR	Pn5PCR	Pn4PCR	Pn3PCR	Pn2PCR	Pn1PCR	Pn0PCR
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 9.3 Input Pull-Up MOS State

Port	Pin State	Reset	Software Standby Mode	Other Operations
Port D	Peripheral module output	OFF		
	Port input	OFF	ON/OFF	
Port H	Peripheral module output	OFF		
	Port input	OFF	ON/OFF	
Port J	Peripheral module output	OFF		
	Port input	OFF	ON/OFF	
Port K	Peripheral module output	OFF		
	Port input	OFF	ON/OFF	

[Legend]

OFF: The input pull-up MOS is always off.

ON/OFF: If PCR is set to 1, the input pull-up MOS is on; if PCR is cleared to 0, the input pull-up MOS is off.

9.1.6 Driving Ability Setting Register (DSR) (n = 1, 6, A, D, and H)

DSR is an 8-bit readable/writable register that selects the driving ability for ports.

Setting the specific bits of DSR to 1 decreases the driving ability of their corresponding pins.

Bit	7	6	5	4	3	2	1	0
Bit Name	Pn7DSR	Pn6DSR	Pn5DSR	Pn4DSR	Pn3DSR	Pn2DSR	Pn1DSR	Pn0DSR
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

9.1.7 Pin State Setting Register (PSR) (n = 1, 6, A, D, and H)

PSR is an 8-bit readable/writable register that enables or disables the pull-down state.

Setting the specific bits of PSR to 1 puts their corresponding pins into pull-down state.

Bit	7	6	5	4	3	2	1	0
Bit Name	Pn7PSR	Pn6PSR	Pn5PSR	Pn4PSR	Pn3PSR	Pn2PSR	Pn1PSR	Pn0PSR
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

9.2 Output Buffer Control

This section describes the output priority of each pin.

The name of each peripheral module pin is followed by "_OE". This (for example: MIOCA4_OE) indicates whether the output of the corresponding function is valid (1) or if another setting is specified (0). Table 9.4 lists each port output signal's valid setting. For details on the corresponding output signals, see the register description of each peripheral module. If the name of each peripheral module pin is followed by A or B, the pin function can be modified by the port function control register (PFCR). For details, see section 9.3, Port Function Controller.

9.2.1 Port 1

(1) P17/$\overline{\text{IRQ7}}$-A/$\overline{\text{ADTRG1}}$/TCLKD-B/SSLD1

The pin function is switched as shown below according to the combination of the RSPI register setting and the P17DDR bit.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	SSLD1_OE	P17DDR
RSPI	SSLD1 output	1	—
I/O port	P17 output	0	1
	P17 input (initial setting)	0	0

(2) P16/$\overline{\text{IRQ6}}$-A/TCLKC-B/SCK3

The pin function is switched as shown below according to the combination of the SCI_3 register setting and P16DDR bit.

		Setting	
		SCI_3	I/O Port
Module Name	Pin Function	SCK3_OE	P16DDR
SCI_3	SCK3 output	1	—
I/O port	P16 output	0	1
	P16 input (initial setting)	0	0

(3) P15/$\overline{\text{IRQ5}}$-A/TCLKB-B/RxD3

The pin function is switched as shown below according to the P15DDR bit setting.

		Setting
		I/O Port
Module Name	Pin Function	P15DDR
I/O port	P15 output	1
	P15 input (initial setting)	0

(4) P14/$\overline{\text{IRQ4}}$-A/TCLKA-B/TxD3

The pin function is switched as shown below according to the combination of the SCI_3 register setting and P14DDR bit.

		Setting	
		SCI_3	I/O Port
Module Name	Pin Function	TxD3_OE	P14DDR
SCI_3	TxD3 output	1	—
I/O port	P14 output	0	1
	P14 input (initial setting)	0	0

(5) P13/$\overline{\text{IRQ3}}$-A/$\overline{\text{ADTRG0}}$/SSLD0

The pin function is switched as shown below according to the combination of the RSPI register setting and the P13DDR bit.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	SSLD0_OE	P13DDR
RSPI	SSLD0 output	1	—
I/O port	P13 output	0	1
	P13 input (initial setting)	0	0

(6) P12/$\overline{\text{IRQ2}}$-A/RSPCKD

The pin function is switched as shown below according to the combination of the RSPI register setting and the P12DDR bit.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	RSPCKD_OE	P12DDR
RSPI	RSPCKD output	1	—
I/O port	P12 output	0	1
	P12 input (initial setting)	0	0

(7) P11/$\overline{\text{IRQ1}}$-AMISOD

The pin function is switched as shown below according to the combination of the RSPI register setting and the P11DDR bit.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	MISOD_OE	P11DDR
RSPI	MISOD output	1	—
I/O port	P11 output	0	1
	P11 input (initial setting)	0	0

(8) P10/$\overline{\text{IRQ0}}$-A/MOSID

The pin function is switched as shown below according to the combination of the RSPI register setting and the P10DDR bit.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	MOSID_OE	P10DDR
RSPI	MOSID output	1	—
I/O port	P10 output	0	1
	P10 input (initial setting)	0	0

9.2.2 Port 3

(1) P37/TIOCB2/TIOCA2/TCLKD-A/PO15

The pin function is switched as shown below according to the combination of the port function control register 9 (PFCR9), TPU_2 register, PPG register, and P37DDR bit settings.

		Setting		
		TPU_2	PPG	I/O Port
Module Name	Pin Function	TIOCB2_OE	PO15_OE	P37DDR
TPU_2	TIOCB2 output	1	—	—
PPG	PO15 output	0	1	—
I/O port	P37 output	0	0	1
	P37 input (initial setting)	0	0	0

(2) P36/TIOCA2/PO14

The pin function is switched as shown below according to the combination of the port function control register 9 (PFCR9), TPU_2 register, PPG register, and P36DDR bit settings.

		Setting		
		TPU_2	PPG	I/O Port
Module Name	Pin Function	TIOCA2_OE	PO14_OE	P36DDR
TPU_2	TIOCA2 output	1	—	—
PPG	PO14 output	0	1	—
I/O port	P36 output	0	0	1
	P36 input (initial setting)	0	0	0

(3) P35/TIOCB1/TIOCA1/TCLKC-A/PO13

The pin function is switched as shown below according to the combination of the port function control register 9 (PFCR9), TPU_1 register, PPG register, and P35DDR bit settings.

		Setting		
		TPU_1	PPG	I/O Port
Module Name	Pin Function	TIOCB1_OE	PO13_OE	P35DDR
TPU_1	TIOCB1 output	1	—	—
PPG	PO13 output	0	1	—
I/O port	P35 output	0	0	1
	P35 input (initial setting)	0	0	0

(4) P34/TIOCA1/PO12

The pin function is switched as shown below according to the combination of the port function control register 9 (PFCR9), TPU_1 register, PPG register, and P34DDR bit settings.

		Setting		
		TPU_1	PPG	I/O Port
Module Name	Pin Function	TIOCA1_OE	PO12_OE	P34DDR
TPU_1	TIOCA1 output	1	—	—
PPG	PO12 output	0	1	—
I/O port	P34 output	0	0	1
	P34 input (initial setting)	0	0	0

(5) P33/TIOCD0/TIOCC0/TCLKB-A/PO11

The pin function is switched as shown below according to the combination of the port function control register 9 (PFCR9), TPU_0 register, PPG register, and P33DDR bit settings.

		Setting		
		TPU_0	PPG	I/O Port
Module Name	Pin Function	TIOCD0_OE	PO11_OE	P33DDR
TPU_0	TIOCD0 output	1	—	—
PPG	PO11 output	0	1	—
I/O port	P33 output	0	0	1
	P33 input (initial setting)	0	0	0

(6) P32/TIOCC0/TCLKA-A/PO10

The pin function is switched as shown below according to the combination of the port function control register 9 (PFCR9), TPU_0 register, PPG register, and P32DDR bit settings.

		Setting		
		TPU_0	PPG	I/O Port
Module Name	Pin Function	TIOCC0_OE	PO10_OE	P32DDR
TPU_0	TIOCC0 output	1	—	—
PPG	PO10 output	0	1	—
I/O port	P32 output	0	0	1
	P32 input (initial setting)	0	0	0

(7) P31/TIOCB0/TIOCA0/PO9

The pin function is switched as shown below according to the combination of the port function control register 9 (PFCR9), TPU_0 register, PPG register, and P31DDR bit settings.

		Setting		
		TPU_0	PPG	I/O Port
Module Name	Pin Function	TIOCB0_OE	PO9_OE	P31DDR
TPU_0	TIOCB0 output	1	—	—
PPG	PO9 output	0	1	—
I/O port	P31 output	0	0	1
	P31 input (initial setting)	0	0	0

(8) P30/TIOCA0/PO8

The pin function is switched as shown below according to the combination of the port function control register 9 (PFCR9), TPU_0 register, PPG register, and P30DDR bit settings.

		Setting		
		TPU_0	PPG	I/O Port
Module Name	Pin Function	TIOCA0_OE	PO8_OE	P30DDR
TPU_0	TIOCA0 output	1	—	—
PPG	PO8 output	0	1	—
I/O port	P30 output	0	0	1
	P30 input (initial setting)	0	0	0

9.2.3 Port 6

(1) P66/CRx_1/$\overline{\text{IRQ14}}$

The pin function is switched as shown below according to the P66DDR bit setting.

		Setting
		I/O Port
Module Name	Pin Function	P66DDR
I/O port	P66 output	1
	P66 input (initial setting)	0

(2) P65/CRx_0-A/(CRx_0, CRx_1)-A/CTx_0-B/(CTx_0 or CTx_1)-B/$\overline{\text{IRQ13}}$

The pin function is switched as shown below according to the combination of the port function control register 5 (PFCR5), RCAN register, and the P65DDR bit setting.

		Setting		
		RCAN		I/O Port
Module Name	Pin Function	CTx_0-B_OE	(CTx_0 or CTx_1)-B_OE	P65DDR
RCAN	CTx_0-B output	1	0	—
	(CTx_0 or CTx_1)-B output	0	1	—
I/O port	P65 output	0	0	1
	P65 input (initial setting)	0	0	0

(3) P64/CTx_0-A/(CTx_0 or CTx_1)-A/CRx_0-B/(CRx_0, CRx_1)-B/$\overline{\text{IRQ12}}$

The pin function is switched as shown below according to the combination of the port function control register 5 (PFCR5), RCAN register, and P64DDR bit settings.

		Setting		
		RCAN		I/O Port
Module Name	Pin Function	CTx_0-A_OE	(CTx_0 or CTx_1)-A_OE	P64DDR
RCAN	CTx_0-A output	1	0	—
	(CTx_0 or CTx_1)-A output	0	1	—
I/O port	P64 output	0	0	1
	P64 input (initial setting)	0	0	0

(4) P63/$\overline{\text{IRQ11}}$

The pin function is switched as shown below according to the P63DDR bit setting.

		Setting
		I/O Port
Module Name	Pin Function	P63DDR
I/O port	P63 output	1
	P63 input (initial setting)	0

(5) P62/CTx_1/SCK4/$\overline{\text{IRQ10}}$

The pin function is switched as shown below according to the combination of the port function control register 5 (PFCR5), RCAN register, SCI_4 register, and P62DDR bit settings.

		Setting		
		RCAN	SCI_4	I/O Port
Module Name	Pin Function	CTx_1_OE	SCK4_OE	P62DDR
RCAN	CTx_1 output	1	—	—
SCI_4	SCK4 output	0	1	—
I/O port	P62 output	0	0	1
	P62 input (initial setting)	0	0	0

(6) P61/RxD4/$\overline{\text{IRQ9}}$

The pin function is switched as shown below according to the P61DDR bit setting.

		Setting
		I/O Port
Module Name	Pin Function	P61DDR
I/O port	P61 output	1
	P61 input (initial setting)	0

(7) P60/TxD4/$\overline{\text{IRQ8}}$

The pin function is switched as shown below according to the combination of the SCI_4 register and P60DDR bit settings.

		Setting	
		SCI_4	I/O Port
Module Name	Pin Function	TxD4_OE	P60DDR
SCI_4	TxD4 output	1	—
I/O port	P60 output	0	1
	P60 input (initial setting)	0	0

9.2.4 Port A

(1) PA7/Bϕ/SSLC3

The pin function is switched as shown below according to the RSPI register and PA7DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	SSLC3_OE	PA7DDR
RSPI	SSLC3 output	1	—
I/O port	B_ϕ output	0	1
	PA7 input (initial setting)	0	0

(2) PA6/SSLC2

The pin function is switched as shown below according to the RSPI register and the PA6DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	SSLC2_OE	PA6DDR
RSPI	SSLC2 output	1	—
I/O port	PA6 output	0	1
	PA6 input (initial setting)	0	0

(3) PA5/SSLC1

The pin function is switched as shown below according to the RSPI register and the PA5DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	SSLC1_OE	PA5DDR
RSPI	SSLC1 output	1	—
I/O port	PA5 output	0	1
	PA5 input (initial setting)	0	0

(4) PA4/SSLC0

The pin function is switched as shown below according to the RSPI register and the PA4DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	SSLC0_OE	PA4DDR
RSPI	SSLC0 output	1	—
I/O port	PA4 output	0	1
	PA4 input (initial setting)	0	0

(5) PA3/MOSIC

The pin function is switched as shown below according to the combination of the RSPI register and the PA3DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	MOSIC_OE	PA3DDR
RSPI	MOSIC output	1	—
I/O port	PA3 output	0	1
	PA3 input (initial setting)	0	0

(6) PA2/MISOC

The pin function is switched as shown below according to the combination of the RSPI register and the PA2DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	MISOC_OE	PA2DDR
RSPI	MISOC output	1	—
I/O port	PA2 output	0	1
	PA2 input (initial setting)	0	0

(7) PA1/RSPCKC

The pin function is switched as shown below according to the combination of the RSPI register and the PA1DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	RSPCKC_OE	PA1DDR
RSPI	RSPCKC output	1	—
I/O port	PA1 output	0	1
	PA1 input (initial setting)	0	0

9.2.5 Port D

(1) PD7/SSLB0

The pin function is switched as shown below according to the combination of the RSPI register and the PD7DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	SSLB0_OE	PD7DDR
RSPI	SSLB0 output	1	—
I/O port	PD7 output	0	1
	PD7 input (initial setting)	0	0

(2) PD6/RSPCKB

The pin function is switched as shown below according to the combination of the RSPI register and the PD6DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	RSPCKB_OE	PD6DDR
RSPI	RSPCKB output	1	—
I/O port	PD6 output	0	1
	PD6 input (initial setting)	0	0

(3) PD5/MISOB

The pin function is switched as shown below according to the combination of the RSPI register and the PD5DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	MISOB_OE	PD5DDR
RSPI	MISOB output	1	—
I/O port	PD5 output	0	1
	PD5 input (initial setting)	0	0

(4) PD4/MOSIB

The pin function is switched as shown below according to the combination of the RSPI register and the PD4DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	MOSIB_OE	PD4DDR
RSPI	MOSIB output	1	—
I/O port	PD4 output	0	1
	PD4 input (initial setting)	0	0

(5) PD3/SSLA0

The pin function is switched as shown below according to the combination of the RSPI register and the PD3DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	SSLA0_OE	PD3DDR
RSPI	SSLA0 output	1	—
I/O port	PD3 output	0	1
	PD3 input (initial setting)	0	0

(6) PD2/RSPCKA

The pin function is switched as shown below according to the combination of the RSPI register and the PD2DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	RSPCKA_OE	PD2DDR
RSPI	RSPCKA output	1	—
I/O port	PD2 output	0	1
	PD2 input (initial setting)	0	0

(7) PD1/MISOA

The pin function is switched as shown below according to the combination of the RSPI register and the PD1DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	MISOA_OE	PD1DDR
RSPI	MISOA output	1	—
I/O port	PD1 output	0	1
	PD1 input (initial setting)	0	0

(8) PD0/MOSIA

The pin function is switched as shown below according to the combination of the RSPI register and the PD0DDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	MOSIA_OE	PD0DDR
RSPI	MOSIA output	1	—
I/O port	PD0 output	0	1
	PD0 input (initial setting)	0	0

9.2.6 Port H

(1) PH7/SSLD3, PH6/SSLD2, PH5/SSLB3, PH4/SSLB2, PH3/SSLB1, PH2/SSLA3, PH1/SSLA2, PH0/SSLA1

Port H is switched as shown below according to the RSPI register and the PHnDDR bit settings.

		Setting	
		RSPI	I/O Port
Module Name	Pin Function	SSLxx_OE	PHnDDR
RSPI	SSLxx output	1	—
I/O port	PHn output	0	1
	PHn input (initial setting)	0	0

[Legend]
n: 7 to 0

9.2.7 Port J

(1) PJ7/TIOCA8/TIOCB8/TCLKH

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_8 register, and PJ7DDR bit settings.

		Setting	
		TPU_8	I/O Port
Module Name	Pin Function	TIOCB8_OE	PJ7DDR
TPU_8	TIOCB8 output	1	—
I/O port	PJ7 output	0	1
	PJ7 input (initial setting)	0	0

(2) PJ6/TIOCA8

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_8 register, and PJ6DDR bit settings.

		Setting	
		TPU_8	I/O Port
Module Name	Pin Function	TIOCA8_OE	PJ6DDR
TPU_8	TIOCA8 output	1	—
I/O port	PJ6 output	0	1
	PJ6 input (initial setting)	0	0

(3) PJ5/TIOCA7/TIOCB7/TCLKG

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_7 register, and PJ5DDR bit settings.

		Setting	
		TPU_7	I/O Port
Module Name	Pin Function	TIOCB7_OE	PJ5DDR
TPU_7	TIOCB7 output	1	—
I/O port	PJ5 output	0	1
	PJ5 input (initial setting)	0	0

(4) PJ4/TIOCA7

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_7 register, and PJ4DDR bit settings.

		Setting	
		TPU_7	I/O Port
Module Name	Pin Function	TIOCA7_OE	PJ4DDR
TPU_7	TIOCA7 output	1	—
I/O port	PJ4 output	0	1
	PJ4 input (initial setting)	0	0

(5) PJ3/TIOCC6/TIOCD6/TCLKF

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_6 register, and PJ3DDR bit settings.

		Setting	
		TPU_6	I/O Port
Module Name	Pin Function	TIOCD6_OE	PJ3DDR
TPU_6	TIOCD6 output	1	—
I/O port	PJ3 output	0	1
	PJ3 input (initial setting)	0	0

(6) PJ2/TIOCC6/TCLKE

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_6 register, and PJ2DDR bit settings.

		Setting	
		TPU_6	I/O Port
Module Name	Pin Function	TIOCC6_OE	PJ2DDR
TPU_6	TIOCC6 output	1	—
I/O port	PJ2 output	0	1
	PJ2 input (initial setting)	0	0

(7) PJ1/TIOCA6/TIOCB6

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_6 register, and PJ1DDR bit settings.

		Setting	
		TPU_6	I/O Port
Module Name	Pin Function	TIOCB6_OE	PJ1DDR
TPU_6	TIOCB6 output	1	—
I/O port	PJ1 output	0	1
	PJ1 input (initial setting)	0	0

(8) PJ0/TIOCA6

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_6 register, and PJ0DDR bit settings.

		Setting	
		TPU_6	I/O Port
Module Name	Pin Function	TIOCA6_OE	PJ0DDR
TPU_6	TIOCA6 output	1	—
I/O port	PJ0 output	0	1
	PJ0 input (initial setting)	0	0

9.2.8 Port K

(1) PK7/TIOCA11/TIOCB11

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_11 register, and PK7DDR bit settings.

		Setting	
		TPU_11	I/O Port
Module Name	Pin Function	TIOCB11_OE	PK7DDR
TPU_11	TIOCB11 output	1	—
I/O port	PK7 output	0	1
	PK7 input (initial setting)	0	0

(2) PK6/TIOCA11

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_11 register, and PK6DDR bit settings.

		Setting	
		TPU_11	I/O Port
Module Name	Pin Function	TIOCA11_OE	PK6DDR
TPU_11	TIOCA11 output	1	—
I/O port	PK6 output	0	1
	PK6 input (initial setting)	0	0

(3) PK5/TIOCA10/TIOCB10

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_10 register, and PK5DDR bit settings.

		Setting	
		TPU_10	I/O Port
Module Name	Pin Function	TIOCB10_OE	PK5DDR
TPU_10	TIOCB10 output	1	—
I/O port	PK5 output	0	1
	PK5 input (initial setting)	0	0

(4) PK4/TIOCA10

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_10, and PK4DDR bit settings.

		Setting	
		TPU_10	I/O Port
Module Name	Pin Function	TIOCA10_OE	PK4DDR
TPU_10	TIOCA10 output	1	—
I/O port	PK4 output	0	1
	PK4 input (initial setting)	0	0

(5) PK3/TIOCC9/TIOCD9

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_9 register, and PK3DDR bit settings.

		Setting	
		TPU_9	I/O Port
Module Name	Pin Function	TIOCD9_OE	PK3DDR
TPU_9	TIOCD9 output	1	—
I/O port	PK3 output	0	1
	PK3 input (initial setting)	0	0

(6) PK2/TIOCC9

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_9 register, and PK2DDR bit settings.

		Setting	
		TPU_9	I/O Port
Module Name	Pin Function	TIOCC9_OE	PK2DDR
TPU_9	TIOCC9 output	1	—
I/O port	PK2 output	0	1
	PK2 input (initial setting)	0	0

(7) PK1/TIOCA9/TIOCB9

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_9 register, and PK1DDR bit settings.

		Setting	
		TPU_9	I/O Port
Module Name	Pin Function	TIOCB9_OE	PK1DDR
TPU_9	TIOCB9 output	1	—
I/O port	PK1 output	0	1
	PK1 input (initial setting)	0	0

(8) PK0/TIOCA9

The pin function is switched as shown below according to the combination of the port function control register A (PFCRA), TPU_9 register, and PK0DDR bit settings.

		Setting	
		TPU_9	I/O Port
Module Name	Pin Function	TIOCA9_OE	PK0DDR
TPU_9	TIOCA9 output	1	—
I/O port	PK0 output	0	1
	PK0 input (initial setting)	0	0

Table 9.4 Available Output Signals and Settings in Each Port

Port		Output Specification Signal Name	Output Signal Name	Signal Selection Register Settings	On-Chip Peripheral Module Settings
P1	7	SSLD1_OE	SSLD1		RSPID.SPCR.MSTR = 1, SPMS=0, RSPID.SPSR.MODF=0, RSPID.SPDCR.SLSEL0 = 0, RSPID.SPPCR.SPOM = 0*1, or RSPID.SPCR.MSTR = 1, SPMS = 0, RSPID.SPSR.MODF = 0, RSPID.SPDCR.SLSEL0 = 0, RSPID.SPPCR.SPOM = 1, RSPID.SSLO1 = 0*2
	6	SCK3_OE	SCK3		SCI3.SMR.C/A = 0, SCI3.SCR.CKE[1:0] = 01, or SCI3.SMR.C/A = 1, SCI3.SCR.CKE1 = 0
	4	TxD3_OE	TxD3		SCI3.SCR.TE = 1
	3	SSLD0_OE	SSLD0		RSPID.SPCR.MSTR = 1, MODFEN = 0, SPMS = 0, RSPID.SPSR.MODF = 0, RSPID.SPPCR.SPOM = 0*1, or RSPID.SPCR.MSTR = 1, MODFEN = 0, SPMS = 0, RSPID.SPSR.MODF = 0, RSPID.SPPCR.SPOM = 1, RSPID.SSLO0 = 0*2
	2	RSPCKD_OE	RSPCKD		RSPID.SPCR.MSTR = 1, RSPID.SPSR.MODF = 0, RSPID.SPPCR.SPOM = 0*1, or RSPID.SPCR.MSTR = 1, RSPID.SPSR.MODF = 0, RSPID.SPPCR.SPOM = 1, RSPID.RSPCKO = 0*2
	1	MISOD_OE	MISOD		RSPID.SPCR.SPE = 1, MSTR = 0, SPMS = 0, RSPID.SPPCR.SPOM = 0*1*3, or RSPID.SPCR.SPE = 1, MSTR = 0, SPMS = 0, RSPID.SPPCR.SPOM = 1, RSPID.MISOO = 0*2*3, or RSPID.SPCR.SPE = 1, MSTR = 0, SPMS = 1, RSPID.SPPCR.SPOM = 0*1*4, or RSPID.SPCR.SPE = 1, MSTR = 0, SPMS = 1, RSPID.SPPCR.SPOM = 1, RSPID.MISOO = 0*2*4
	0	MOSID_OE	MOSID		RSPID.SPCR.MSTR = 1, RSPID.SPSR.MODF = 0, RSPID.SPPCR.SPOM = 0*1, or RSPID.SPCR.MSTR = 1, RSPID.SPSR.MODF = 0, RSPID.SPPCR.SPOM = 1, RSPID.MOSIO = 0*2
P3	7	TIOCB2_OE	TIOCB2		TPU0.TIOR_2.IOB3 = 0, IOB[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 011, TPU0.TCR.CCLR[1:0] = 00/01/11, or TPU0.TIOR_2.IOB3 = 0, IOB[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 000/100/101/110/111
		PO15_OE	PO15		PPG.NDERH.NDER15 = 1
	6	TIOCA2_OE	TIOCA2		TPU0.TIOR_2.IOA3 = 0, IOA[1:0] = 01/10/11, TPU0.TCR.CCLR[1:0] = 00/10/11, or TPU0.TIOR_2.IOA3 = 0, IOA[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 000/010/100/101/110/111
		PO14_OE	PO14		PPG.NDERH.NDER14 = 1

Port		Output Specification Signal Name	Output Signal Name	Signal Selection Register Settings	On-Chip Peripheral Module Settings
P3	5	TIOCB1_OE	TIOCB1		TPU0.TIOR_1.IOB3 = 0, IOB[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 011, TPU0.TCR.CCLR[1:0] = 00/01/11, or TPU0.TIOR_1.IOB3 = 0, IOB[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 000/100/101/110/111
		PO13_OE	PO13		PPG.NDERH.NDER13 = 1
	4	TIOCA1_OE	TIOCA1		TPU0.TIOR_1.IOA3 = 0, IOA[1:0] = 01/10/11, TPU0.TCR.CCLR[1:0] = 00/10/11, or TPU0.TIOR_1.IOA3 = 0, IOA[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 000/010/100/101/110/111
		PO12_OE	PO12		PPG.NDERH.NDER12 = 1
	3	TIOCD0_OE	TIOCD0		TPU0.TIORL_0.IOD3 = 0, IOD[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 011, BFB = 0, TPU0.TCR.CCLR2 = 0, or TPU0.TIORL_0.IOD3 = 0, IOD[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 011, BFB = 0, TPU0.TCR.CCLR[1:0] = 00/01/11, or TPU0.TIORL_0.IOD3 = 0, IOD[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 000, BFB = 0
		PO11_OE	PO11		PPG.NDERH.NDER11 = 1
	2	TIOCC0_OE	TIOCC0		TPU0.TIORL_0.IOC3 = 0, IOC[1:0] = 01/10/11, TPU0.TMDR.BFA = 0, TPU0.TCR.CCLR2 = 0, or TPU0.TIORL_0.IOC3 = 0, IOC[1:0] = 01/10/11, TPU0.TMDR.BFA = 0, TPU0.TCR.CCLR[1:0] = 00/10/11, or TPU0.TIORL_0.IOC3 = 0, IOC[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 000/010, BFA = 0
		PO10_OE	PO10		PPG.NDERH.NDER10 = 1
	1	TIOCB0_OE	TIOCB0		TPU0.TIORH_0.IOB3 = 0, IOB[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 011, TPU0.TCR.CCLR2 = 1, or TPU0.TIORH_0.IOB3 = 0, IOB[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 011, TPU0.TCR.CCLR[1:0] = 00/01/11, or TPU0.TIORH_0.IOB3 = 0, IOB[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 000
		PO9_OE	PO9		PPG.NDERH.NDER9 = 1
	0	TIOCA0_OE	TIOCA0		TPU0.TIORH_0.IOA3 = 0, IOA[1:0] = 01/10/11, TPU0.TCR.CCLR2 = 1, or TPU0.TIORH_0.IOA3 = 0, IOA[1:0] = 01/10/11, TPU0.TCR.CCLR[1:0] = 00/10/11, or TPU0.TIORH_0.IOA3 = 0, IOA[1:0] = 01/10/11, TPU0.TMDR.MD[2:0] = 000/010
		PO8_OE	PO8		PPG.NDERH.NDER8 = 1
P6	5	CTx0-B_OE	CTx0	PFCR5.RCANMD = 0, RCAN0S = 1	RCANGL.RCANMON0.RCANE = 1

Port		Output Specification Signal Name	Output Signal Name	Signal Selection Register Settings	On-Chip Peripheral Module Settings
P6	5	(CTx0 or CTx1)-B_OE	(CTx0 or CTx1)	PFCR5.RCANMD = 1, RCAN0S = 1	RCANGL.RCANMON0.RCANE = 1, RCANMON1.RCANE = 1*5
	4	CTx0-A_OE	CTx0	PFCR5.RCANMD = 0, RCAN0S = 0	RCANGL.RCANMON0.RCANE = 1
		(CTx0 or CTx1)-A_OE	(CTx0 or CTx1)	PFCR5.RCANMD = 1, RCAN0S = 0	RCANGL.RCANMON0.RCANE = 1, RCANMON1.RCANE = 1*5
	2	CTx1_OE	CTx1	PFCR5.RCANMD = 0, RCAN1S = 0	RCANGL.RCANMON1.RCANE = 1
		SCK4_OE	SCK4		SCI4.SMR.C/A = 0, SCI4.SCR.CKE[1:0] = 01, or SCI4.SMR.C/A = 1, SCI4.SCR.CKE1 = 0
	0	TxD4_OE	TxD4		SCI4.SCR.TE = 1
PA	7	SSLC3_OE	SSLC3		RSPIC.SPCR.MSTR = 1, SPMS = 0, RSPIC.SPSR.MODF = 0, RSPIC.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPIC.SPPCR.SPOM = 0*1, or RSPIC.SPCR.MSTR = 1, SPMS = 0, RSPIC.SPSR.MODF = 0, RSPIC.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPIC.SPPCR.SPOM = 1, RSPIC.SSLO3 = 0*2
		Bϕ_OE	Bϕ		SYSC.SCKCR0.PSTOP1 = 0, PORT.PADDR.PA7DDR = 1
	6	SSLC2_OE	SSLC2		RSPIC.SPCR.MSTR = 1, SPMS = 0, RSPIC.SPSR.MODF = 0, RSPIC.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPIC.SPPCR.SPOM = 0*1, or RSPIC.SPCR.MSTR = 1, SPMS = 0, RSPIC.SPSR.MODF = 0, RSPIC.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPIC.SPPCR.SPOM = 1, RSPIC.SSLO2 = 0*2
	5	SSLC1_OE	SSLC1		RSPIC.SPCR.MSTR = 1, SPMS = 0, RSPIC.SPSR.MODF = 0, RSPIC.SPDCR.SLSEL0 = 0, RSPIC.SPPCR.SPOM = 0*1, or RSPIC.SPCR.MSTR = 1, SPMS = 0, RSPIC.SPSR.MODF = 0, RSPIC.SPDCR.SLSEL0 = 0, RSPIC.SPPCR.SPOM = 1, RSPIC.SSLO1 = 0*2
	4	SSLC0_OE	SSLC0		RSPIC.SPCR.MSTR = 1, MODFEN = 0, SPMS = 0, RSPIC.SPSR.MODF = 0, RSPIC.SPPCR.SPOM = 0*1, or RSPIC.SPCR.MSTR = 1, MODFEN = 0, SPMS = 0, RSPIC.SPSR.MODF = 0, RSPIC.SPPCR.SPOM = 1, RSPIC.SSLO0 = 0*2
	3	MOSIC_OE	MOSIC		RSPIC.SPCR.MSTR = 1, RSPIC.SPSR.MODF = 0, RSPIC.SPPCR.SPOM = 0*1, or RSPIC.SPCR.MSTR = 1, RSPIC.SPSR.MODF = 0, RSPIC.SPPCR.SPOM = 1, RSPIC.MOSIO = 0*2

Port		Output Specification Signal Name	Output Signal Name	Signal Selection Register Settings	On-Chip Peripheral Module Settings
PA	2	MISOC_OE	MISOC		RSPIC.SPCR.SPE = 1, MSTR = 0, SPMS = 0, RSPIC.SPPCR.SPOM = 0*1*3, or RSPIC.SPCR.SPE = 1, MSTR = 0, SPMS = 0, RSPIC.SPPCR.SPOM = 1, RSPIC.MISOO = 0*2*3, or RSPIC.SPCR.SPE = 1, MSTR = 0, SPMS = 1, RSPIC.SPPCR.SPOM = 0*1*4, or RSPIC.SPCR.SPE = 1, MSTR = 0, SPMS = 1, RSPIC.SPPCR.SPOM = 1, RSPIC.MISOO = 0*2*4
	1	RSPCKC_OE	RSPCKC		RSPIC.SPCR.MSTR = 1, RSPIC.SPSR.MODF = 0, RSPIC.SPPCR.SPOM = 0*1, or RSPIC.SPCR.MSTR = 1, RSPIC.SPSR.MODF = 0, RSPIC.SPPCR.SPOM = 1, RSPIC.RSPCKO = 0*2
PD	7	SSLB0_OE	SSLB0		RSPIB.SPCR.MSTR = 1, MODFEN = 0, SPMS = 0, RSPIB.SPSR.MODF = 0, RSPIB.SPPCR.SPOM = 0*1,or RSPIB.SPCR.MSTR = 1, MODFEN = 0, SPMS = 0, RSPIB.SPSR.MODF = 0, RSPIB.SPPCR.SPOM = 1, RSPIB.SSLO0 = 0*2
	6	RSPCKB_OE	RSPCKB		RSPIB.SPCR.MSTR = 1, RSPIB.SPSR.MODF = 0, RSPIB.SPPCR.SPOM = 0*1, or RSPIB.SPCR.MSTR = 1, RSPIB.SPSR.MODF = 0, RSPIB.SPPCR.SPOM = 1, RSPIB.RSPCKO = 0*2
	5	MISOB_OE	MISOB		RSPIB.SPCR.SPE = 1, MSTR = 0, SPMS = 0, RSPIB.SPPCR.SPOM = 0*1*3, or RSPIB.SPCR.SPE = 1, MSTR = 0, SPMS = 0, RSPIB.SPPCR.SPOM = 1, RSPIB.MISOO = 0*2*3, or RSPIB.SPCR.SPE = 1, MSTR = 0, SPMS = 1, RSPIB.SPPCR.SPOM = 0*1*4, or RSPIB.SPCR.SPE = 1, MSTR = 0, SPMS = 1, RSPIB.SPPCR.SPOM = 1, RSPIB.MISOO = 0*2*4
	4	MOSIB_OE	MOSIB		RSPIB.SPCR.MSTR = 1, RSPIB.SPSR.MODF = 0, RSPIB.SPPCR.SPOM = 0*1, or RSPIB.SPCR.MSTR = 1, RSPIB.SPSR.MODF = 0, RSPIB.SPPCR.SPOM = 1, RSPIB.MOSIO = 0*2
	3	SSLA0_OE	SSLA0	PFCR8.RSPISA = 0	RSPIA.SPCR.MSTR = 1, MODFEN = 0, SPMS = 0, RSPIA.SPSR.MODF = 0, RSPIA.SPPCR.SPOM = 0*1, or RSPIA.SPCR.MSTR = 1, MODFEN = 0, SPMS = 0, RSPIA.SPSR.MODF = 0, RSPIA.SPPCR.SPOM = 1, RSPIA.SSLO0 = 0*2
	2	RSPCKA_OE	RSPCKA	PFCR8.RSPISA = 0	RSPIA.SPCR.MSTR = 1, RSPIA.SPSR.MODF = 0, RSPIA.SPPCR.SPOM = 0*1, or RSPIA.SPCR.MSTR = 1, RSPIA.SPSR.MODF = 0, RSPIA.SPPCR.SPOM = 1, RSPIA.RSPCKO = 0*2

Port		Output Specification Signal Name	Output Signal Name	Signal Selection Register Settings	On-Chip Peripheral Module Settings
PD	1	MISOA_OE	MISOA	PFCR8.RSPISA = 0	RSPIA.SPCR.SPE = 1, MSTR = 0, SPMS = 0, RSPIA.SPPCR.SPOM = 0*1*3, or RSPIA.SPCR.SPE = 1, MSTR = 0, SPMS = 0, RSPIA.SPPCR.SPOM = 1, RSPIA.MISOO = 0*2*3, or RSPIA.SPCR.SPE = 1, MSTR = 0, SPMS = 1, RSPIA.SPPCR.SPOM = 0*1*4, or RSPIA.SPCR.SPE = 1, MSTR = 0, SPMS = 1, RSPIA.SPPCR.SPOM = 1, RSPIA.MISOO = 0*2*4
	0	MOSIA_OE	MOSIA	PFCR8.RSPISA = 0	RSPIA.SPCR.MSTR=1, RSPIA.SPSR.MODF = 0, RSPIA.SPPCR.SPOM = 0*1, or RSPIA.SPCR.MSTR = 1, RSPIA.SPSR.MODF = 0, RSPIA.SPPCR.SPOM = 1, RSPIA.MOSIO = 0*2
PH	7	SSLD3_OE	SSLD3		RSPID.SPCR.MSTR = 1, SPMS = 0, RSPID.SPSR.MODF = 0, RSPID.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPID.SPPCR.SPOM = 0*1, or RSPID.SPCR.MSTR = 1, SPMS = 0, RSPID.SPSR.MODF = 0, RSPID.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPID.SPPCR.SPOM = 1, RSPID.SSLO3 = 0*2
	6	SSLD2_OE	SSLD2		RSPID.SPCR.MSTR = 1, SPMS = 0, RSPID.SPSR.MODF = 0, RSPID.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPID.SPPCR.SPOM = 0*1, or RSPID.SPCR.MSTR = 1, SPMS = 0, RSPID.SPSR.MODF = 0, RSPID.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPID.SPPCR.SPOM = 1, RSPID.SSLO2 = 0*2
	5	SSLB3_OE	SSLB3		RSPIB.SPCR.MSTR = 1, SPMS = 0, RSPIB.SPSR.MODF = 0, RSPIB.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPIB.SPPCR.SPOM = 0*1, or RSPIB.SPCR.MSTR = 1, SPMS = 0, RSPIB.SPSR.MODF = 0, RSPIB.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPIB.SPPCR.SPOM = 1, RSPIB.SSLO3 = 0*2
	4	SSLB2_OE	SSLB2		RSPIB.SPCR.MSTR = 1, SPMS = 0, RSPIB.SPSR.MODF = 0, RSPIB.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPIB.SPPCR.SPOM = 0*1, or RSPIB.SPCR.MSTR = 1, SPMS = 0, RSPIB.SPSR.MODF = 0, RSPIB.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPIB.SPPCR.SPOM = 1, RSPIB.SSLO2 = 0*2
	3	SSLB1_OE	SSLB1		RSPIB.SPCR.MSTR = 1, SPMS = 0, RSPIB.SPSR.MODF = 0, RSPIB.SPDCR.SLSEL0 = 0, RSPIB.SPPCR.SPOM = 0*1, or RSPIB.SPCR.MSTR = 1, SPMS = 0, RSPIB.SPSR.MODF = 0, RSPIB.SPDCR.SLSEL0 = 0, RSPIB.SPPCR.SPOM = 1, RSPIB.SSLO1 = 0*2

Port		Output Specification Signal Name	Output Signal Name	Signal Selection Register Settings	On-Chip Peripheral Module Settings
PH	2	SSLA3-A_OE	SSLA3	PFCR8.RSPISA = 0	RSPIA.SPCR.MSTR = 1, SPMS = 0, RSPIA.SPSR.MODF = 0, RSPIA.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPIA.SPPCR.SPOM = 0*1, or RSPIA.SPCR.MSTR = 1, SPMS = 0, RSPIA.SPSR.MODF = 0, RSPIA.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPIA.SPPCR.SPOM = 1, RSPIA.SSLO3 = 0*2
	1	SSLA2-A_OE	SSLA2	PFCR8.RSPISA = 0	RSPIA.SPCR.MSTR = 1, SPMS = 0, RSPIA.SPSR.MODF = 0, RSPIA.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPIA.SPPCR.SPOM = 0*1, or RSPIA.SPCR.MSTR = 1, SPMS = 0, RSPIA.SPSR.MODF = 0, RSPIA.SPDCR.SLSEL0 = 0, SLSEL1 = 0, RSPIA.SPPCR.SPOM = 1, RSPIA.SSLO2 = 0*2
	0	SSLA1-A_OE	SSLA1	PFCR8.RSPISA = 0	RSPIA.SPCR.MSTR = 1, SPMS = 0, RSPIA.SPSR.MODF = 0, RSPIA.SPDCR.SLSEL0 = 0, RSPIA.SPPCR.SPOM = 0*1, or RSPIA.SPCR.MSTR = 1, SPMS = 0, RSPIA.SPSR.MODF = 0, RSPIA.SPDCR.SLSEL0 = 0, RSPIA.SPPCR.SPOM = 1, RSPIA.SSLO1 = 0*2
PJ	7	TIOCB8_OE	TIOCB8		TPU1.TIOR_2.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 011, TPU1.TCR.CCLR[1:0] = 00/01/11, or TPU1.TIOR_2.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000/100/101/110/111
	6	TIOCA8_OE	TIOCA8		TPU1.TIOR_2.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TCR.CCLR[1:0] = 00/10/11, or TPU1.TIOR_2.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000/010/100/101/110/111
	5	TIOCB7_OE	TIOCB7		TPU1.TIOR_1.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 011, TPU1.TCR.CCLR[1:0] = 00/01/11, or TPU1.TIOR_1.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000/100/101/110/111
	4	TIOCA7_OE	TIOCA7		TPU1.TIOR_1.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TCR.CCLR[1:0] = 00/10/11, or TPU1.TIOR_1.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000/010/100/101/110/111
	3	TIOCD6_OE	TIOCD6		TPU1.TIORL_0.IOD3 = 0, IOD[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 011, BFB = 0, TPU1.TCR.CCLR2 = 0, or TPU1.TIORL_0.IOD3 = 0, IOD[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 011, BFB = 0, TPU1.TCR.CCLR[1:0] = 00/01/11, or TPU1.TIORL_0.IOD3 = 0, IOD[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000, BFB = 0

Port		Output Specification Signal Name	Output Signal Name	Signal Selection Register Settings	On-Chip Peripheral Module Settings
PJ	2	TIOCC6_OE	TIOCC6		TPU1.TIORL_0.IOC3 = 0, IOC[1:0] = 01/10/11, TPU1.TMDR.BFA = 0, TPU1.TCR.CCLR2 = 0, or TPU1.TIORL_0.IOC3 = 0, IOC[1:0] = 01/10/11, TPU1.TMDR.BFA = 0, TPU1.TCR.CCLR[1:0] = 00/10/11, or TPU1.TIORL_0.IOC3 = 0, IOC[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000/010, BFA = 0
	1	TIOCB6_OE	TIOCB6		TPU1.TIORH_0.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 011, TPU1.TCR.CCLR2 = 1, or TPU1.TIORH_0.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 011, TPU1.TCR.CCLR[1:0] = 00/01/11, or TPU1.TIORH_0.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000
	0	TIOCA6_OE	TIOCA6		TPU1.TIORH_0.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TCR.CCLR2 = 1, or TPU1.TIORH_0.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TCR.CCLR[1:0] = 00/10/11, or TPU1.TIORH_0.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000/010
PK	7	TIOCB11_OE	TIOCB11		TPU1.TIOR_5.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 011, TPU1.TCR.CCLR[1:0] = 00/01/11, or TPU1.TIOR_5.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000/100/101/110/111
	6	TIOCA11_OE	TIOCA11		TPU1.TIOR_5.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TCR.CCLR[1:0] = 00/10/11, or TPU1.TIOR_5.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000/010/100/101/110/111
	5	TIOCB10_OE	TIOCB10		TPU1.TIOR_4.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 011, TPU1.TCR.CCLR[1:0] = 00/01/11, or TPU1.TIOR_4.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000/100/101/110/111
	4	TIOCA10_OE	TIOCA10		TPU1.TIOR_4.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TCR.CCLR[1:0] = 00/10/11, or TPU1.TIOR_4.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000/010/100/101/110/111
	3	TIOCD9_OE	TIOCD9		TPU1.TIORL_3.IOD3 = 0, IOD[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 011, BFB = 0, TPU1.TCR.CCLR2 = 0, or TPU1.TIORL_3.IOD3 = 0, IOD[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 011, BFB = 0, TPU1.TCR.CCLR[1:0] = 00/01/11, or TPU1.TIORL_3.IOD3 = 0, IOD[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000, BFB = 0

Port		Output Specification Signal Name	Output Signal Name	Signal Selection Register Settings	On-Chip Peripheral Module Settings
PK	2	TIOCC9_OE	TIOCC9		TPU1.TIORL_3.IOC3 = 0, IOC[1:0] = 01/10/11, TPU1.TMDR.BFA = 0, TPU1.TCR.CCLR2 = 0, or TPU1.TIORL_3.IOC3 = 0, IOC[1:0] = 01/10/11, TPU1.TMDR.BFA = 0, TPU1.TCR.CCLR[1:0] = 00/10/11, or TPU1.TIORL_3.IOC3 = 0, IOC[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000/010, BFA = 0
	1	TIOCB9_OE	TIOCB9		TPU1.TIORH_3.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 011, TPU1.TCR.CCLR2 = 1, or TPU1.TIORH_3.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 011, TPU1.TCR.CCLR[1:0] = 00/01/11, or TPU1.TIORH_3.IOB3 = 0, IOB[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000
	0	TIOCA9_OE	TIOCA9		TPU1.TIORH_3.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TCR.CCLR2 = 1, or TPU1.TIORH_3.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TCR.CCLR[1:0] = 00/10/11, or TPU1.TIORH_3.IOA3 = 0, IOA[1:0] = 01/10/11, TPU1.TMDR.MD[2:0] = 000/010

Notes: 1. SPOM = 0 → CMOS output
2. SPOM = 1 → Open-drain output
3. This setting is made at SPI operation (four-wire method) when the polarity of the SSL signals set by SSLiP (RSPI.SSLP) is active level.
4. This setting is made at clock synchronous operation (three-wire method).
5. Be sure to use this setting in 1 channel × 64 mailboxes mode.

9.3 Port Function Controller

The port function controller controls the I/O ports.

The port function controller incorporates the following registers.

- Port function control register 5 (PFCR5)
- Port function control register 6 (PFCR6)
- Port function control register 8 (PFCR8)
- Port function control register 9 (PFCR9)
- Port function control register A (PFCRA)
- Port function control register B (PFCRB)
- Port function control register C (PFCRC)
- Port function control register D (PFCRD)

9.3.1 Port Function Control Register 5 (PFCR5)

PFCR5 selects the I/O pins for the RCAN.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	RCANMD	—	—	RCANS1	RCANS0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 5	—	All 0	R/W	Reserved The initial value should not be changed.
4	RCANMD	0	R/W	RCAN Mode Select Selects the RCAN mode. 0: 2 channels × 32 mailboxes mode CTx_0 is output from P64 or P65, and CTx_1 is output from P62. CRx_0 is input from P65 or P64, and CRx_1 is input from P66. 1: 1 channel × 64 mailboxes mode CTx_0 and CTx_1 are output from P64 or P65. CRx_0 and CRx_1 are input from P65 or P64.
3 to 2	—	All 0	R/W	Reserved The initial value should not be changed.
1	RCANS1	0	R/W	RCAN_1 Control Pin Select (Valid only when RCANMD = 0) Selects the I/O port to control RCAN_1. 0: Specifies P62 as CTx_1 1: Prohibits use of P62 as CTx_1
0	RCANS0	0	R/W	RCAN_0 Control Pin Select Selects the I/O port to control RCAN_0. 0: Specifies P64 as CTx_0, and P65 as CRx_0 (when RCANMD = 0) Specifies P64 as CTx_0 or CTx_1, and P65 as CRx_0 or CRx_1 (when RCANMD = 1) 1: Specifies P65 as CTx_0, and P64 as CRx_0 (when RCANMD = 0) Specifies P65 as CTx_0 or CTx_1, and P64 as CRx_0 or CRx_1 (when RCANMD = 1)

9.3.2 Port Function Control Register 6 (PFCR6)

PFCR6 selects the external input pin for the TPU clock input pin.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	TCLKS	—	—	—
Initial Value:	0	1	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	—	0	R/W	Reserved The initial value should not be changed.
6	—	1	R/W	Reserved The initial value should not be changed.
5	—	0	R/W	Reserved The initial value should not be changed.
4	—	0	R	Reserved This bit is always read as 0 and cannot be modified. The initial value should not be changed.
3	TCLKS	0	R/W	TPU (Unit 0) External Clock Input Pin Select Selects the input pin for TPU external clock. 0: Specifies P32, P33, P35, and P37 as external clock input pins 1: Specifies P14 to P17 as external clock input pins
2 to 0	—	All 0	R/W	Reserved The initial value should not be changed.

9.3.3 Port Function Control Register 8 (PFCR8)

PFCR8 selects the SCI channel that is linked with the HWLIN and input/output pins of RSPIA.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	HLIS2	HLIS1	HLIS0	RSPISA	—	—
Initial Value:	0	0	1	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 6	—	All 0	R/W	Reserved The initial value should not be changed.
5 to 3	HLIS2 HLIS1 HLIS0	1 0 0	R/W R/W R/W	HWLIN Control Select These bits select the SCI channel that is linked with HWLIN. 000: Setting prohibited 001: Setting prohibited 010: Setting prohibited 011: SCI channel 3 is linked with the HWLIN 100: SCI channel 4 is linked with the HWLIN Other than above: Setting prohibited
2	RSPISA	0	R/W	RSPI_A Control Pin Select Selects the I/O port to control RSPI_A. 0: Specifies pins PD0 to PD3 and PH0 to PH2 as RSPI_A control pins 1: Prohibits the use of PD0 to PD3 and PH0 to PH2 as RSPI_A control pins
1 to 0	—	All 0	R/W	Reserved The initial value should not be changed.

9.3.4 Port Function Control Register 9 (PFCR9)

PFCR9 selects the multiple functions for the TPU (unit 0) I/O pins.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	TPUMS2	TPUMS1	TPUMS0A	TPUMS0B
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 4	—	All 0	R/W	Reserved The initial value should not be changed.
3	TPUMS2	0	R/W	TPU I/O Pin Multiplex Function Select Selects TIOCA2 function 0: Specifies P36 as output compare output and input capture 1: Specifies P37 as input capture input and P36 as output compare
2	TPUMS1	0	R/W	TPU I/O Pin Multiplex Function Select Selects TIOCA1 function 0: Specifies P34 as output compare output and input capture 1: Specifies P35 as input capture input and P34 as output compare
1	TPUMS0A	0	R/W	TPU I/O Pin Multiplex Function Select Selects TIOCA0 function 0: Specifies P30 as output compare output and input capture 1: Specifies P31 as input capture input and P30 as output compare

Bit	Bit Name	Initial Value	R/W	Description
0	TPUMS0B	0	R/W	TPU I/O Pin Multiplex Function Select Selects TIOCC0 function 0: Specifies P32 as output compare output and input capture 1: Specifies P33 as input capture input and P32 as output compare

9.3.5 Port Function Control Register A (PFCRA)

PFCRA selects the multiple functions for the TPU (unit 1) I/O pins.

Bit	7	6	5	4	3	2	1	0
Bit Name	TPUMS11	TPUMS10	TPUMS9A	TPUMS9B	TPUMS8	TPUMS7	TPUMS6A	TPUMS6B
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	TPUMS11	0	R/W	TPU I/O Pin Multiplex Function Select Selects TIOCA11 function 0: Specifies PK6 as output compare output and input capture 1: Specifies PK7 as input capture input and PK6 as output compare
6	TPUMS10	0	R/W	TPU I/O Pin Multiplex Function Select Selects TIOCA10 function 0: Specifies PK4 as output compare output and input capture 1: Specifies PK5 as input capture input and PK4 as output compare
5	TPUMS9A	0	R/W	TPU I/O Pin Multiplex Function Select Selects TIOCA9 function 0: Specifies PK0 as output compare output and input capture 1: Specifies PK1 as input capture input and PK0 as output compare

Bit	Bit Name	Initial Value	R/W	Description
4	TPUMS9B	0	R/W	TPU I/O Pin Multiplex Function Select Selects TIOCC9 function 0: Specifies PK2 as output compare output and input capture 1: Specifies PK3 as input capture input and PK2 as output compare
3	TPUMS8	0	R/W	TPU I/O Pin Multiplex Function Select Selects TIOCA8 function 0: Specifies PJ6 as output compare output and input capture 1: Specifies PJ7 as input capture input and PJ6 as output compare
2	TPUMS7	0	R/W	TPU I/O Pin Multiplex Function Select Selects TIOCA7 function 0: Specifies PJ4 as output compare output and input capture 1: Specifies PJ5 as input capture input and PJ4 as output compare
1	TPUMS6A	0	R/W	TPU I/O Pin Multiplex Function Select Selects TIOCA6 function 0: Specifies PJ0 as output compare output and input capture 1: Specifies PJ1 as input capture input and PJ0 as output compare
0	TPUMS6B	0	R/W	TPU I/O Pin Multiplex Function Select Selects TIOCC6 function 0: Specifies PJ2 as output compare output and input capture 1: Specifies PJ3 as input capture input and PJ2 as output compare

9.3.6 Port Function Control Register B (PFCRB)

PFCRB selects the input pins for $\overline{\text{IRQ14}}$ to $\overline{\text{IRQ8}}$.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	ITS14	ITS13	ITS12	ITS11	ITS10	ITS9	ITS8
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	—	0	R/W	Reserved This bit is always read as 0. The write value should always be 0.
6	ITS14	0	R/W	$\overline{\text{IRQ14}}$ Pin Select Selects an input pin for $\overline{\text{IRQ14}}$. 0: Pin P66 must not be used as $\overline{\text{IRQ14}}$ input 1: Pin P66 is used as $\overline{\text{IRQ14}}$ input
5	ITS13	0	R/W	$\overline{\text{IRQ13}}$ Pin Select Selects an input pin for $\overline{\text{IRQ13}}$. 0: Pin P65 must not be used as $\overline{\text{IRQ13}}$ input 1: Pin P65 is used as $\overline{\text{IRQ13}}$ input
4	ITS12	0	R/W	$\overline{\text{IRQ12}}$ Pin Select Selects an input pin for $\overline{\text{IRQ12}}$. 0: Pin P64 must not be used as $\overline{\text{IRQ12}}$ input 1: Pin P64 is used as $\overline{\text{IRQ12}}$ input
3	ITS11	0	R/W	$\overline{\text{IRQ11}}$ Pin Select Selects an input pin for $\overline{\text{IRQ11}}$. 0: Pin P63 must not be used as $\overline{\text{IRQ11}}$ input 1: Pin P63 is used as $\overline{\text{IRQ11}}$ input

Bit	Bit Name	Initial Value	R/W	Description
2	ITS10	0	R/W	$\overline{\text{IRQ10}}$ Pin Select Selects an input pin for $\overline{\text{IRQ10}}$. 0: Pin P62 must not be used as $\overline{\text{IRQ10}}$ input 1: Pin P62 is used as $\overline{\text{IRQ10}}$ input
1	ITS9	0	R/W	$\overline{\text{IRQ9}}$ Pin Select Selects an input pin for $\overline{\text{IRQ9}}$. 0: Pin P61 must not be used as $\overline{\text{IRQ9}}$ input 1: Pin P61 is used as $\overline{\text{IRQ9}}$ input
0	ITS8	0	R/W	$\overline{\text{IRQ8}}$ Pin Select Selects an input pin for $\overline{\text{IRQ8}}$. 0: Pin P60 must not be used as $\overline{\text{IRQ8}}$ input 1: Pin P60 is used as $\overline{\text{IRQ8}}$ input

9.3.7 Port Function Control Register C (PFCRC)

PFCRC selects the input pins for $\overline{\text{IRQ7}}$ to $\overline{\text{IRQ0}}$.

Bit	7	6	5	4	3	2	1	0
Bit Name	ITS7	ITS6	ITS5	ITS4	ITS3	ITS2	ITS1	ITS0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	ITS7	0	R/W	$\overline{\text{IRQ7}}$ Pin Select Selects an input pin for $\overline{\text{IRQ7}}$. 0: Pin P17 is used as $\overline{\text{IRQ7}}$-A input 1: Pin P57 is used as $\overline{\text{IRQ7}}$-B input
6	ITS6	0	R/W	$\overline{\text{IRQ6}}$ Pin Select Selects an input pin for $\overline{\text{IRQ6}}$. 0: Pin P16 is used as $\overline{\text{IRQ6}}$-A input 1: Pin P56 is used as $\overline{\text{IRQ6}}$-B input

Bit	Bit Name	Initial Value	R/W	Description
5	ITS5	0	R/W	$\overline{\text{IRQ5}}$ Pin Select Selects an input pin for $\overline{\text{IRQ5}}$. 0: Pin P15 is used as $\overline{\text{IRQ5}}$-A input 1: Pin P55 is used as $\overline{\text{IRQ5}}$-B input
4	ITS4	0	R/W	$\overline{\text{IRQ4}}$ Pin Select Selects an input pin for $\overline{\text{IRQ4}}$. 0: Pin P14 is used as $\overline{\text{IRQ4}}$-A input 1: Pin P54 is used as $\overline{\text{IRQ4}}$-B input
3	ITS3	0	R/W	$\overline{\text{IRQ3}}$ Pin Select Selects an input pin for $\overline{\text{IRQ3}}$. 0: Pin P13 is used as $\overline{\text{IRQ3}}$-A input 1: Pin P53 is used as $\overline{\text{IRQ3}}$-B input
2	ITS2	0	R/W	$\overline{\text{IRQ2}}$ Pin Select Selects an input pin for $\overline{\text{IRQ2}}$. 0: Pin P12 is used as $\overline{\text{IRQ2}}$-A input 1: Pin P52 is used as $\overline{\text{IRQ2}}$-B input
1	ITS1	0	R/W	$\overline{\text{IRQ1}}$ Pin Select Selects an input pin for $\overline{\text{IRQ1}}$. 0: Pin P11 is used as $\overline{\text{IRQ1}}$-A input 1: Pin P51 is used as $\overline{\text{IRQ1}}$-B input
0	ITS0	0	R/W	$\overline{\text{IRQ0}}$ Pin Select Selects an input pin for $\overline{\text{IRQ0}}$. 0: Pin P10 is used as $\overline{\text{IRQ0}}$-A input 1: Pin P50 is used as $\overline{\text{IRQ0}}$-B input

9.3.8 Port Function Control Register D (PFCRD)

PFCRD sets the driving abilities of all ports.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	DRVDWNE
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 1	—	All 0	R	Reserved The initial value should not be changed.
0	DRVDWNE	0	R/W	Driving Ability Down Enable Enables or disables the driving ability to be decreased. 0: Driving ability set by DSR of each port 1: Driving abilities of all ports decreased

9.4 Usage Notes

9.4.1 Notes on Input Buffer Control Register (ICR) Setting

- When the ICR setting is changed, the LSI may malfunction due to an edge occurred internally according to the pin states. To change the ICR setting, fix the pin high or disable the input function corresponding to the pin by setting the on-chip peripheral module registers.
- If an input is enabled by setting ICR while multiple input functions are assigned to the pin, the pin state is reflected in all the inputs of individual modules. Care the settings of unused input function on each module side.
- When a pin is used as an output, data to be output from the pin will be latched as the pin state if the input function corresponding to the pin is enabled. To use the pin as an output, disable the input function for the pin by setting ICR.

9.4.2 Notes on Port Function Control Register (PFCR) Settings

- The PFC controls I/O ports. To specify the function of each pin, select the input/output destination before enabling the input/output function.
- When the input/output destination is changed by the corresponding selection bit, an edge may occur if the previous pin level differs from the pin level after the change. To change the pin direction correctly, follow the procedure shown below.
  1. Disable the input function corresponding to the pin by the on-chip module registers.
  2. Select the input function by setting PFCR.
  3. Enable the input function.
- If a pin function has both a selection bit that modifies the input/output destination and an enable bit that enables the pin function, first specify the input/output destination by the selection bit and then enable the pin function by the enable bit.
- When the driving ability is set to be decreased, the DRVDWNE in PFCRD is given priority over DSR and becomes as follows.

DRVDWNE	DSR	Driving Ability
1	—	Small
0	0	Large (initial value)
0	1	Small

Section 10 16-Bit Timer Pulse Unit (TPU)

This LSI has two on-chip 16-bit timer pulse units (TPU), unit 0 and unit 1, each comprises six channels. Therefore, this LSI includes twelve channels.

Functions of unit 0 and unit 1 are shown in table 10.1 and table 10.2 respectively. Block diagrams of unit 0 and unit 1 are shown in figure 10.1 and figure 10.2 respectively.

This section explains regarding unit 0. This explanation is common to unit 1.

10.1 Features

- Maximum 16 pulse inputs/outputs
- Selection of eight counter input clocks for each channel
- The following operations can be set for each channel:
  - Waveform output at compare match
  - Input capture function
  - Counter clear operation
  - Synchronous operations:
    - Multiple timer counters (TCNT) can be written to simultaneously
    - Simultaneous clearing by compare match and input capture possible
    - Simultaneous input/output for registers possible by counter synchronous operation
    - Maximum of 15-phase PWM output possible by combination with synchronous operation

Note: This LSI does not have the TIOCA3, TIOCB3, TIOCC3, TIOCD3, TIOCA4, TIOCB4, TIOCA5, and TIOCB5 input/output pins on channels 3, 4, and 5. Therefore, the input capture input, waveform output on compare match (0, 1, or toggle output), and PWM waveform output are not available.

- Buffer operation settable for channels 0 and 3
- Phase counting mode settable independently for each of channels 1, 2, 4, and 5
- Cascaded operation
- Fast access via internal 16-bit bus
- 26 interrupt sources
- Automatic transfer of register data
- Programmable pulse generator (PPG) output trigger can be generated (unit 0 only)
- Conversion start trigger for the A/D converter can be generated (unit 0 only)
- Module stop mode can be set

Table 10.1 TPU (Unit 0) Functions

Item	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
Count clock	Pφ/1	Pφ/1	Pφ/1	Pφ/1	Pφ/1	Pφ/1
	Pφ/4	Pφ/4	Pφ/4	Pφ/4	Pφ/4	Pφ/4
	Pφ/16	Pφ/16	Pφ/16	Pφ/16	Pφ/16	Pφ/16
	Pφ/64	Pφ/64	Pφ/64	Pφ/64	Pφ/64	Pφ/64
	TCLKA	Pφ/256	Pφ/1024	Pφ/256	Pφ/1024	Pφ/256
	TCLKB	TCLKA	TCLKA	Pφ/1024	TCLKA	TCLKA
	TCLKC	TCLKB	TCLKB	Pφ/4096	TCLKC	TCLKC
	TCLKD		TCLKC	TCLKA		TCLKD
General registers (TGR)	TGRA_0	TGRA_1	TGRA_2	TGRA_3	TGRA_4	TGRA_5
	TGRB_0	TGRB_1	TGRB_2	TGRB_3	TGRB_4	TGRB_5
General registers/ buffer registers	TGRC_0	—	—	TGRC_3	—	—
	TGRD_0			TGRD_3		
I/O pins	TIOCA0	TIOCA1	TIOCA2	TIOCA3*[1]	TIOCA4*[1]	TIOCA5*[1]
	TIOCB0	TIOCB1	TIOCB2	TIOCB3*[1]	TIOCB4*[1]	TIOCB5*[1]
	TIOCC0			TIOCC3*[1]		
	TIOCD0			TIOCD3*[1]		
Counter clear function	TGR compare match or input capture	TGR compare match or input capture	TGR compare match or input capture	TGR compare match or input capture*[2]	TGR compare match or input capture*[2]	TGR compare match or input capture*[2]

Item		Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
Compare match output	0 output	O	O	O	O*1	O*1	O*1
	1 output	O	O	O	O*1	O*1	O*1
	Toggle output	O	O	O	O*1	O*1	O*1
Input capture function		O	O	O	O*2	O*2	O*2
Synchronous operation		O	O	O	O	O	O
PWM mode		O	O	O	O*1	O*1	O*1
Phase counting mode		—	O	O	—	O	O
Buffer operation		O	—	—	O	—	—
DTC activation		TGR compare match or input capture	TGR compare match or input capture	TGR compare match or input capture	TGR compare match or input capture*2	TGR compare match or input capture*2	TGR compare match or input capture*2
DMAC activation		TGRA_0 compare match or input capture	TGRA_1 compare match or input capture	TGRA_2 compare match or input capture	TGRA_3 compare match or input capture*2	TGRA_4 compare match or input capture*2	TGRA_5 compare match or input capture*2
A/D converter trigger		TGRA_0 compare match or input capture	TGRA_1 compare match or input capture	TGRA_2 compare match or input capture	TGRA_3 compare match or input capture*2	TGRA_4 compare match or input capture*2	TGRA_5 compare match or input capture*2
PPG trigger		TGRA_0/ TGRB_0 compare match or input capture	TGRA_1/ TGRB_1 compare match or input capture	TGRA_2/ TGRB_2 compare match or input capture	TGRA_3/ TGRB_3 compare match or input capture	—	—

Item	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
Interrupt sources	5 sources	4 sources	4 sources	5 sources	4 sources	4 sources
	Compare match or input capture 0A	Compare match or input capture 1A	Compare match or input capture 2A	Compare match or input capture 3A	Compare match or input capture 4A	Compare match or input capture 5A
	Compare match or input capture 0B	Compare match or input capture 1B	Compare match or input capture 2B	Compare match or input capture 3B	Compare match or input capture 4B	Compare match or input capture 5B
	Compare match or input capture 0C			Compare match or input capture 3C		
	Compare match or input capture 0D			Compare match or input capture 3D		
	Overflow	Overflow	Overflow	Overflow	Overflow	Overflow
		Underflow	Underflow		Underflow	Underflow

[Legend] O: Possible
—: Not possible

Notes: 1. This LSI does not have the TIOCA3, TIOCB3, TIOCC3, TIOCD3, TIOCA4, TIOCB4, TIOCA5, and TIOCB5 input/output pins on channels 3, 4, and 5. Therefore, the input capture input, waveform output on compare match (0, 1, or toggle output), and PWM waveform output are not available.
2. In this LSI, the input capture function is not available on channels 3, 4, and 5.

Table 10.2 TPU (Unit 1) Functions

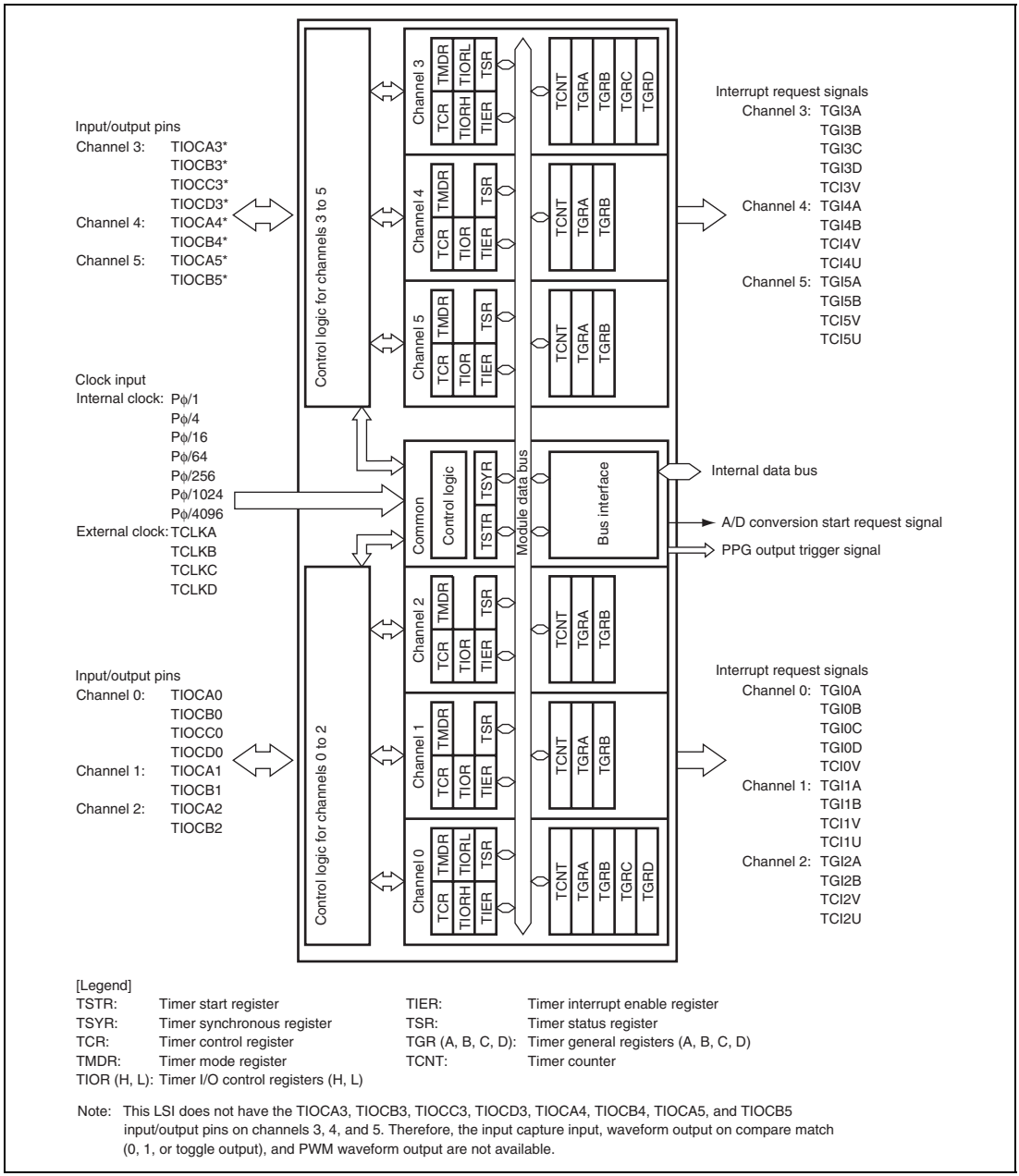
Item		Channel 6	Channel 7	Channel 8	Channel 9	Channel 10	Channel 11
Count clock		Pφ/1 Pφ/4 Pφ/16 Pφ/64 TCLKE TCLKF TCLKG TCLKH	Pφ/1 Pφ/4 Pφ/16 Pφ/64 Pφ/256 TCLKE TCLKF	Pφ/1 Pφ/4 Pφ/16 Pφ/64 Pφ/1024 TCLKE TCLKF TCLKG	Pφ/1 Pφ/4 Pφ/16 Pφ/64 Pφ/256 Pφ/1024 Pφ/4096 TCLKE	Pφ/1 Pφ/4 Pφ/16 Pφ/64 Pφ/1024 TCLKE TCLKG	Pφ/1 Pφ/4 Pφ/16 Pφ/64 Pφ/256 TCLKE TCLKG TCLKH
General registers (TGR)		TGRA_6 TGRB_6	TGRA_7 TGRB_7	TGRA_8 TGRB_8	TGRA_9 TGRB_9	TGRA_10 TGRB_10	TGRA_11 TGRB_11
General registers/ buffer registers		TGRC_6 TGRD_6	—	—	TGRC_9 TGRD_9	—	—
I/O pins		TIOCA6 TIOCB6 TIOCC6 TIOCD6	TIOCA7 TIOCB6	TIOCA8 TIOCB8	TIOCA9 TIOCB9 TIOCC9 TIOCD9	TIOCA10 TIOCB10	TIOCA11 TIOCB11
Counter clear function		TGR compare match or input capture	TGR compare match or input capture	TGR compare match or input capture	TGR compare match or input capture	TGR compare match or input capture	TGR compare match or input capture
Compare match output	0 output	O	O	O	O	O	O
	1 output	O	O	O	O	O	O
	Toggle output	O	O	O	O	O	O
Input capture function		O	O	O	O	O	O
Synchronous operation		O	O	O	O	O	O
PWM mode		O	O	O	O	O	O
Phase counting mode		—	O	O	—	O	O
Buffer operation		O	—	—	O	—	—
DTC activation		TGR compare match or input capture	TGR compare match or input capture	TGR compare match or input capture	TGR compare match or input capture	TGR compare match or input capture	TGR compare match or input capture

Item	Channel 6	Channel 7	Channel 8	Channel 9	Channel 10	Channel 11
DMAC activation	TGRA_6 compare match or input capture	TGRA_7 compare match or input capture	TGRA_8 compare match or input capture	TGRA_9 compare match or input capture	TGRA_10 compare match or input capture	TGRA_11 compare match or input capture
Interrupt sources	5 sources	4 sources	4 sources	5 sources	4 sources	4 sources
	Compare match or input capture 6A	Compare match or input capture 7A	Compare match or input capture 8A	Compare match or input capture 9A	Compare match or input capture 10A	Compare match or input capture 11A
	Compare match or input capture 6B	Compare match or input capture 7B	Compare match or input capture 8B	Compare match or input capture 9B	Compare match or input capture 10B	Compare match or input capture 11B
	Compare match or input capture 6C			Compare match or input capture 9C		
	Compare match or input capture 6D			Compare match or input capture 9D		
	Overflow	Overflow	Overflow	Overflow	Overflow	Overflow
		Underflow	Underflow		Underflow	Underflow

[Legend]

O: Possible

—: Not possible

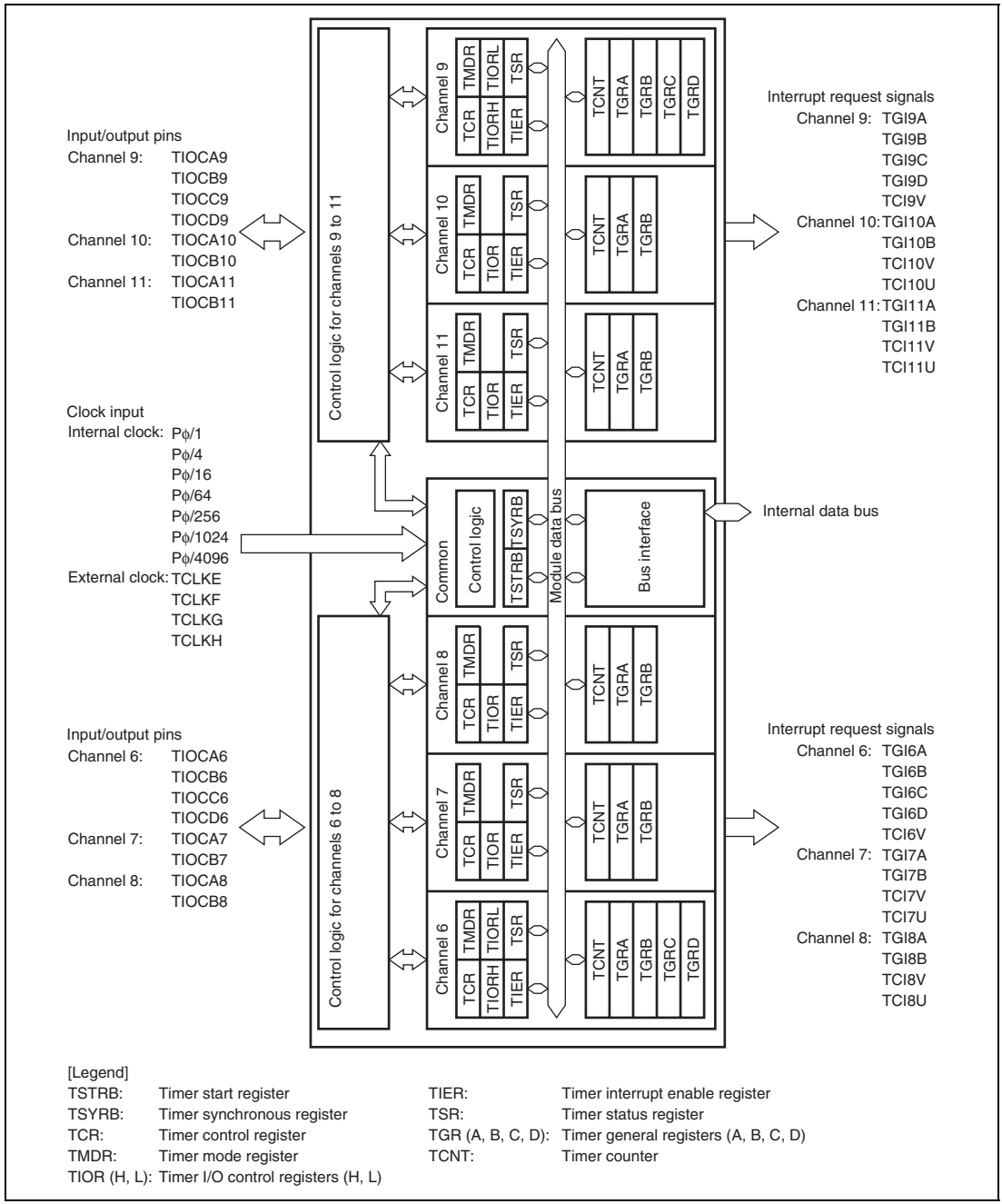


[Legend]

TSTR:	Timer start register	TIER:	Timer interrupt enable register
TSYR:	Timer synchronous register	TSR:	Timer status register
TCR:	Timer control register	TGR (A, B, C, D):	Timer general registers (A, B, C, D)
TMDR:	Timer mode register	TCNT:	Timer counter
TIOR (H, L):	Timer I/O control registers (H, L)		

Note: This LSI does not have the TIOCA3, TIOCB3, TIOCC3, TIOCD3, TIOCA4, TIOCB4, TIOCA5, and TIOCB5 input/output pins on channels 3, 4, and 5. Therefore, the input capture input, waveform output on compare match (0, 1, or toggle output), and PWM waveform output are not available.

Figure 10.1 Block Diagram of TPU (Unit 0)



[Legend]

TSTRB:	Timer start register	TIER:	Timer interrupt enable register
TSYRB:	Timer synchronous register	TSR:	Timer status register
TCR:	Timer control register	TGR (A, B, C, D):	Timer general registers (A, B, C, D)
TMDR:	Timer mode register	TCNT:	Timer counter
TIOR (H, L):	Timer I/O control registers (H, L)		

Figure 10.2 Block Diagram of TPU (Unit 1)

10.2 Input/Output Pins

Table 10.3 shows TPU pin configurations.

Table 10.3 Pin Configuration

Unit	Channel	Symbol	I/O	Function
0	All	TCLKA	Input	External clock A input pin (Channel 1 and 5 phase counting mode A phase input)
		TCLKB	Input	External clock B input pin (Channel 1 and 5 phase counting mode B phase input)
		TCLKC	Input	External clock C input pin (Channel 2 and 4 phase counting mode A phase input)
		TCLKD	Input	External clock D input pin (Channel 2 and 4 phase counting mode B phase input)
	0	TIOCA0	I/O	TGRA_0 input capture input/output compare output/PWM output pin
		TIOCB0	I/O	TGRB_0 input capture input/output compare output/PWM output pin
		TIOCC0	I/O	TGRC_0 input capture input/output compare output/PWM output pin
		TIOCD0	I/O	TGRD_0 input capture input/output compare output/PWM output pin
	1	TIOCA1	I/O	TGRA_1 input capture input/output compare output/PWM output pin
		TIOCB1	I/O	TGRB_1 input capture input/output compare output/PWM output pin
	2	TIOCA2	I/O	TGRA_2 input capture input/output compare output/PWM output pin
		TIOCB2	I/O	TGRB_2 input capture input/output compare output/PWM output pin
	3	TIOCA3*	I/O	TGRA_3 input capture input/output compare output/PWM output pin
		TIOCB3*	I/O	TGRB_3 input capture input/output compare output/PWM output pin
		TIOCC3*	I/O	TGRC_3 input capture input/output compare output/PWM output pin
		TIOCD3*	I/O	TGRD_3 input capture input/output compare output/PWM output pin
	4	TIOCA4*	I/O	TGRA_4 input capture input/output compare output/PWM output pin
		TIOCB4*	I/O	TGRB_4 input capture input/output compare output/PWM output pin
	5	TIOCA5*	I/O	TGRA_5 input capture input/output compare output/PWM output pin
		TIOCB5*	I/O	TGRB_5 input capture input/output compare output/PWM output pin

Unit	Channel	Symbol	I/O	Function
1	All	TCLKE	Input	External clock E input pin (Channel 7 and 11 phase counting mode A phase input)
		TCLKF	Input	External clock F input pin (Channel 7 and 11 phase counting mode B phase input)
		TCLKG	Input	External clock G input pin (Channel 8 and 10 phase counting mode A phase input)
		TCLKH	Input	External clock H input pin (Channel 8 and 10 phase counting mode B phase input)
	6	TIOCA6	I/O	TGRA_6 input capture input/output compare output/PWM output pin
		TIOCB6	I/O	TGRB_6 input capture input/output compare output/PWM output pin
		TIOCC6	I/O	TGRC_6 input capture input/output compare output/PWM output pin
		TIOCD6	I/O	TGRD_6 input capture input/output compare output/PWM output pin
	7	TIOCA7	I/O	TGRA_7 input capture input/output compare output/PWM output pin
		TIOCB7	I/O	TGRB_7 input capture input/output compare output/PWM output pin
	8	TIOCA8	I/O	TGRA_8 input capture input/output compare output/PWM output pin
		TIOCB8	I/O	TGRB_8 input capture input/output compare output/PWM output pin
	9	TIOCA9	I/O	TGRA_9 input capture input/output compare output/PWM output pin
		TIOCB9	I/O	TGRB_9 input capture input/output compare output/PWM output pin
		TIOCC9	I/O	TGRC_9 input capture input/output compare output/PWM output pin
		TIOCD9	I/O	TGRD_9 input capture input/output compare output/PWM output pin
	10	TIOCA10	I/O	TGRA_10 input capture input/output compare output/PWM output pin
		TIOCB10	I/O	TGRB_10 input capture input/output compare output/PWM output pin
	11	TIOCA11	I/O	TGRA_11 input capture input/output compare output/PWM output pin
		TIOCB11	I/O	TGRB_11 input capture input/output compare output/PWM output pin

Note: * This LSI does not have the TIOCA3, TIOCB3, TIOCC3, TIOCD3, TIOCA4, TIOCB4, TIOCA5, and TIOCB5 input/output pins on channels 3, 4, and 5.

10.3 Register Descriptions

The TPU has the following registers in each channel.

Registers in unit 0 and unit 1 have the same functions. This section gives explanations regarding unit 0.

Unit 0

- Channel 0
  - Timer control register_0 (TCR_0)
  - Timer mode register_0 (TMDR_0)
  - Timer I/O control register H_0 (TIORH_0)
  - Timer I/O control register L_0 (TIORL_0)
  - Timer interrupt enable register_0 (TIER_0)
  - Timer status register_0 (TSR_0)
  - Timer counter_0 (TCNT_0)
  - Timer general register A_0 (TGRA_0)
  - Timer general register B_0 (TGRB_0)
  - Timer general register C_0 (TGRC_0)
  - Timer general register D_0 (TGRD_0)
- Channel 1
  - Timer control register_1 (TCR_1)
  - Timer mode register_1 (TMDR_1)
  - Timer I/O control register _1 (TIOR_1)
  - Timer interrupt enable register_1 (TIER_1)
  - Timer status register_1 (TSR_1)
  - Timer counter_1 (TCNT_1)
  - Timer general register A_1 (TGRA_1)
  - Timer general register B_1 (TGRB_1)

- Channel 2
  - Timer control register_2 (TCR_2)
  - Timer mode register_2 (TMDR_2)
  - Timer I/O control register_2 (TIOR_2)
  - Timer interrupt enable register_2 (TIER_2)
  - Timer status register_2 (TSR_2)
  - Timer counter_2 (TCNT_2)
  - Timer general register A_2 (TGRA_2)
  - Timer general register B_2 (TGRB_2)
- Channel 3
  - Timer control register_3 (TCR_3)
  - Timer mode register_3 (TMDR_3)
  - Timer I/O control register H_3 (TIORH_3)
  - Timer I/O control register L_3 (TIORL_3)
  - Timer interrupt enable register_3 (TIER_3)
  - Timer status register_3 (TSR_3)
  - Timer counter_3 (TCNT_3)
  - Timer general register A_3 (TGRA_3)
  - Timer general register B_3 (TGRB_3)
  - Timer general register C_3 (TGRC_3)
  - Timer general register D_3 (TGRD_3)
- Channel 4
  - Timer control register_4 (TCR_4)
  - Timer mode register_4 (TMDR_4)
  - Timer I/O control register _4 (TIOR_4)
  - Timer interrupt enable register_4 (TIER_4)
  - Timer status register_4 (TSR_4)
  - Timer counter_4 (TCNT_4)
  - Timer general register A_4 (TGRA_4)
  - Timer general register B_4 (TGRB_4)

- Channel 5
  - Timer control register_5 (TCR_5)
  - Timer mode register_5 (TMDR_5)
  - Timer I/O control register_5 (TIOR_5)
  - Timer interrupt enable register_5 (TIER_5)
  - Timer status register_5 (TSR_5)
  - Timer counter_5 (TCNT_5)
  - Timer general register A_5 (TGRA_5)
  - Timer general register B_5 (TGRB_5)
- Common Registers
  - Timer start register (TSTR)
  - Timer synchronous register (TSYR)

Unit 1

- Channel 6
  - Timer control register_6 (TCR_6)
  - Timer mode register_6 (TMDR_6)
  - Timer I/O control register H_6 (TIORH_6)
  - Timer I/O control register L_6 (TIORL_6)
  - Timer interrupt enable register_6 (TIER_6)
  - Timer status register_6 (TSR_6)
  - Timer counter_6 (TCNT_6)
  - Timer general register A_6 (TGRA_6)
  - Timer general register B_6 (TGRB_6)
  - Timer general register C_6 (TGRC_6)
  - Timer general register D_6 (TGRD_6)

- Channel 7
  - Timer control register_7 (TCR_7)
  - Timer mode register_7 (TMDR_7)
  - Timer I/O control register _7 (TIOR_7)
  - Timer interrupt enable register_7 (TIER_7)
  - Timer status register_7 (TSR_7)
  - Timer counter_7 (TCNT_7)
  - Timer general register A_7 (TGRA_7)
  - Timer general register B_7 (TGRB_7)
- Channel 8
  - Timer control register_8 (TCR_8)
  - Timer mode register_8 (TMDR_8)
  - Timer I/O control register_8 (TIOR_8)
  - Timer interrupt enable register_8 (TIER_8)
  - Timer status register_8 (TSR_8)
  - Timer counter_8 (TCNT_8)
  - Timer general register A_8 (TGRA_8)
  - Timer general register B_8 (TGRB_8)
- Channel 9
  - Timer control register_9 (TCR_9)
  - Timer mode register_9 (TMDR_9)
  - Timer I/O control register H_9 (TIORH_9)
  - Timer I/O control register L_9 (TIORL_9)
  - Timer interrupt enable register_9 (TIER_9)
  - Timer status register_9 (TSR_9)
  - Timer counter_9 (TCNT_9)
  - Timer general register A_9 (TGRA_9)
  - Timer general register B_9 (TGRB_9)
  - Timer general register C_9 (TGRC_9)
  - Timer general register D_9 (TGRD_9)

- Channel 10
  - Timer control register_10 (TCR_10)
  - Timer mode register_10 (TMDR_10)
  - Timer I/O control register _10 (TIOR_10)
  - Timer interrupt enable register_10 (TIER_10)
  - Timer status register_10 (TSR_10)
  - Timer counter_10 (TCNT_10)
  - Timer general register A_10 (TGRA_10)
  - Timer general register B_10 (TGRB_10)
- Channel 11
  - Timer control register_11 (TCR_11)
  - Timer mode register_11 (TMDR_11)
  - Timer I/O control register_11 (TIOR_11)
  - Timer interrupt enable register_11 (TIER_11)
  - Timer status register_11 (TSR_11)
  - Timer counter_11 (TCNT_11)
  - Timer general register A_11 (TGRA_11)
  - Timer general register B_11 (TGRB_11)
- Common Registers
  - Timer start register (TSTRB)
  - Timer synchronous register (TSYRB)

10.3.1 Timer Control Register (TCR)

TCR controls the TCNT operation for each channel. The TPU has a total of six TCR registers, one for each channel. TCR register settings should be made only while TCNT operation is stopped.

Bit	7	6	5	4	3	2	1	0
Bit Name	CCLR2	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	CCLR2	0	R/W	Counter Clear 2 to 0
6	CCLR1	0	R/W	These bits select the TCNT counter clearing source. See tables 10.4 and 10.5 for details.
5	CCLR0	0	R/W	
4	CKEG1	0	R/W	Clock Edge 1 and 0
3	CKEG0	0	R/W	These bits select the input clock edge. For details, see table 10.6. When the input clock is counted using both edges, the input clock period is halved (e.g. Pϕ/4 both edges = Pϕ/2 rising edge). If phase counting mode is used on channels 1, 2, 4, and 5, this setting is ignored and the phase counting mode setting has priority. Internal clock edge selection is valid when the input clock is Pϕ/4 or slower. This setting is ignored if the input clock is Pϕ/1, or when overflow/underflow of another channel is selected.
2	TPSC2	0	R/W	Timer Prescaler 2 to 0
1	TPSC1	0	R/W	These bits select the TCNT counter clock. The clock source can be selected independently for each channel. See tables 10.7 to 10.12 for details. To select the external clock as the clock source, the DDR bit and ICR bit for the corresponding pin should be set to 0 and 1, respectively. For details, see section 9, I/O Ports.
0	TPSC0	0	R/W	

Table 10.4 CCLR2 to CCLR0 (Channels 0 and 3)

Channel	Bit 7 CCLR2	Bit 6 CCLR1	Bit 5 CCLR0	Description
0, 3	0	0	0	TCNT clearing disabled
	0	0	1	TCNT cleared by TGRA compare match/input capture
	0	1	0	TCNT cleared by TGRB compare match/input capture
	0	1	1	TCNT cleared by counter clearing for another channel performing synchronous clearing/ synchronous operation*1
	1	0	0	TCNT clearing disabled
	1	0	1	TCNT cleared by TGRC compare match/input capture*2
	1	1	0	TCNT cleared by TGRD compare match/input capture*2
	1	1	1	TCNT cleared by counter clearing for another channel performing synchronous clearing/ synchronous operation*1

Notes: 1. Synchronous operation is selected by setting the SYNC bit in TSYR to 1.
2. When TGRC or TGRD is used as a buffer register, TCNT is not cleared because the buffer register setting has priority and compare match/input capture does not occur.

Table 10.5 CCLR2 to CCLR0 (Channels 1, 2, 4, and 5)

Channel	Bit 7 Reserved*2	Bit 6 CCLR1	Bit 5 CCLR0	Description
1, 2, 4, 5	0	0	0	TCNT clearing disabled
	0	0	1	TCNT cleared by TGRA compare match/input capture
	0	1	0	TCNT cleared by TGRB compare match/input capture
	0	1	1	TCNT cleared by counter clearing for another channel performing synchronous clearing/ synchronous operation*1

Notes: 1. Synchronous operation is selected by setting the SYNC bit in TSYR to 1.
2. Bit 7 is reserved in channels 1, 2, 4, and 5. It is always read as 0 and cannot be modified.

Table 10.6 Input Clock Edge Selection

Clock Edge Selection		Input Clock	
CKEG1	**CKEG0**	**Internal Clock**	**External Clock**
0	0	Counted at falling edge	Counted at rising edge
0	1	Counted at rising edge	Counted at falling edge
1	X	Counted at both edges	Counted at both edges

[Legend]

X: Don't care

Table 10.7 TPSC2 to TPSC0 (Channel 0)

Channel	Bit 2 TPSC2	Bit 1 TPSC1	Bit 0 TPSC0	Description
0	0	0	0	Internal clock: counts on Pϕ/1
	0	0	1	Internal clock: counts on Pϕ/4
	0	1	0	Internal clock: counts on Pϕ/16
	0	1	1	Internal clock: counts on Pϕ/64
	1	0	0	External clock: counts on TCLKA pin input
	1	0	1	External clock: counts on TCLKB pin input
	1	1	0	External clock: counts on TCLKC pin input
	1	1	1	External clock: counts on TCLKD pin input

Table 10.8 TPSC2 to TPSC0 (Channel 1)

Channel	Bit 2 TPSC2	Bit 1 TPSC1	Bit 0 TPSC0	Description
1	0	0	0	Internal clock: counts on Pϕ/1
	0	0	1	Internal clock: counts on Pϕ/4
	0	1	0	Internal clock: counts on Pϕ/16
	0	1	1	Internal clock: counts on Pϕ/64
	1	0	0	External clock: counts on TCLKA pin input
	1	0	1	External clock: counts on TCLKB pin input
	1	1	0	Internal clock: counts on Pϕ/256
	1	1	1	Counts on TCNT2 overflow/underflow

Note: This setting is ignored when channel 1 is in phase counting mode.

Table 10.9 TPSC2 to TPSC0 (Channel 2)

Channel	Bit 2 TPSC2	Bit 1 TPSC1	Bit 0 TPSC0	Description
2	0	0	0	Internal clock: counts on Pϕ/1
	0	0	1	Internal clock: counts on Pϕ/4
	0	1	0	Internal clock: counts on Pϕ/16
	0	1	1	Internal clock: counts on Pϕ/64
	1	0	0	External clock: counts on TCLKA pin input
	1	0	1	External clock: counts on TCLKB pin input
	1	1	0	External clock: counts on TCLKC pin input
	1	1	1	Internal clock: counts on Pϕ/1024

Note: This setting is ignored when channel 2 is in phase counting mode.

Table 10.10 TPSC2 to TPSC0 (Channel 3)

Channel	Bit 2 TPSC2	Bit 1 TPSC1	Bit 0 TPSC0	Description
3	0	0	0	Internal clock: counts on Pϕ/1
	0	0	1	Internal clock: counts on Pϕ/4
	0	1	0	Internal clock: counts on Pϕ/16
	0	1	1	Internal clock: counts on Pϕ/64
	1	0	0	External clock: counts on TCLKA pin input
	1	0	1	Internal clock: counts on Pϕ/1024
	1	1	0	Internal clock: counts on Pϕ/256
	1	1	1	Internal clock: counts on Pϕ/4096

Table 10.11 TPSC2 to TPSC0 (Channel 4)

Channel	Bit 2 TPSC2	Bit 1 TPSC1	Bit 0 TPSC0	Description
4	0	0	0	Internal clock: counts on Pϕ/1
	0	0	1	Internal clock: counts on Pϕ/4
	0	1	0	Internal clock: counts on Pϕ/16
	0	1	1	Internal clock: counts on Pϕ/64
	1	0	0	External clock: counts on TCLKA pin input
	1	0	1	External clock: counts on TCLKC pin input
	1	1	0	Internal clock: counts on Pϕ/1024
	1	1	1	Counts on TCNT5 overflow/underflow

Note: This setting is ignored when channel 4 is in phase counting mode.

Table 10.12 TPSC2 to TPSC0 (Channel 5)

Channel	Bit 2 TPSC2	Bit 1 TPSC1	Bit 0 TPSC0	Description
5	0	0	0	Internal clock: counts on Pϕ/1
	0	0	1	Internal clock: counts on Pϕ/4
	0	1	0	Internal clock: counts on Pϕ/16
	0	1	1	Internal clock: counts on Pϕ/64
	1	0	0	External clock: counts on TCLKA pin input
	1	0	1	External clock: counts on TCLKC pin input
	1	1	0	Internal clock: counts on Pϕ/256
	1	1	1	External clock: counts on TCLKD pin input

Note: This setting is ignored when channel 5 is in phase counting mode.

10.3.2 Timer Mode Register (TMDR)

TMDR sets the operating mode for each channel. The TPU has six TMDR registers, one for each channel. TMDR register settings should be made only while TCNT operation is stopped.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	BFB	BFA	MD3	MD2	MD1	MD0
Initial Value	1	1	0	0	0	0	0	0
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7, 6	—	All 1	R	Reserved These bits are always read as 1 and cannot be modified.
5	BFB	0	R/W	Buffer Operation B Specifies whether TGRB is to normally operate, or TGRB and TGRD are to be used together for buffer operation. When TGRD is used as a buffer register, TGRD input capture/output compare is not generated. In channels 1, 2, 4, and 5, which have no TGRD, bit 5 is reserved. It is always read as 0 and cannot be modified. 0: TGRB operates normally 1: TGRB and TGRD used together for buffer operation
4	BFA	0	R/W	Buffer Operation A Specifies whether TGRA is to normally operate, or TGRA and TGRC are to be used together for buffer operation. When TGRC is used as a buffer register, TGRC input capture/output compare is not generated. In channels 1, 2, 4, and 5, which have no TGRC, bit 4 is reserved. It is always read as 0 and cannot be modified. 0: TGRA operates normally 1: TGRA and TGRC used together for buffer operation
3	MD3	0	R/W	Modes 3 to 0 Set the timer operating mode. MD3 is a reserved bit. The write value should always be 0. See table 10.13 for details.
2	MD2	0	R/W	
1	MD1	0	R/W	
0	MD0	0	R/W	

Table 10.13 MD3 to MD0

Bit 3 MD3*[1]	Bit 2 MD2*[2]	Bit 1 MD1	Bit 0 MD0	Description	
0	0	0	0	Normal operation	
0	0	0	1	Reserved	(setting prohibited)
0	0	1	0	PWM mode 1	
0	0	1	1	PWM mode 2	
0	1	0	0	Phase counting mode 1	
0	1	0	1	Phase counting mode 2	
0	1	1	0	Phase counting mode 3	
0	1	1	1	Phase counting mode 4	
1	X	X	X	—	(setting prohibited)

[Legend]

X: Don't care

Notes: 1. MD3 is a reserved bit. The write value should always be 0.

2. Phase counting mode cannot be set for channels 0 and 3. In this case, 0 should always be written to MD2.

10.3.3 Timer I/O Control Register (TIOR)

TIOR controls TGR. The TPU has eight TIOR registers, two each for channels 0 and 3, and one each for channels 1, 2, 4, and 5. Set TIOR while TCNT operation is stopped. Note that TIOR is affected by the TMDR setting.

The initial output specified by TIOR is valid when the counter is stopped (the CST bit in TSTR is cleared to 0). In PWM mode 2, the output at the point at which the counter is cleared to 0 is specified.

When TGRC or TGRD is designated for buffer operation, this setting is invalid and the register operates as a buffer register.

- TIORH_0, TIOR_1, TIOR_2, TIORH_3, TIOR_4, TIOR_5

Bit	7	6	5	4	3	2	1	0
Bit Name	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- TIORL_0, TORL_3

Bit	7	6	5	4	3	2	1	0
Bit Name	IOD3	IOD2	IOD1	IOD0	IOC3	IOC2	IOC1	IOC0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- TIORH_0, TIOR_1, TIOR_2, TIORH_3, TIOR_4, TIOR_5

Bit	Bit Name	Initial Value	R/W	Description
7	IOB3	0	R/W	I/O Control B3 to B0
6	IOB2	0	R/W	Specify the function of TGRB.
5	IOB1	0	R/W	For details, see tables 10.14, 10.16 to 10.18, 10.20, and 10.21.
4	IOB0	0	R/W	
3	IOA3	0	R/W	I/O Control A3 to A0
2	IOA2	0	R/W	Specify the function of TGRA.
1	IOA1	0	R/W	For details, see tables 10.22, 10.24 to 10.26, 10.28, and 10.29.
0	IOA0	0	R/W	

- TIORL_0, TIORL_3

Bit	Bit Name	Initial Value	R/W	Description
7	IOD3	0	R/W	I/O Control D3 to D0
6	IOD2	0	R/W	Specify the function of TGRD.
5	IOD1	0	R/W	For details, see tables 10.15 and 10.19.
4	IOD0	0	R/W	
3	IOC3	0	R/W	I/O Control C3 to C0
2	IOC2	0	R/W	Specify the function of TGRC.
1	IOC1	0	R/W	For details, see tables 10.23 and 10.27.
0	IOC0	0	R/W	

Table 10.14 TIORH_0

				Description	
Bit 7 IOB3	**Bit 6 IOB2**	**Bit 5 IOB1**	**Bit 4 IOB0**	**TGRB_0 Function**	**TIOCB0 Pin Function**
0	0	0	0	Output compare register	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	0	0	0	Input capture register	Capture input source is TIOCB0 pin Input capture at rising edge
1	0	0	1		Capture input source is TIOCB0 pin Input capture at falling edge
1	0	1	X		Capture input source is TIOCB0 pin Input capture at both edges
1	1	X	X		Capture input source is channel 1/count clock Input capture at TCNT_1 count-up/count-down*

[Legend]

X: Don't care

Note: When bits TPSC2 to TPSC0 in TCR_1 are set to B'000 and Pϕ/1 is used as the TCNT_1 count clock, this setting is invalid and input capture is not generated.

Table 10.15 TIORL_0

				Description	
Bit 7 IOD3	**Bit 6 IOD2**	**Bit 5 IOD1**	**Bit 4 IOD0**	**TGRD_0 Function**	**TIOCD0 Pin Function**
0	0	0	0	Output compare register*2	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	0	0	0	Input capture register*2	Capture input source is TIOCD0 pin Input capture at rising edge
1	0	0	1		Capture input source is TIOCD0 pin Input capture at falling edge
1	0	1	X		Capture input source is TIOCD0 pin Input capture at both edges
1	1	X	X		Capture input source is channel 1/count clock Input capture at TCNT_1 count-up/count-down*1

[Legend]

X: Don't care

Notes: 1. When bits TPSC2 to TPSC0 in TCR_1 are set to B'000 and Pϕ/1 is used as the TCNT_1 count clock, this setting is invalid and input capture is not generated.

2. When the BFB bit in TMDR_0 is set to 1 and TGRD_0 is used as a buffer register, input capture/output compare is not generated. In this case, do not select 0, 1, or toggle output at compare match.

Table 10.16 TIOR_1

				Description	
Bit 7 IOB3	**Bit 6 IOB2**	**Bit 5 IOB1**	**Bit 4 IOB0**	**TGRB_1 Function**	**TIOCB1 Pin Function**
0	0	0	0	Output compare register	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	0	0	0	Input capture register	Capture input source is TIOCB1 pin Input capture at rising edge
1	0	0	1		Capture input source is TIOCB1 pin Input capture at falling edge
1	0	1	X		Capture input source is TIOCB1 pin Input capture at both edges
1	1	X	X		TGRC_0 compare match/input capture Input capture at generation of TGRC_0 compare match/input capture

[Legend]

X: Don't care

Table 10.17 TIOR_2

				Description	
Bit 7 IOB3	**Bit 6 IOB2**	**Bit 5 IOB1**	**Bit 4 IOB0**	**TGRB_2 Function**	**TIOCB2 Pin Function**
0	0	0	0	Output compare register	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	X	0	0	Input capture register	Capture input source is TIOCB2 pin Input capture at rising edge
1	X	0	1		Capture input source is TIOCB2 pin Input capture at falling edge
1	X	1	X		Capture input source is TIOCB2 pin Input capture at both edges

[Legend]

X: Don't care

Table 10.18 TIORH_3

				Description	
Bit 7 IOB3	**Bit 6 IOB2**	**Bit 5 IOB1**	**Bit 4 IOB0**	**TGRB_3 Function**	**TIOCB3 Pin Function**
0	0	0	0	Output compare register	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	0	0	0	Input capture register	Capture input source is TIOCB3 pin Input capture at rising edge
1	0	0	1		Capture input source is TIOCB3 pin Input capture at falling edge
1	0	1	X		Capture input source is TIOCB3 pin Input capture at both edges
1	1	X	X		Capture input source is channel 4/count clock Input capture at TCNT_4 count-up/count-down*

[Legend]

X: Don't care

Note: When bits TPSC2 to TPSC0 in TCR_4 are set to B'000 and Pϕ/1 is used as the TCNT_4 count clock, this setting is invalid and input capture is not generated.

Table 10.19 TIORL_3

				Description	
Bit 7 IOD3	**Bit 6 IOD2**	**Bit 5 IOD1**	**Bit 4 IOD0**	**TGRD_3 Function**	**TIOCD3 Pin Function**
0	0	0	0	Output compare register*2	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	0	0	0	Input capture register*2	Capture input source is TIOCD3 pin Input capture at rising edge
1	0	0	1		Capture input source is TIOCD3 pin Input capture at falling edge
1	0	1	X		Capture input source is TIOCD3 pin Input capture at both edges
1	1	X	X		Capture input source is channel 4/count clock Input capture at TCNT_4 count-up/count-down*1

[Legend]

X: Don't care

Notes: 1. When bits TPSC2 to TPSC0 in TCR_4 are set to B'000 and Pϕ/1 is used as the TCNT_4 count clock, this setting is invalid and input capture is not generated.

2. When the BFB bit in TMDR_3 is set to 1 and TGRD_3 is used as a buffer register, input capture/output compare is not generated. In this case, do not select 0, 1, or toggle output at compare match.

Table 10.20 TIOR_4

				Description	
Bit 7 IOB3	**Bit 6 IOB2**	**Bit 5 IOB1**	**Bit 4 IOB0**	**TGRB_4 Function**	**TIOCB4 Pin Function**
0	0	0	0	Output compare register	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	0	0	0	Input capture register	Capture input source is TIOCB4 pin Input capture at rising edge
1	0	0	1		Capture input source is TIOCB4 pin Input capture at falling edge
1	0	1	X		Capture input source is TIOCB4 pin Input capture at both edges
1	1	X	X		Capture input source is TGRC_3 compare match/input capture Input capture at generation of TGRC_3 compare match/input capture

[Legend]

X: Don't care

Table 10.21 TIOR_5

				Description	
Bit 7 IOB3	**Bit 6 IOB2**	**Bit 5 IOB1**	**Bit 4 IOB0**	**TGRB_5 Function**	**TIOCB5 Pin Function**
0	0	0	0	Output compare register	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	X	0	0	Input capture register	Capture input source is TIOCB5 pin Input capture at rising edge
1	X	0	1		Capture input source is TIOCB5 pin Input capture at falling edge
1	X	1	X		Capture input source is TIOCB5 pin Input capture at both edges

[Legend]

X: Don't care

Table 10.22 TIORH_0

				Description	
Bit 3 IOA3	**Bit 2 IOA2**	**Bit 1 IOA1**	**Bit 0 IOA0**	**TGRA_0 Function**	**TIOCA0 Pin Function**
0	0	0	0	Output compare register	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	0	0	0	Input capture register	Capture input source is TIOCA0 pin Input capture at rising edge
1	0	0	1		Capture input source is TIOCA0 pin Input capture at falling edge
1	0	1	X		Capture input source is TIOCA0 pin Input capture at both edges
1	1	X	X		Capture input source is channel 1/count clock Input capture at TCNT_1 count-up/count-down

[Legend]

X: Don't care

Table 10.23 TIORL_0

				Description	
Bit 3 IOC3	**Bit 2 IOC2**	**Bit 1 IOC1**	**Bit 0 IOC0**	**TGRC_0 Function**	**TIOCC0 Pin Function**
0	0	0	0	Output compare register*	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	0	0	0	Input capture register*	Capture input source is TIOCC0 pin Input capture at rising edge
1	0	0	1		Capture input source is TIOCC0 pin Input capture at falling edge
1	0	1	X		Capture input source is TIOCC0 pin Input capture at both edges
1	1	X	X		Capture input source is channel 1/count clock Input capture at TCNT_1 count-up/count-down

[Legend]

X: Don't care

Note: When the BFA bit in TMDR_0 is set to 1 and TGRC_0 is used as a buffer register, input capture/output compare is not generated. In this case, do not select 0, 1, or toggle output at compare match.

Table 10.24 TIOR_1

				Description	
Bit 3 IOA3	**Bit 2 IOA2**	**Bit 1 IOA1**	**Bit 0 IOA0**	**TGRA_1 Function**	**TIOCA1 Pin Function**
0	0	0	0	Output compare register	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	0	0	0	Input capture register	Capture input source is TIOCA1 pin Input capture at rising edge
1	0	0	1		Capture input source is TIOCA1 pin Input capture at falling edge
1	0	1	X		Capture input source is TIOCA1 pin Input capture at both edges
1	1	X	X		Capture input source is TGRA_0 compare match/input capture Input capture at generation of channel 0/TGRA_0 compare match/input capture

[Legend]

X: Don't care

Table 10.25 TIOR_2

				Description	
Bit 3 IOA3	**Bit 2 IOA2**	**Bit 1 IOA1**	**Bit 0 IOA0**	**TGRA_2 Function**	**TIOCA2 Pin Function**
0	0	0	0	Output compare register	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	X	0	0	Input capture register	Capture input source is TIOCA2 pin Input capture at rising edge
1	X	0	1		Capture input source is TIOCA2 pin Input capture at falling edge
1	X	1	X		Capture input source is TIOCA2 pin Input capture at both edges

[Legend]

X: Don't care

Table 10.26 TIORH_3

				Description	
Bit 3 IOA3	**Bit 2 IOA2**	**Bit 1 IOA1**	**Bit 0 IOA0**	**TGRA_3 Function**	**TIOCA3 Pin Function**
0	0	0	0	Output compare register	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	0	0	0	Input capture register	Capture input source is TIOCA3 pin Input capture at rising edge
1	0	0	1		Capture input source is TIOCA3 pin Input capture at falling edge
1	0	1	X		Capture input source is TIOCA3 pin Input capture at both edges
1	1	X	X		Capture input source is channel 4/count clock Input capture at TCNT_4 count-up/count-down

[Legend]

X: Don't care

Table 10.27 TIORL_3

				Description	
Bit 3 IOC3	**Bit 2 IOC2**	**Bit 1 IOC1**	**Bit 0 IOC0**	**TGRC_3 Function**	**TIOCC3 Pin Function**
0	0	0	0	Output compare register*	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	0	0	0	Input capture register*	Capture input source is TIOCC3 pin Input capture at rising edge
1	0	0	1		Capture input source is TIOCC3 pin Input capture at falling edge
1	0	1	X		Capture input source is TIOCC3 pin Input capture at both edges
1	1	X	X		Capture input source is channel 4/count clock Input capture at TCNT_4 count-up/count-down

[Legend]

X: Don't care

Note: When the BFA bit in TMDR_3 is set to 1 and TGRC_3 is used as a buffer register, input capture/output compare is not generated. In this case, do not select 0, 1, or toggle output at compare match.

Table 10.28 TIOR_4

				Description	
Bit 3 IOA3	**Bit 2 IOA2**	**Bit 1 IOA1**	**Bit 0 IOA0**	**TGRA_4 Function**	**TIOCA4 Pin Function**
0	0	0	0	Output compare register	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	0	0	0	Input capture register	Capture input source is TIOCA4 pin Input capture at rising edge
1	0	0	1		Capture input source is TIOCA4 pin Input capture at falling edge
1	0	1	X		Capture input source is TIOCA4 pin Input capture at both edges
1	1	X	X		Capture input source is TGRA_3 compare match/input capture Input capture at generation of TGRA_3 compare match/input capture

[Legend]

X: Don't care

Table 10.29 TIOR_5

				Description	
Bit 3 IOA3	**Bit 2 IOA2**	**Bit 1 IOA1**	**Bit 0 IOA0**	**TGRA_5 Function**	**TIOCA5 Pin Function**
0	0	0	0	Output compare register	Output disabled
0	0	0	1		Initial output is 0 output 0 output at compare match
0	0	1	0		Initial output is 0 output 1 output at compare match
0	0	1	1		Initial output is 0 output Toggle output at compare match
0	1	0	0		Output disabled
0	1	0	1		Initial output is 1 output 0 output at compare match
0	1	1	0		Initial output is 1 output 1 output at compare match
0	1	1	1		Initial output is 1 output Toggle output at compare match
1	X	0	0	Input capture register	Input capture source is TIOCA5 pin Input capture at rising edge
1	X	0	1		Input capture source is TIOCA5 pin Input capture at falling edge
1	X	1	X		Input capture source is TIOCA5 pin Input capture at both edges

[Legend]

X: Don't care

10.3.4 Timer Interrupt Enable Register (TIER)

TIER controls enabling or disabling of interrupt requests for each channel. The TPU has six TIER registers, one for each channel.

Bit	7	6	5	4	3	2	1	0
Bit Name	TTGE	—	TCIEU	TCIEV	TGIED	TCIEC	TGIEB	TGIEA
Initial Value	0	1	0	0	0	0	0	0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial value	R/W	Description
7	TTGE	0	R/W	A/D Conversion Start Request Enable Enables or disables generation of A/D conversion start requests by TGRA input capture/compare match. 0: Disables A/D conversion start request generation 1: Enables A/D conversion start request generation
6	—	1	R	Reserved This bit is always read as 1 and cannot be modified.
5	TCIEU	0	R/W	Underflow Interrupt Enable Enables or disables interrupt requests (TCIU) by the TCFU flag when the TCFU flag in TSR is set to 1 in channels 1, 2, 4, and 5. In channels 0 and 3, bit 5 is reserved. It is always read as 0 and cannot be modified. 0: Disables interrupt requests (TCIU) by TCFU 1: Enables interrupt requests (TCIU) by TCFU
4	TCIEV	0	R/W	Overflow Interrupt Enable Enables or disables interrupt requests (TCIV) by the TCFV flag when the TCFV flag in TSR is set to 1. 0: Disables interrupt requests (TCIV) by TCFV 1: Enables interrupt requests (TCIV) by TCFV

Bit	Bit Name	Initial value	R/W	Description
3	TGIED	0	R/W	TGR Interrupt Enable D Enables or disables interrupt requests (TGID) by the TGFD bit when the TGFD bit in TSR is set to 1 in channels 0 and 3. In channels 1, 2, 4, and 5, bit 3 is reserved. It is always read as 0 and cannot be modified. 0: Disables interrupt requests (TGID) by TGFD bit 1: Enables interrupt requests (TGID) by TGFD bit
2	TGIEC	0	R/W	TGR Interrupt Enable C Enables or disables interrupt requests (TGIC) by the TGFC bit when the TGFC bit in TSR is set to 1 in channels 0 and 3. In channels 1, 2, 4, and 5, bit 2 is reserved. It is always read as 0 and cannot be modified. 0: Disables interrupt requests (TGIC) by TGFC bit 1: Enables interrupt requests (TGIC) by TGFC bit
1	TGIEB	0	R/W	TGR Interrupt Enable B Enables or disables interrupt requests (TGIB) by the TGFB bit when the TGFB bit in TSR is set to 1. 0: Disables interrupt requests (TGIB) by TGFB bit 1: Enables interrupt requests (TGIB) by TGFB bit
0	TGIEA	0	R/W	TGR Interrupt Enable A Enables or disables interrupt requests (TGIA) by the TGFA bit when the TGFA bit in TSR is set to 1. 0: Disables interrupt requests (TGIA) by TGFA bit 1: Enables interrupt requests (TGIA) by TGFA bit

10.3.5 Timer Status Register (TSR)

TSR indicates the status of each channel. The TPU has six TSR registers, one for each channel.

Bit	7	6	5	4	3	2	1	0
Bit Name	TCFD	—	TCFU	TCFV	TGFD	TGFC	TGFB	TGFA
Initial Value	1	1	0	0	0	0	0	0
R/W	R	R	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*

Note: * Only 0 can be written to bits 5 to 0, to clear flags.

Bit	Bit Name	Initial value	R/W	Description
7	TCFD	1	R	Count Direction Flag Status flag that shows the direction in which TCNT counts in channels 1, 2, 4, and 5. In channels 0 and 3, bit 7 is reserved. It is always read as 1 and cannot be modified. 0: TCNT counts down 1: TCNT counts up
6	—	1	R	Reserved This bit is always read as 1 and cannot be modified.
5	TCFU	0	R/(W)*	Underflow Flag Status flag that indicates that a TCNT underflow has occurred when channels 1, 2, 4, and 5 are set to phase counting mode. In channels 0 and 3, bit 5 is reserved. It is always read as 0 and cannot be modified. [Setting condition] When the TCNT value underflows (changes from H'0000 to H'FFFF) [Clearing condition] When a 0 is written to TCFU after reading TCFU = 1 (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.)

Bit	Bit Name	Initial value	R/W	Description
4	TCFV	0	R/(W)*	Overflow Flag Status flag that indicates that a TCNT overflow has occurred. [Setting condition] When the TCNT value overflows (changes from H'FFFF to H'0000) [Clearing condition] When a 0 is written to TCFV after reading TCFV = 1 (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.)
3	TGFD	0	R/(W)*	Input Capture/Output Compare Flag D Status flag that indicates the occurrence of TGRD input capture or compare match in channels 0 and 3. In channels 1, 2, 4, and 5, bit 3 is reserved. It is always read as 0 and cannot be modified. [Setting conditions] • When TCNT = TGRD while TGRD is functioning as output compare register • When TCNT value is transferred to TGRD by input capture signal while TGRD is functioning as input capture register [Clearing conditions] • When DTC is activated by a TGID interrupt while the DISEL bit in MRB of DTC is 0 • When 0 is written to TGFD after reading TGFD = 1 (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.)

Bit	Bit Name	Initial value	R/W	Description
2	TGFC	0	R/(W)*	Input Capture/Output Compare Flag C Status flag that indicates the occurrence of TGRC input capture or compare match in channels 0 and 3. In channels 1, 2, 4, and 5, bit 2 is reserved. It is always read as 0 and cannot be modified. [Setting conditions] • When TCNT = TGRC while TGRC is functioning as output compare register • When TCNT value is transferred to TGRC by input capture signal while TGRC is functioning as input capture register [Clearing conditions] • When DTC is activated by a TGIC interrupt while the DISEL bit in MRB of DTC is 0 • When 0 is written to TGFC after reading TGFC = 1 (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.)
1	TGFB	0	R/(W)*	Input Capture/Output Compare Flag B Status flag that indicates the occurrence of TGRB input capture or compare match. [Setting conditions] • When TCNT = TGRB while TGRB is functioning as output compare register • When TCNT value is transferred to TGRB by input capture signal while TGRB is functioning as input capture register [Clearing conditions] • When DTC is activated by a TGIB interrupt while the DISEL bit in MRB of DTC is 0 • When 0 is written to TGFB after reading TGFB = 1 (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.)

Bit	Bit Name	Initial value	R/W	Description
0	TGFA	0	R/(W)*	Input Capture/Output Compare Flag A Status flag that indicates the occurrence of TGRA input capture or compare match. [Setting conditions] • When TCNT = TGRA while TGRA is functioning as output compare register • When TCNT value is transferred to TGRA by input capture signal while TGRA is functioning as input capture register [Clearing conditions] • When DTC is activated by a TGIA interrupt while the DISEL bit in MRB of DTC is 0 • When 0 is written to TGFA after reading TGFA = 1 (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.)

Note: * Only 0 can be written to clear the flag.

10.3.6 Timer Counter (TCNT)

TCNT is a 16-bit readable/writable counter. The TPU has six TCNT counters, one for each channel.

TCNT is initialized to H'0000 by a reset or in hardware standby mode.

TCNT cannot be accessed in 8-bit units. TCNT must always be accessed in 16-bit units.

Note: This LSI does not have hardware standby mode.

Bit	15	14	13	12	11	10	9	8
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

10.3.7 Timer General Register (TGR)

TGR is a 16-bit readable/writable register with a dual function as output compare and input capture registers. The TPU has 16 TGR registers, four each for channels 0 and 3 and two each for channels 1, 2, 4, and 5. TGRC and TGRD for channels 0 and 3 can also be designated for operation as buffer registers. The TGR registers cannot be accessed in 8-bit units; they must always be accessed in 16-bit units. TGR and buffer register combinations during buffer operations are TGRA–TGRC and TGRB–TGRD.

Bit	15	14	13	12	11	10	9	8
Bit Name								
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

10.3.8 Timer Start Register (TSTR)

TSTR starts or stops operation for channels 0 to 5. When setting the operating mode in TMDR or setting the count clock in TCR, first stop the TCNT counter.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	CST5	CST4	CST3	CST2	CST1	CST0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial value	R/W	Description
7, 6	—	All 0	R/W	Reserved These bits are always read as 0. The write value should always be 0.
5	CST5	0	R/W	Counter Start 5 to 0 These bits select operation or stopping of TCNT. If 0 is written to the CST bit during operation with the TIOC pin designated for output, the counter stops but the TIOC pin output compare output level is retained. If TIOR is written to when the CST bit is cleared to 0, the pin output level will be changed to the set initial output value. 0: TCNT_5 to TCNT_0 count operation is stopped 1: TCNT_5 to TCNT_0 performs count operation
4	CST4	0	R/W	
3	CST3	0	R/W	
2	CST2	0	R/W	
1	CST1	0	R/W	
0	CST0	0	R/W	

10.3.9 Timer Synchronous Register (TSYR)

TSYR selects independent operation or synchronous operation for the TCNT counters of channels 0 to 5. A channel performs synchronous operation when the corresponding bit in TSYR is set to 1.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial value	R/W	Description
7, 6	—	All 0	R/W	Reserved These bits are always read as 0. The write value should always be 0.
5	SYNC5	0	R/W	Timer Synchronization 5 to 0 These bits select whether operation is independent of or synchronized with other channels. When synchronous operation is selected, synchronous presetting of multiple channels, and synchronous clearing through counter clearing on another channel are possible. To set synchronous operation, the SYNC bits for at least two channels must be set to 1. To set synchronous clearing, in addition to the SYNC bit, the TCNT clearing source must also be set by means of bits CCLR2 to CCLR0 in TCR. 0: TCNT_5 to TCNT_0 operate independently (TCNT presetting/clearing is unrelated to other channels) 1: TCNT_5 to TCNT_0 perform synchronous operation (TCNT synchronous presetting/synchronous clearing is possible)
4	SYNC4	0	R/W	
3	SYNC3	0	R/W	
2	SYNC2	0	R/W	
1	SYNC1	0	R/W	
0	SYNC0	0	R/W	

10.4 Operation

10.4.1 Basic Functions

Each channel has a TCNT and TGR register. TCNT performs up-counting, and is also capable of free-running operation, periodic counting, and external event counting.

Each TGR can be used as an input capture register or output compare register.

(1) Counter Operation

When one of bits CST0 to CST5 is set to 1 in TSTR, the TCNT counter for the corresponding channel starts counting. TCNT can operate as a free-running counter or periodic counter.

(a) Example of count operation setting procedure

Figure 10.3 shows an example of the count operation setting procedure.

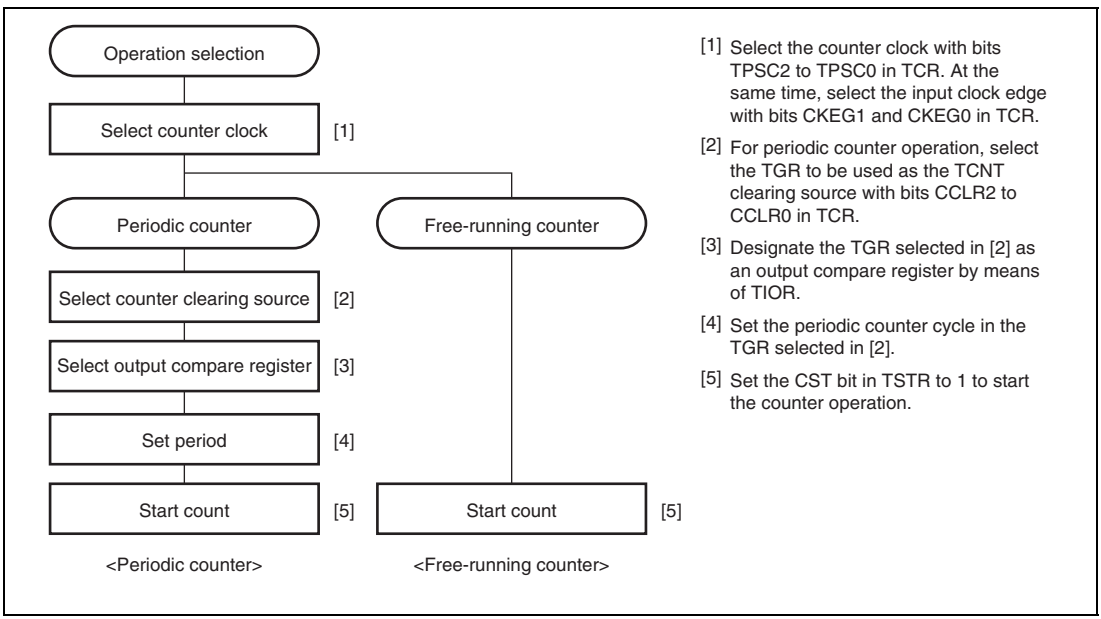


Figure 10.3 Example of Counter Operation Setting Procedure

(b) Free-running count operation and periodic count operation

Immediately after a reset, the TPU's TCNT counters are all designated as free-running counters. When the relevant bit in TSTR is set to 1, the corresponding TCNT counter starts up-count operation as a free-running counter. When TCNT overflows (changes from H'FFFF to H'0000), the TCFV bit in TSR is set to 1. If the value of the corresponding TCIEV bit in TIER is 1 at this point, the TPU requests an interrupt. After overflow, TCNT starts counting up again from H'0000.

Figure 10.4 illustrates free-running counter operation.

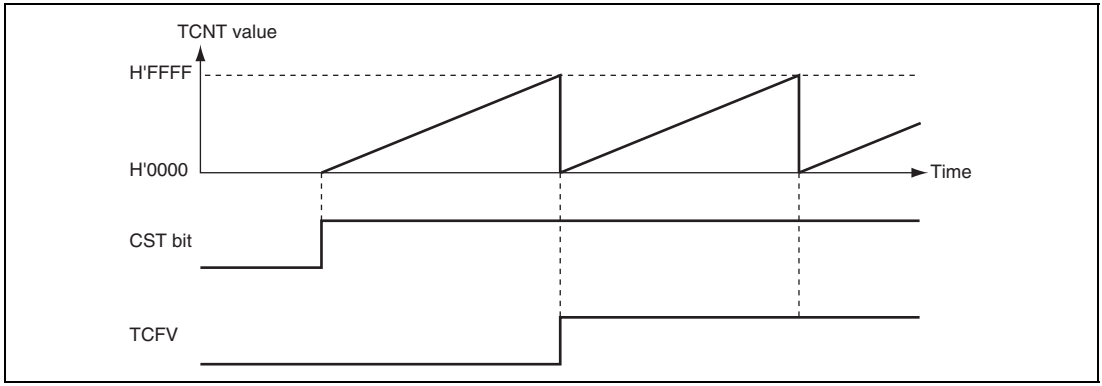


Figure 10.4 Free-Running Counter Operation

When compare match is selected as the TCNT clearing source, the TCNT counter for the relevant channel performs periodic count operation. The TGR register for setting the period is designated as an output compare register, and counter clearing by compare match is selected by means of bits CCLR2 to CCLR0 in TCR. After the settings have been made, TCNT starts count-up operation as a periodic counter when the corresponding bit in TSTR is set to 1. When the count value matches the value in TGR, the TGF bit in TSR is set to 1 and TCNT is cleared to H'0000.

If the value of the corresponding TGIE bit in TIER is 1 at this point, the TPU requests an interrupt. After a compare match, TCNT starts counting up again from H'0000.

Figure 10.5 illustrates periodic counter operation.

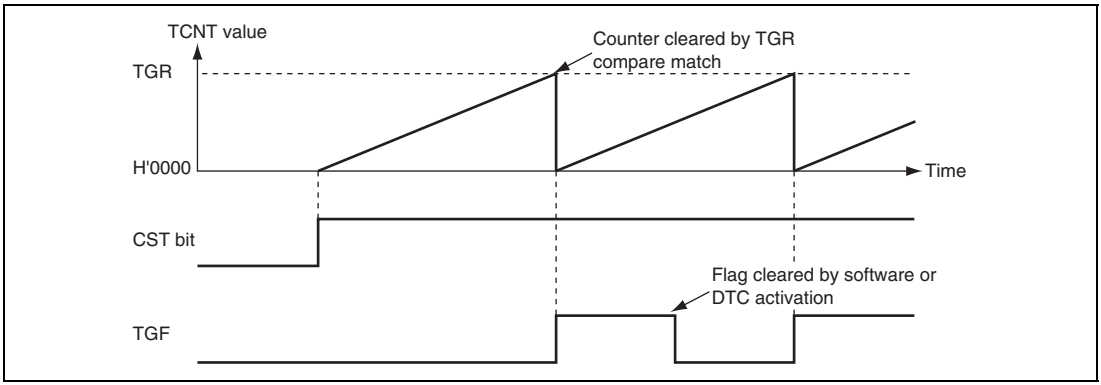


Figure 10.5 Periodic Counter Operation

(2) Waveform Output Function by Compare Match

The TPU can perform 0, 1, or toggle output from the corresponding output pin using a compare match.

Note: This LSI does not have the TIOCA3, TIOCB3, TIOCC3, TIOCD3, TIOCA4, TIOCB4, TIOCA5, and TIOCB5 input/output pins on channels 3, 4, and 5. Therefore, the waveform output on compare match (0, 1, or toggle output), and PWM waveform output are not available.

(a) Example of setting procedure for waveform output by compare match

Figure 10.6 shows an example of the setting procedure for waveform output by a compare match.

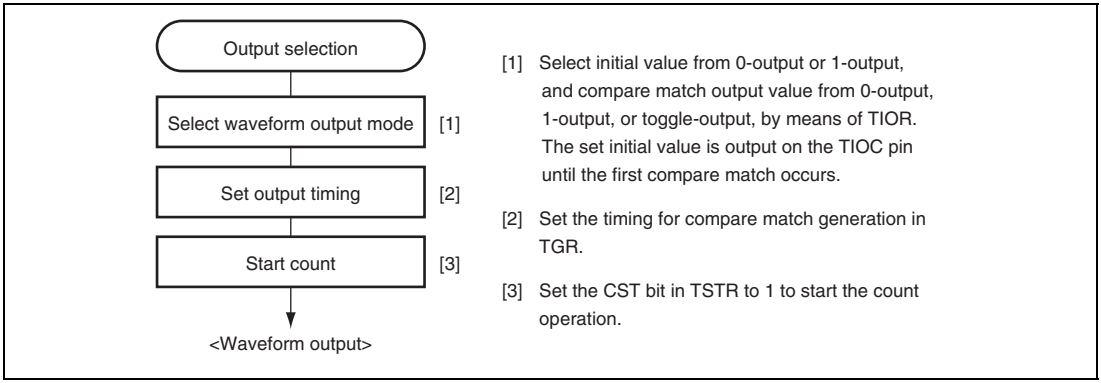


Figure 10.6 Example of Setting Procedure for Waveform Output by Compare Match

(b) Examples of waveform output operation

Figure 10.7 shows an example of 0 output/1 output.

In this example, TCNT has been designated as a free-running counter, and settings have been made so that 1 is output by compare match A, and 0 is output by compare match B. When the set level and the pin level match, the pin level does not change.

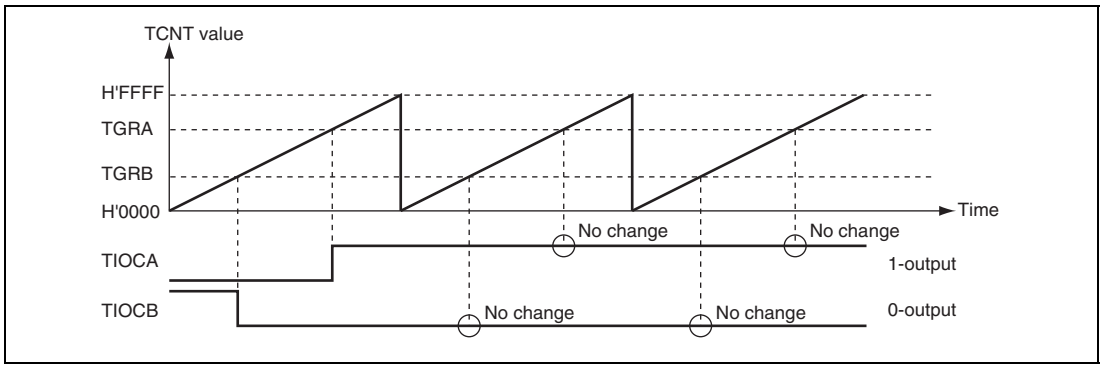


Figure 10.7 Example of 0-Output/1-Output Operation

Figure 10.8 shows an example of toggle output.

In this example, TCNT has been designated as a periodic counter (with counter clearing performed by compare match B), and settings have been made so that output is toggled by both compare match A and compare match B.

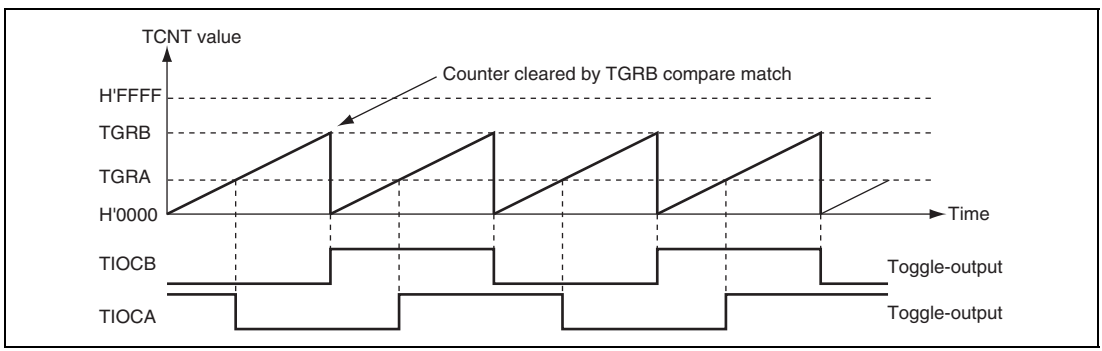


Figure 10.8 Example of Toggle Output Operation

(3) Input Capture Function

The TCNT value can be transferred to TGR on detection of the TIOC pin input edge.

Rising edge, falling edge, or both edges can be selected as the detection edge. For channels 0, 1, 3, and 4, it is also possible to specify another channel's counter input clock or compare match signal as the input capture source.

Note: When another channel's counter input clock is used as the input capture input for channels 0 and 3, Pϕ/1 should not be selected as the counter input clock used for input capture input. Input capture will not be generated if Pϕ/1 is selected.

(a) Example of setting procedure for input capture operation

Figure 10.9 shows an example of the setting procedure for input capture operation.

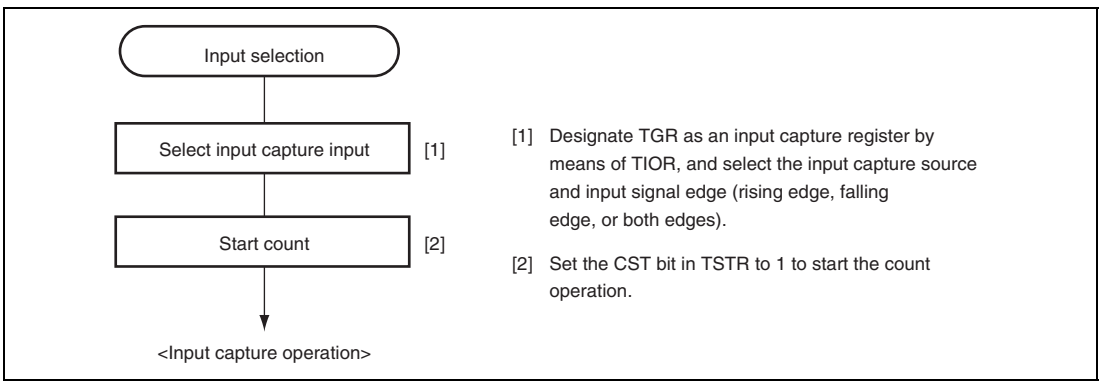


Figure 10.9 Example of Setting Procedure for Input Capture Operation

(b) Example of input capture operation

Figure 10.10 shows an example of input capture operation.

In this example, both rising and falling edges have been selected as the TIOCA pin input capture input edge, falling edge has been selected as the TIOCB pin input capture input edge, and counter clearing by TGRB input capture has been designated for TCNT.

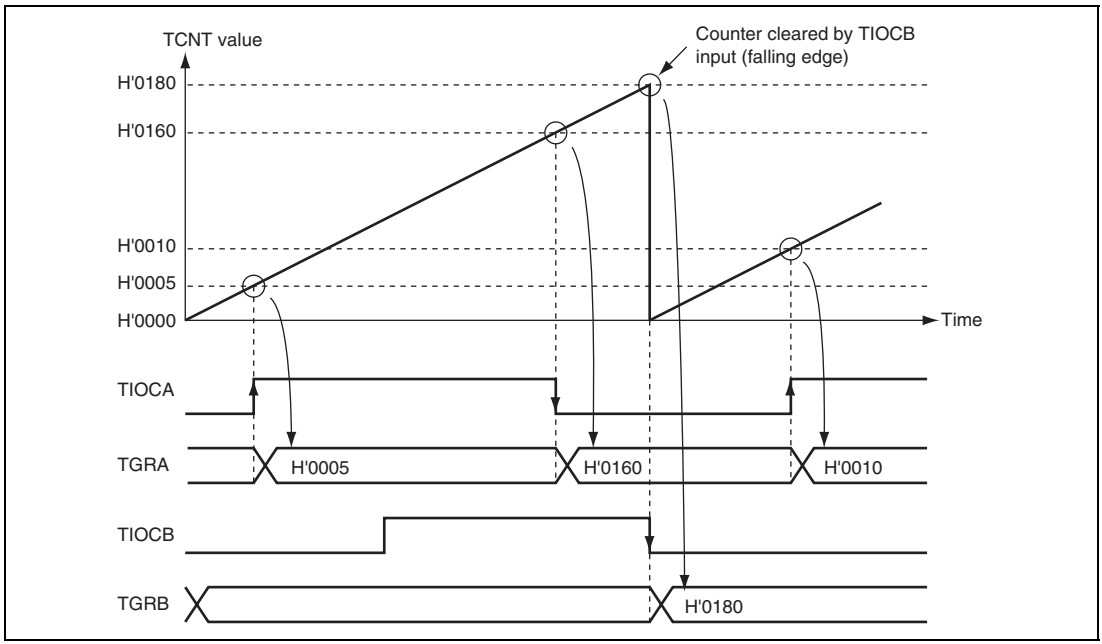


Figure 10.10 Example of Input Capture Operation

10.4.2 Synchronous Operation

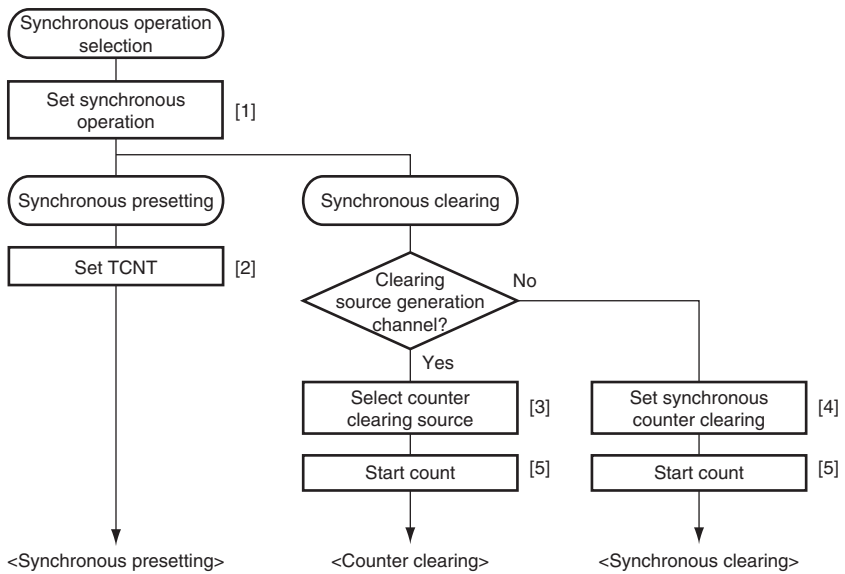
In synchronous operation, the values in multiple TCNT counters can be rewritten simultaneously (synchronous presetting). Also, multiple TCNT counters can be cleared simultaneously (synchronous clearing) by making the appropriate setting in TCR.

Synchronous operation enables TGR to be incremented with respect to a single time base.

Channels 0 to 5 can all be designated for synchronous operation.

(1) Example of Synchronous Operation Setting Procedure

Figure 10.11 shows an example of the synchronous operation setting procedure.



[1] Set the SYNC bits in TSYR corresponding to the channels to be designated for synchronous operation to 1.

[2] When the TCNT counter of any of the channels designated for synchronous operation is written to, the same value is simultaneously written to the other TCNT counters.

[3] Use bits CCLR2 to CCLR0 in TCR to specify TCNT clearing by input capture or output compare.

[4] Use bits CCLR2 to CCLR0 in TCR to designate synchronous clearing for the counter clearing source.

[5] Set the CST bits in TSTR for the relevant channels to 1, to start the count operation.

Figure 10.11 Example of Synchronous Operation Setting Procedure

(2) Example of Synchronous Operation

Figure 10.12 shows an example of synchronous operation.

In this example, synchronous operation and PWM mode 1 have been designated for channels 0 to 2, TGRB_0 compare match has been set as the channel 0 counter clearing source, and synchronous clearing has been set for the channel 1 and 2 counter clearing source.

Three-phase PWM waveforms are output from pins TIOCA0, TIOCA1, and TIOCA2. At this time, synchronous presetting and synchronous clearing by TGRB_0 compare match are performed for channel 0 to 2 TCNT counters, and the data set in TGRB_0 is used as the PWM cycle.

For details on PWM modes, see section 10.4.5, PWM Modes.

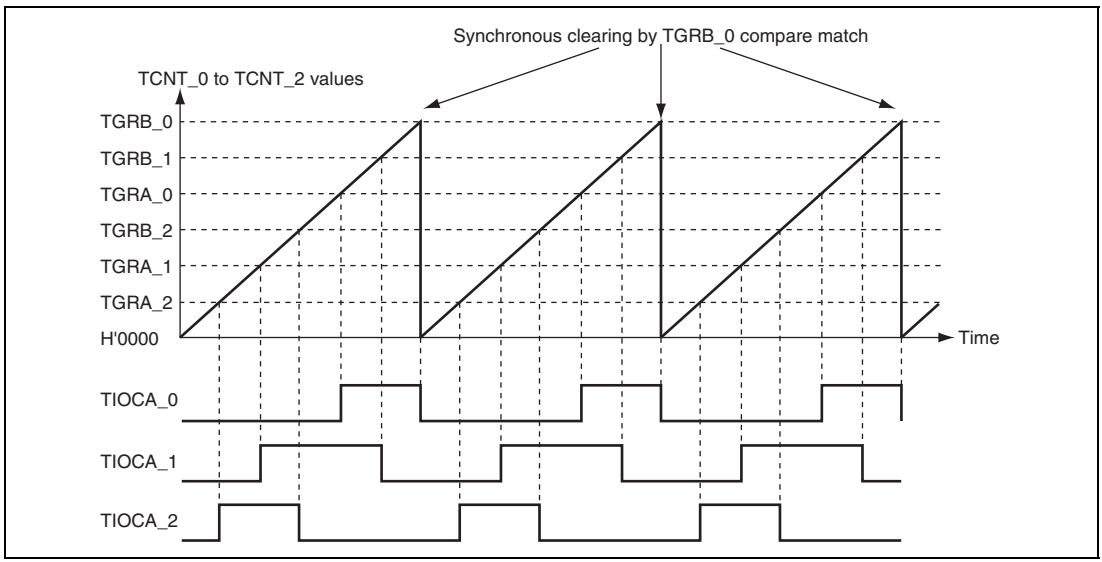


Figure 10.12 Example of Synchronous Operation

10.4.3 Buffer Operation

Buffer operation, provided for channels 0 and 3, enables TGRC and TGRD to be used as buffer registers.

Buffer operation differs depending on whether TGR has been designated as an input capture register or a compare match register.

Table 10.30 shows the register combinations used in buffer operation.

Table 10.30 Register Combinations in Buffer Operation

Channel	Timer General Register	Buffer Register
0	TGRA_0	TGRC_0
	TGRB_0	TGRD_0
3	TGRA_3	TGRC_3
	TGRB_3	TGRD_3

- When TGR is an output compare register

  When a compare match occurs, the value in the buffer register for the corresponding channel is transferred to the timer general register.

  This operation is illustrated in figure 10.13.

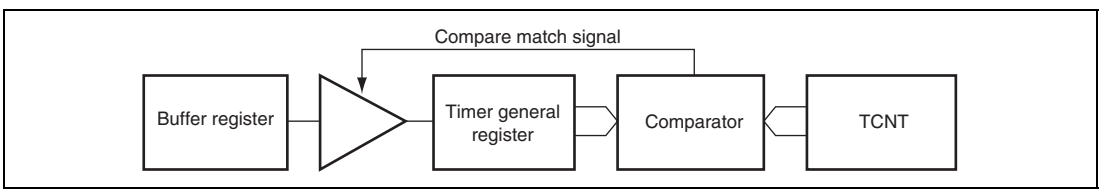


Figure 10.13 Compare Match Buffer Operation

- When TGR is an input capture register

  When input capture occurs, the value in TCNT is transferred to TGR and the value previously held in TGR is transferred to the buffer register.

  This operation is illustrated in figure 10.14.

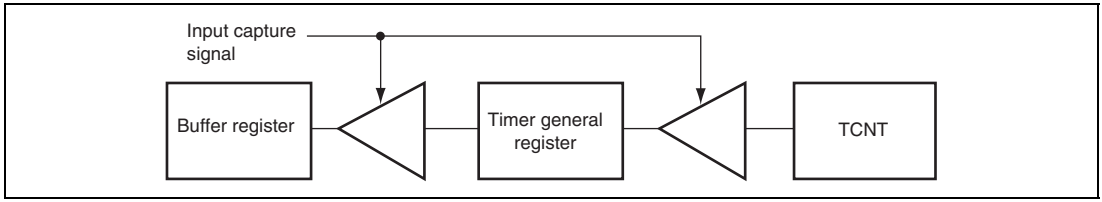


Figure 10.14 Input Capture Buffer Operation

(1) Example of Buffer Operation Setting Procedure

Figure 10.15 shows an example of the buffer operation setting procedure.

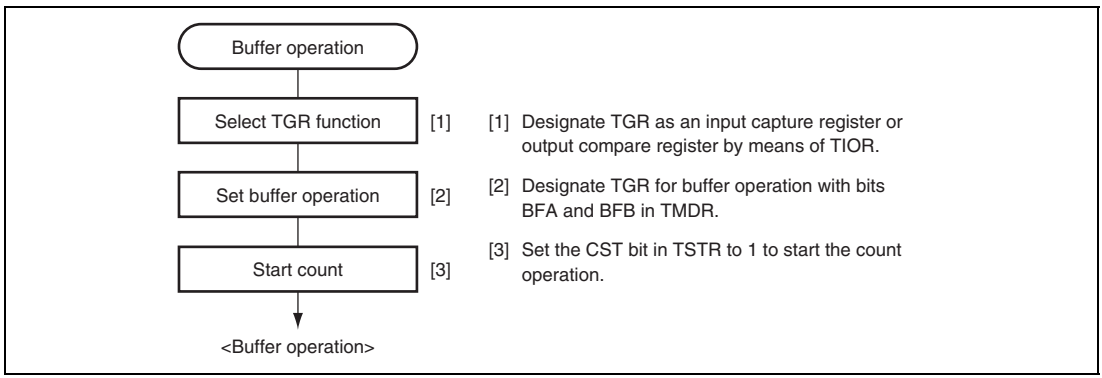


Figure 10.15 Example of Buffer Operation Setting Procedure

(2) Examples of Buffer Operation

(a) When TGR is an output compare register

Figure 10.16 shows an operation example in which PWM mode 1 has been designated for channel 0, and buffer operation has been designated for TGRA and TGRC. The settings used in this example are TCNT clearing by compare match B, 1 output at compare match A, and 0 output at compare match B.

As buffer operation has been set, when compare match A occurs, the output changes and the value in buffer register TGRC is simultaneously transferred to timer general register TGRA. This operation is repeated each time compare match A occurs.

For details on PWM modes, see section 10.4.5, PWM Modes.

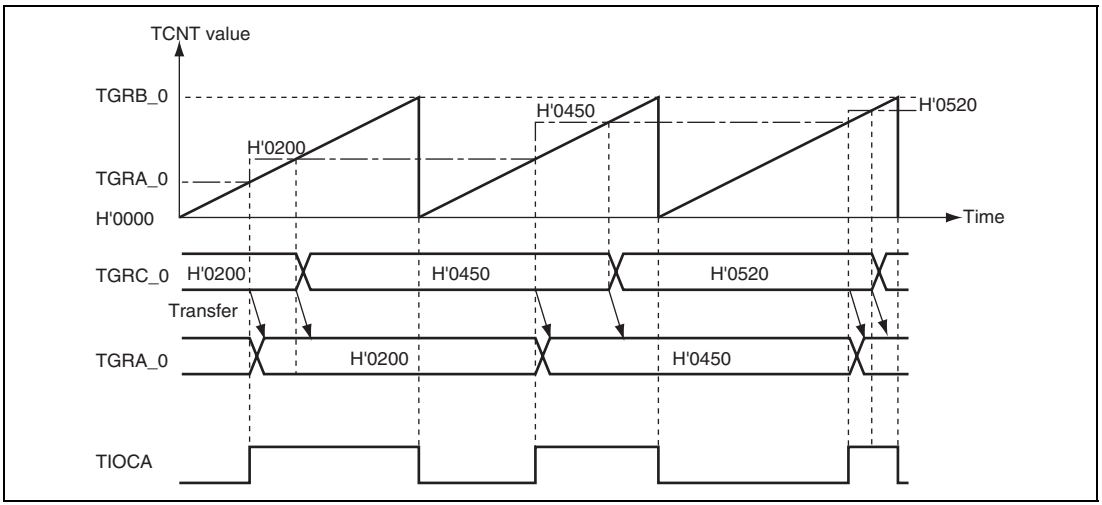


Figure 10.16 Example of Buffer Operation (1)

(b) When TGR is an input capture register

Figure 10.17 shows an operation example in which TGRA has been designated as an input capture register, and buffer operation has been designated for TGRA and TGRC.

Counter clearing by TGRA input capture has been set for TCNT, and both rising and falling edges have been selected as the TIOCA pin input capture input edge.

As buffer operation has been set, when the TCNT value is stored in TGRA upon occurrence of input capture A, the value previously stored in TGRA is simultaneously transferred to TGRC.

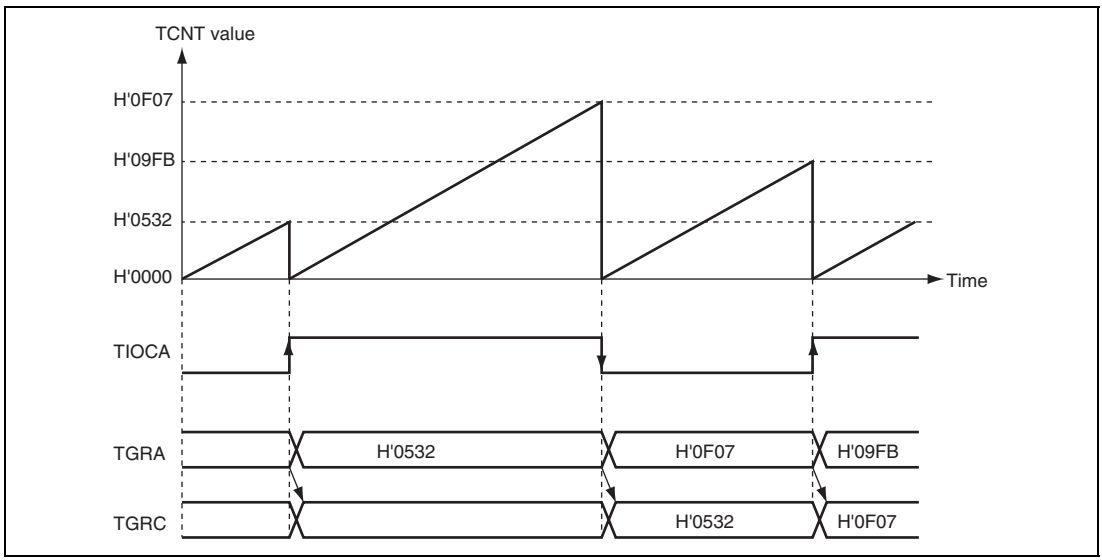


Figure 10.17 Example of Buffer Operation (2)

10.4.4 Cascaded Operation

In cascaded operation, two 16-bit counters for different channels are used together as a 32-bit counter.

This function works by counting the channel 1 (channel 4) counter clock at overflow/underflow of TCNT_2 (TCNT_5) as set in bits TPSC2 to TPSC0 in TCR.

Underflow occurs only when the lower 16-bit TCNT is in phase-counting mode.

Table 10.31 shows the register combinations used in cascaded operation.

Note: When phase counting mode is set for channel 1 or 4, the counter clock setting is invalid and the counter operates independently in phase counting mode.

Table 10.31 Cascaded Combinations

Combination	Upper 16 Bits	Lower 16 Bits
Channels 1 and 2	TCNT_1	TCNT_2
Channels 4 and 5	TCNT_4	TCNT_5

(1) Example of Cascaded Operation Setting Procedure

Figure 10.18 shows an example of the setting procedure for cascaded operation.

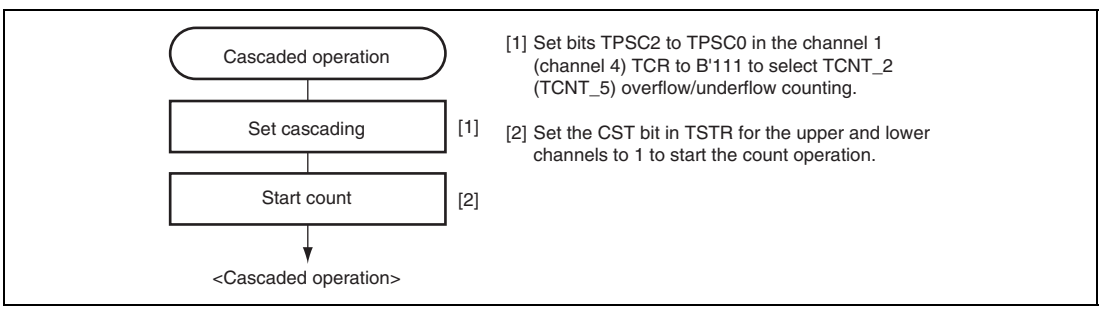


Figure 10.18 Example of Cascaded Operation Setting Procedure

(2) Examples of Cascaded Operation

Figure 10.19 illustrates the operation when counting upon TCNT_2 overflow/underflow has been set for TCNT_1, TGRA_1 and TGRA_2 have been designated as input capture registers, and the TIOC pin rising edge has been selected.

When a rising edge is input to the TIOCA1 and TIOCA2 pins simultaneously, the upper 16 bits of the 32-bit data are transferred to TGRA_1, and the lower 16 bits to TGRA_2.

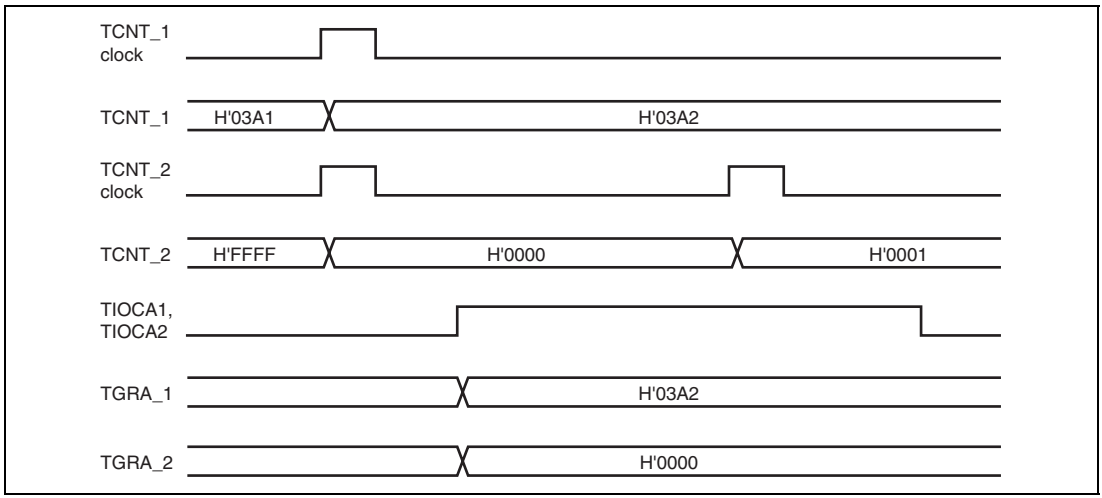


Figure 10.19 Example of Cascaded Operation (1)

Figure 10.20 illustrates the operation when counting upon TCNT_2 overflow/underflow has been set for TCNT_1, and phase counting mode has been designated for channel 2.

TCNT_1 is incremented by TCNT_2 overflow and decremented by TCNT_2 underflow.

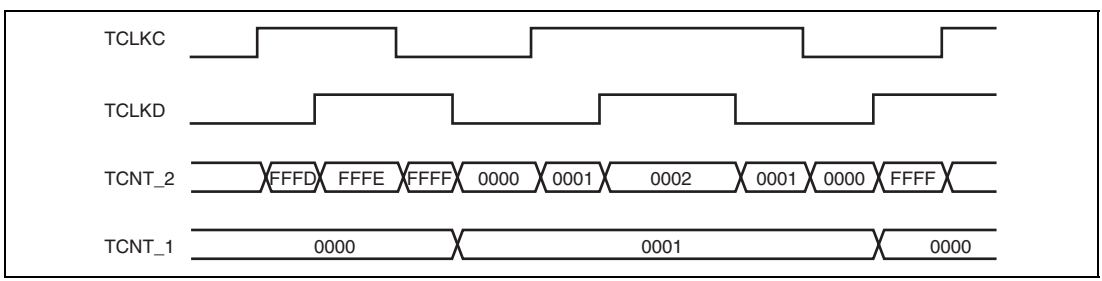


Figure 10.20 Example of Cascaded Operation (2)

10.4.5 PWM Modes

In PWM mode, PWM waveforms are output from the output pins. 0-, 1-, or toggle-output can be selected as the output level in response to compare match of each TGR.

Settings of TGR registers can output a PWM waveform in the range of 0% to 100% duty cycle.

Designating TGR compare match as the counter clearing source enables the cycle to be set in that register. All channels can be designated for PWM mode independently. Synchronous operation is also possible.

There are two PWM modes, as described below.

1. PWM mode 1

   PWM output is generated from the TIOCA and TIOCC pins by pairing TGRA with TGRB and TGRC with TGRD. The outputs specified by bits IOA3 to IOA0 and IOC3 to IOC0 in TIOR are output from the TIOCA and TIOCC pins at compare matches A and C, respectively. The outputs specified by bits IOB3 to IOB0 and IOD3 to IOD0 in TIOR are output at compare matches B and D, respectively. The initial output value is the value set in TGRA or TGRC. If the set values of paired TGRs are identical, the output value does not change when a compare match occurs.

   In PWM mode 1, a maximum 8-phase PWM output is possible.

2. PWM mode 2

   PWM output is generated using one TGR as the cycle register and the others as duty cycle registers. The output specified in TIOR is performed by means of compare matches. Upon counter clearing by a synchronous register compare match, the output value of each pin is the initial value set in TIOR. If the set values of the cycle and duty cycle registers are identical, the output value does not change when a compare match occurs.

   In PWM mode 2, a maximum 15-phase PWM output is possible by combined use with synchronous operation.

   The correspondence between PWM output pins and registers is shown in table 10.32.

Table 10.32 PWM Output Registers and Output Pins

		Output Pins	
Channel	**Registers**	**PWM Mode 1**	**PWM Mode 2**
0	TGRA_0	TIOCA0	TIOCA0
	TGRB_0		TIOCB0
	TGRC_0	TIOCC0	TIOCC0
	TGRD_0		TIOCD0
1	TGRA_1	TIOCA1	TIOCA1
	TGRB_1		TIOCB1
2	TGRA_2	TIOCA2	TIOCA2
	TGRB_2		TIOCB2
3	TGRA_3	TIOCA3	TIOCA3
	TGRB_3		TIOCB3
	TGRC_3	TIOCC3	TIOCC3
	TGRD_3		TIOCD3
4	TGRA_4	TIOCA4	TIOCA4
	TGRB_4		TIOCB4
5	TGRA_5	TIOCA5	TIOCA5
	TGRB_5		TIOCB5

Note: In PWM mode 2, PWM output is not possible for the TGR register in which the cycle is set.

(1) Example of PWM Mode Setting Procedure

Figure 10.21 shows an example of the PWM mode setting procedure.

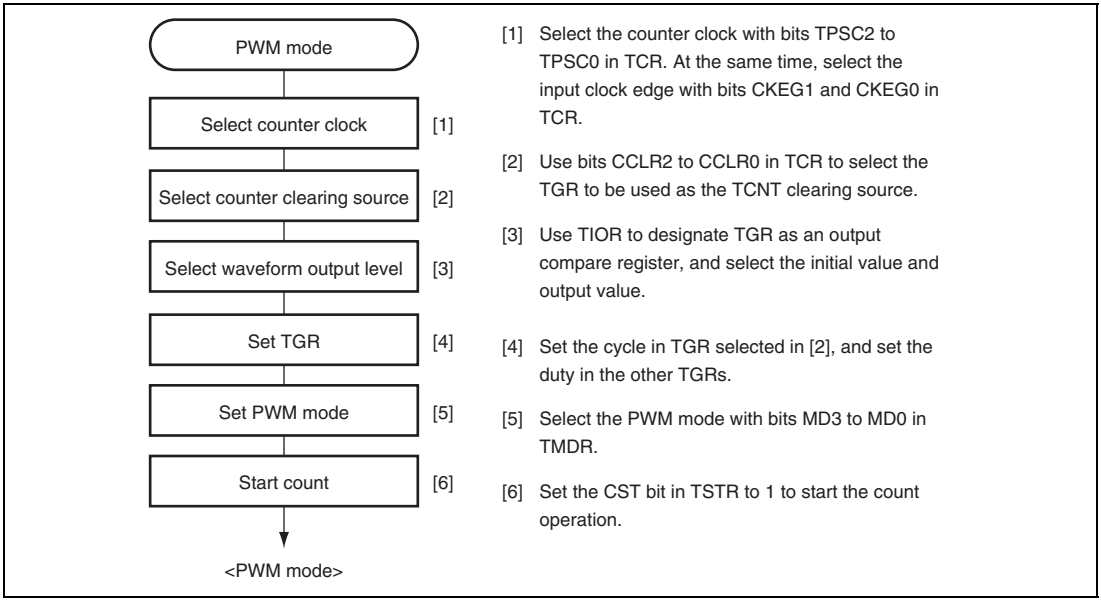


Figure 10.21 Example of PWM Mode Setting Procedure

(2) Examples of PWM Mode Operation

Figure 10.22 shows an example of PWM mode 1 operation.

In this example, TGRA compare match is set as the TCNT clearing source, 0 is set for the TGRA initial output value and output value, and 1 is set as the TGRB output value.

In this case, the value set in TGRA is used as the cycle, and the value set in TGRB register as the duty cycle.

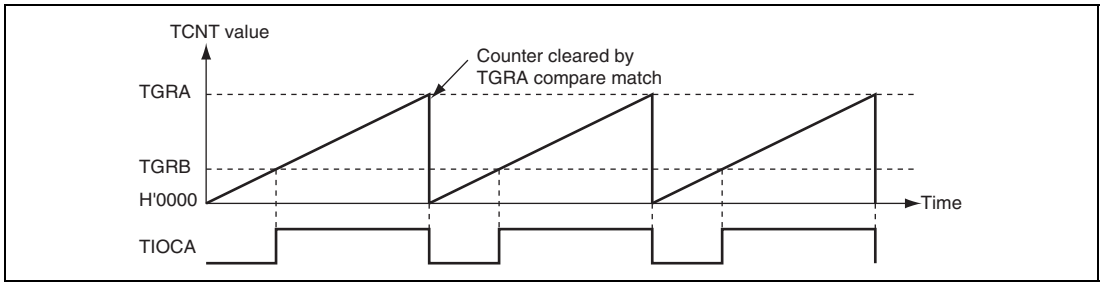


Figure 10.22 Example of PWM Mode Operation (1)

Figure 10.23 shows an example of PWM mode 2 operation.

In this example, synchronous operation is designated for channels 0 and 1, TGRB_1 compare match is set as the TCNT clearing source, and 0 is set for the initial output value and 1 for the output value of the other TGR registers (TGRA_0 to TGRD_0, TGRA_1), to output a 5-phase PWM waveform.

In this case, the value set in TGRB_1 is used as the cycle, and the values set in the other TGRs as the duty cycle.

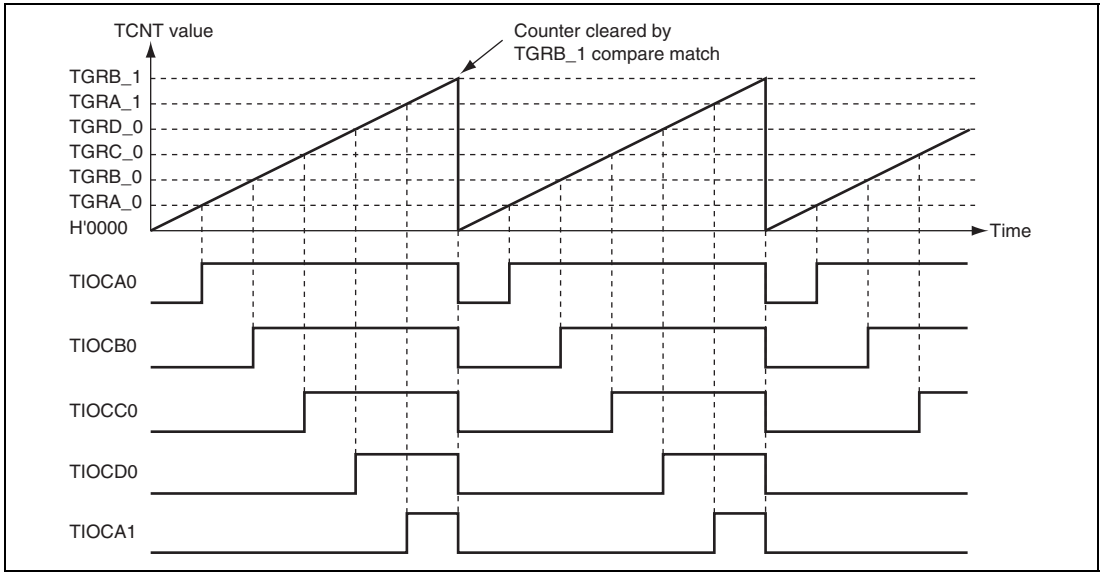


Figure 10.23 Example of PWM Mode Operation (2)

Figure 10.24 shows examples of PWM waveform output with 0% duty cycle and 100% duty cycle in PWM mode.

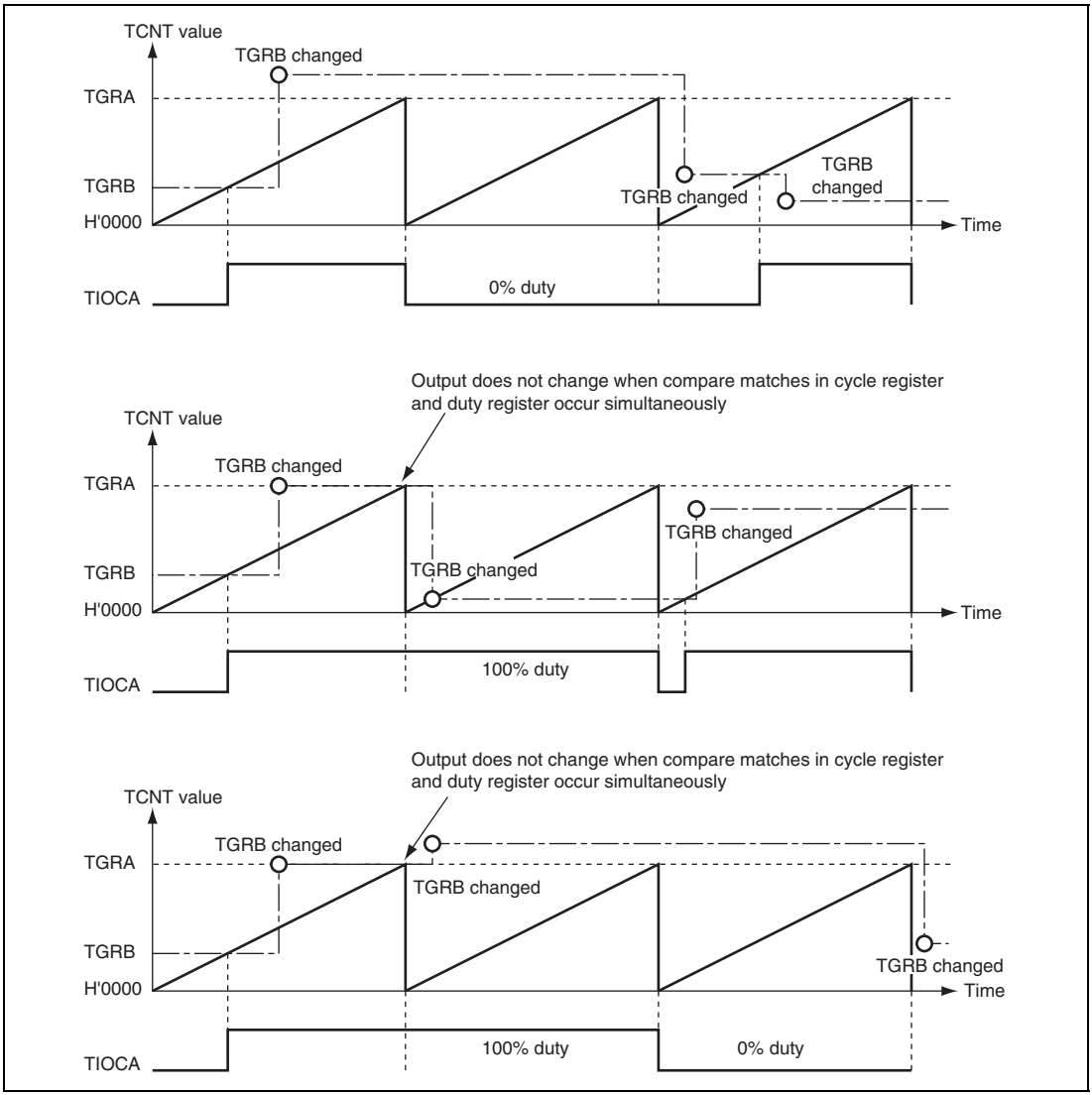


Figure 10.24 Example of PWM Mode Operation (3)

10.4.6 Phase Counting Mode

In phase counting mode, the phase difference between two external clock inputs is detected and TCNT is incremented/decremented accordingly. This mode can be set for channels 1, 2, 4, and 5.

When phase counting mode is set, an external clock is selected as the counter input clock and TCNT operates as an up/down-counter regardless of the setting of bits TPSC2 to TPSC0 and bits CKEG1 and CKEG0 in TCR. However, the functions of bits CCLR1 and CCLR0 in TCR, and of TIOR, TIER, and TGR are valid, and input capture/compare match and interrupt functions can be used.

This can be used for two-phase encoder pulse input.

When overflow occurs while TCNT is counting up, the TCFV flag in TSR is set; when underflow occurs while TCNT is counting down, the TCFU flag is set.

The TCFD bit in TSR is the count direction flag. Reading the TCFD flag provides an indication of whether TCNT is counting up or down.

Table 10.33 shows the correspondence between external clock pins and channels.

Table 10.33 Clock Input Pins in Phase Counting Mode

	External Clock Pins	
Channels	**A-Phase**	**B-Phase**
When channel 1 or 5 is set to phase counting mode	TCLKA	TCLKB
When channel 2 or 4 is set to phase counting mode	TCLKC	TCLKD

(1) Example of Phase Counting Mode Setting Procedure

Figure 10.25 shows an example of the phase counting mode setting procedure.

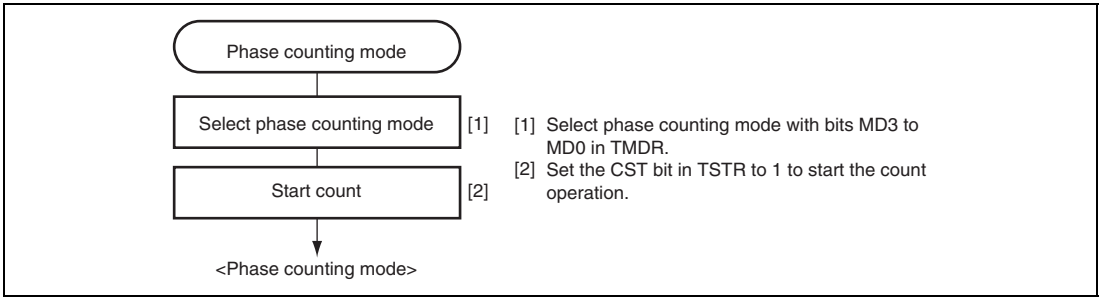


Figure 10.25 Example of Phase Counting Mode Setting Procedure

(2) Examples of Phase Counting Mode Operation

In phase counting mode, TCNT counts up or down according to the phase difference between two external clocks. There are four modes, according to the count conditions.

(a) Phase counting mode 1

Figure 10.26 shows an example of phase counting mode 1 operation, and table 10.34 summarizes the TCNT up/down-count conditions.

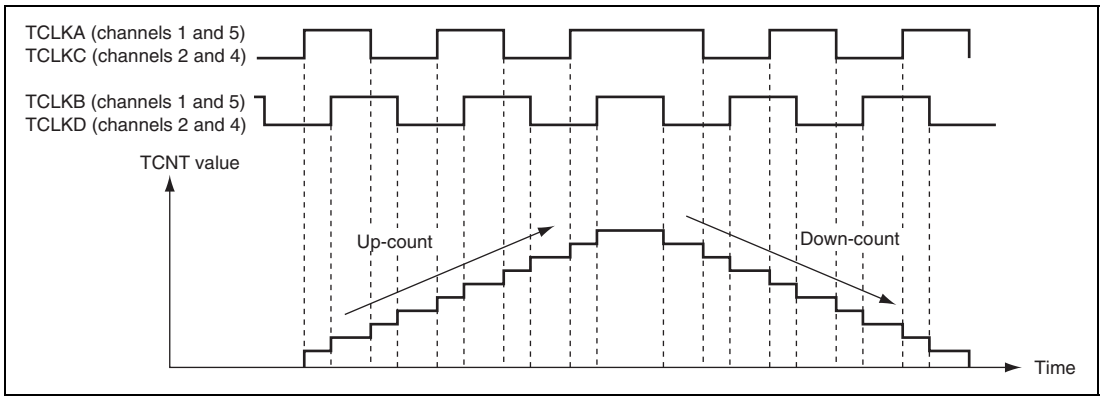


Figure 10.26 Example of Phase Counting Mode 1 Operation

Table 10.34 Up/Down-Count Conditions in Phase Counting Mode 1

TCLKA (Channels 1 and 5) TCLKC (Channels 2 and 4)	TCLKB (Channels 1 and 5) TCLKD (Channels 2 and 4)	Operation
High level	↑	Up-count
Low level	↓	
↑	Low level	
↓	High level	
High level	↓	Down-count
Low level	↑	
↑	High level	
↓	Low level	

[Legend]
↑: Rising edge
↓: Falling edge

(b) Phase counting mode 2

Figure 10.27 shows an example of phase counting mode 2 operation, and table 10.35 summarizes the TCNT up/down-count conditions.

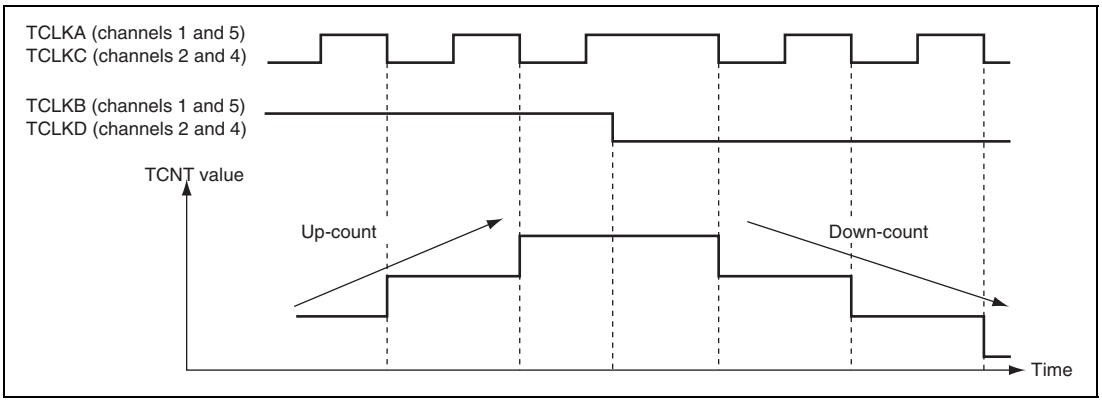


Figure 10.27 Example of Phase Counting Mode 2 Operation

Table 10.35 Up/Down-Count Conditions in Phase Counting Mode 2

TCLKA (Channels 1 and 5) TCLKC (Channels 2 and 4)	TCLKB (Channels 1 and 5) TCLKD (Channels 2 and 4)	Operation
High level	↑	Don't care
Low level	↓	Don't care
↑	Low level	Don't care
↓	High level	Up-count
High level	↓	Don't care
Low level	↑	Don't care
↑	High level	Don't care
↓	Low level	Down-count

[Legend]
↑: Rising edge
↓: Falling edge

(c) Phase counting mode 3

Figure 10.28 shows an example of phase counting mode 3 operation, and table 10.36 summarizes the TCNT up/down-count conditions.

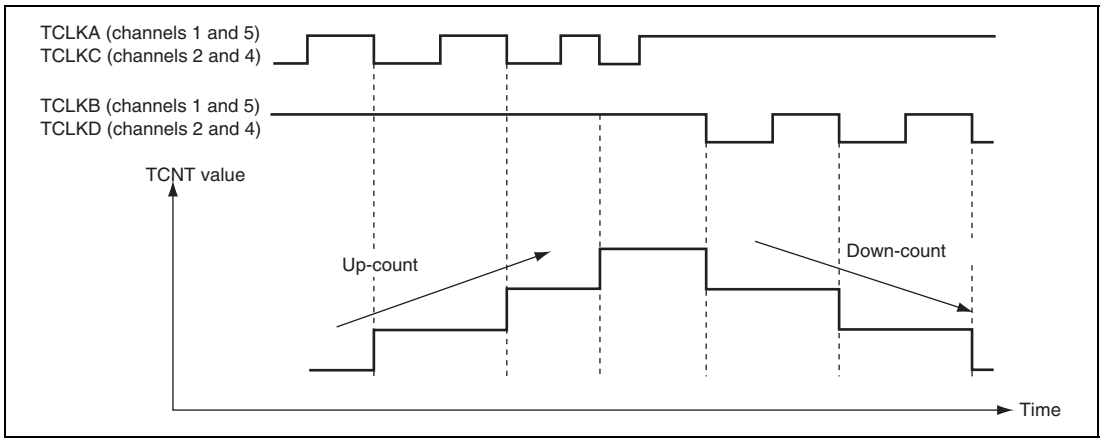


Figure 10.28 Example of Phase Counting Mode 3 Operation

Table 10.36 Up/Down-Count Conditions in Phase Counting Mode 3

TCLKA (Channels 1 and 5) TCLKC (Channels 2 and 4)	TCLKB (Channels 1 and 5) TCLKD (Channels 2 and 4)	Operation
High level	↑	Don't care
Low level	↓	Don't care
↑	Low level	Don't care
↓	High level	Up-count
High level	↓	Down-count
Low level	↑	Don't care
↑	High level	Don't care
↓	Low level	Don't care

[Legend]
↑: Rising edge
↓: Falling edge

(d) Phase counting mode 4

Figure 10.29 shows an example of phase counting mode 4 operation, and table 10.37 summarizes the TCNT up/down-count conditions.

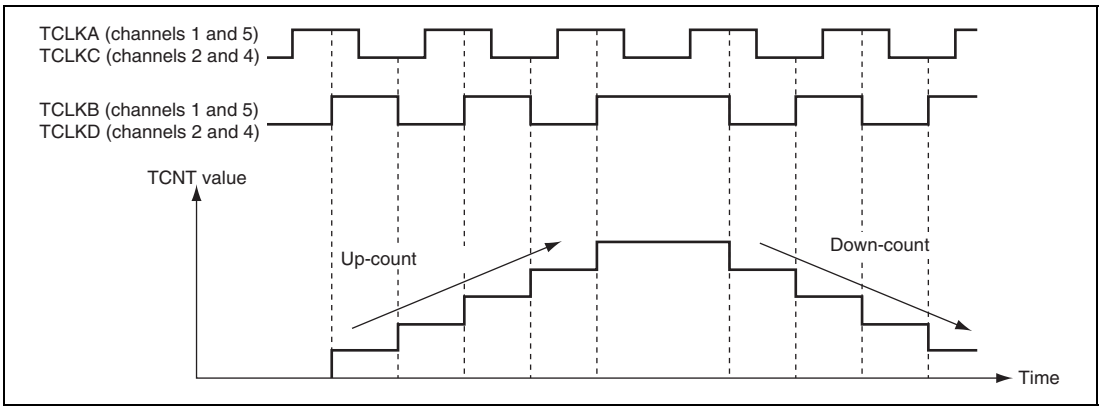


Figure 10.29 Example of Phase Counting Mode 4 Operation

Table 10.37 Up/Down-Count Conditions in Phase Counting Mode 4

TCLKA (Channels 1 and 5) TCLKC (Channels 2 and 4)	TCLKB (Channels 1 and 5) TCLKD (Channels 2 and 4)	Operation
High level	↑	Up-count
Low level	↓	
↑	Low level	Don't care
↓	High level	
High level	↓	Down-count
Low level	↑	
↑	High level	Don't care
↓	Low level	

[Legend]
↑: Rising edge
↓: Falling edge

(3) Phase Counting Mode Application Example

Figure 10.30 shows an example in which phase counting mode is designated for channel 1, and channel 1 is linked with channel 0 to input servo motor 2-phase encoder pulses in order to detect the position or speed.

Channel 1 is set to phase counting mode 1, and the encoder pulse A-phase and B-phase are input to TCLKA and TCLKB.

Channel 0 operates with TCNT counter clearing by TGRC_0 compare match; TGRA_0 and TGRC_0 are used for the compare match function and are set with the speed control cycle and position control cycle. TGRB_0 is used for input capture, with TGRB_0 and TGRD_0 operating in buffer mode. The channel 1 counter input clock is designated as the TGRB_0 input capture source, and the pulse width of 2-phase encoder 4-multiplication pulses is detected.

TGRA_1 and TGRB_1 for channel 1 are designated for input capture, channel 0 TGRA_0 and TGRC_0 compare matches are selected as the input capture source, and the up/down-counter values for the control cycles are stored.

This procedure enables accurate position/speed detection to be achieved.

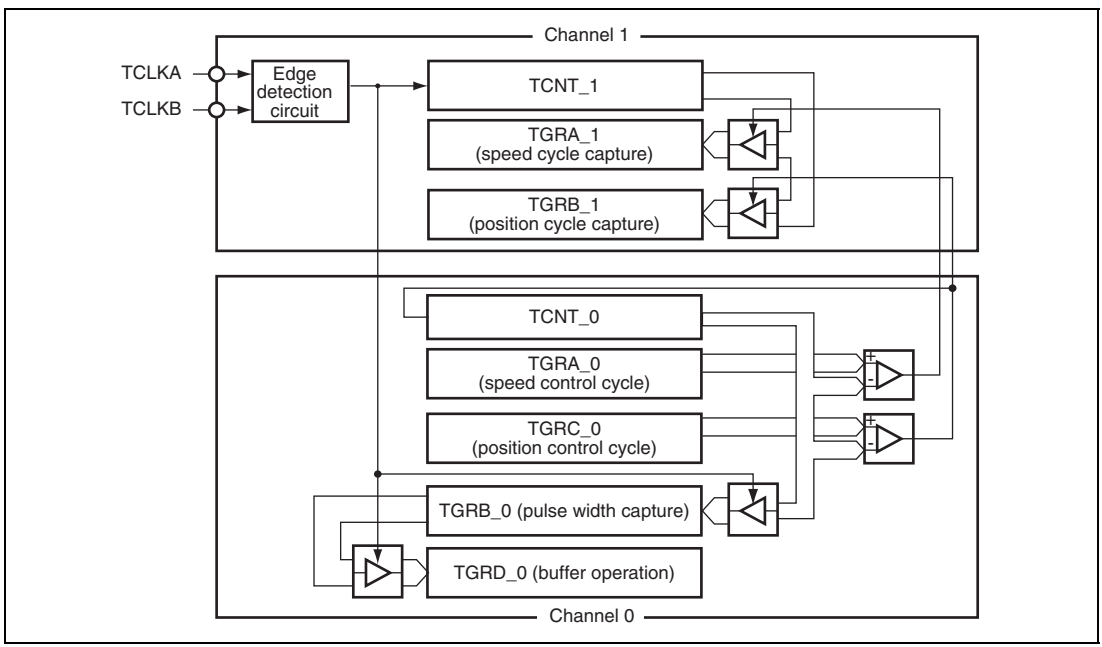


Figure 10.30 Phase Counting Mode Application Example

10.5 Interrupt Sources

There are three kinds of TPU interrupt sources: TGR input capture/compare match, TCNT overflow, and TCNT underflow. Each interrupt source has its own status flag and enable/disable bit, allowing generation of interrupt request signals to be enabled or disabled individually.

When an interrupt request is generated, the corresponding status flag in TSR is set to 1. If the corresponding enable/disable bit in TIER is set to 1 at this time, an interrupt is requested. The interrupt request is cleared by clearing the status flag to 0.

Table 10.38 lists the TPU interrupt sources.

Table 10.38 TPU Interrupts

Channel	Name	Interrupt Source	Interrupt Flag	DTC Activation (in_vnum)	
0	TGI0A	TGRA_0 input capture/compare match	TGFA_0	Possible	(00000)
	TGI0B	TGRB_0 input capture/compare match	TGFB_0	Possible	(00001)
	TGI0C	TGRC_0 input capture/compare match	TGFC_0	Possible	(00010)
	TGI0D	TGRD_0 input capture/compare match	TGFD_0	Possible	(00011)
	TCI0V	TCNT_0 overflow	TCFV_0	Not possible	
1	TGI1A	TGRA_1 input capture/compare match	TGFA_1	Possible	(00100)
	TGI1B	TGRB_1 input capture/compare match	TGFB_1	Possible	(00101)
	TCI1V	TCNT_1 overflow	TCFV_1	Not possible	
	TCI1U	TCNT_1 underflow	TCFU_1	Not possible	
2	TGI2A	TGRA_2 input capture/compare match	TGFA_2	Possible	(00110)
	TGI2B	TGRB_2 input capture/compare match	TGFB_2	Possible	(00111)
	TCI2V	TCNT_2 overflow	TCFV_2	Not possible	
	TCI2U	TCNT_2 underflow	TCFU_2	Not possible	
3	TGI3A	TGRA_3 input capture/compare match	TGFA_3	Possible	(01000)
	TGI3B	TGRB_3 input capture/compare match	TGFB_3	Possible	(01001)
	TGI3C	TGRC_3 input capture/compare match	TGFC_3	Possible	(01010)
	TGI3D	TGRD_3 input capture/compare match	TGFD_3	Possible	(01011)
	TCI3V	TCNT_3 overflow	TCFV_3	Not possible	

Channel	Name	Interrupt Source	Interrupt Flag	DTC Activation	
4	TGI4A	TGRA_4 input capture/compare match	TGFA_4	Possible	(01100)
	TGI4B	TGRB_4 input capture/compare match	TGFB_4	Possible	(01101)
	TCI4V	TCNT_4 overflow	TCFV_4	Not possible	
	TCI4U	TCNT_4 underflow	TCFU_4	Not possible	
5	TGI5A	TGRA_5 input capture/compare match	TGFA_5	Possible	(10000)
	TGI5B	TGRB_5 input capture/compare match	TGFB_5	Possible	(10001)
	TCI5V	TCNT_5 overflow	TCFV_5	Not possible	
	TCI5U	TCNT_5 underflow	TCFU_5	Not possible	

Notes: 1. This table shows the initial state immediately after a reset. The relative channel priority levels can be changed by the interrupt controller.
2. In this LSI, the input capture function is not available on channels 3, 4, and 5.

(1) Input Capture/Compare Match Interrupt

An interrupt is requested if the TGIE bit in TIER is set to 1 when the TGF flag in TSR is set to 1 by the occurrence of a TGR input capture/compare match on a channel. The interrupt request is cleared by clearing the TGF flag to 0. The TPU has 16 input capture/compare match interrupts, four each for channels 0 and 3, and two each for channels 1, 2, 4, and 5.

(2) Overflow Interrupt

An interrupt is requested if the TCIEV bit in TIER is set to 1 when the TCFV flag in TSR is set to 1 by the occurrence of a TCNT overflow on a channel. The interrupt request is cleared by clearing the TCFV flag to 0. The TPU has six overflow interrupts, one for each channel.

(3) Underflow Interrupt

An interrupt is requested if the TCIEU bit in TIER is set to 1 when the TCFU flag in TSR is set to 1 by the occurrence of a TCNT underflow on a channel. The interrupt request is cleared by clearing the TCFU flag to 0. The TPU has four underflow interrupts, one each for channels 1, 2, 4, and 5.

10.6 DTC Activation

The DTC can be activated by the TGR input capture/compare match interrupt for a channel. For details, see section 8, Data Transfer Controller (DTC).

A total of 16 TPU input capture/compare match interrupts can be used as DTC activation sources, four each for channels 0 and 3, and two each for channels 1, 2, 4, and 5.

10.7 A/D Converter Activation

The TGRA input capture/compare match for each channel can activate the A/D converter.

If the TTGE bit in TIER is set to 1 when the TGFA flag in TSR is set to 1 by the occurrence of a TGRA input capture/compare match on a particular channel, a request to start A/D conversion is sent to the A/D converter. If the TPU conversion start trigger has been selected on the A/D converter side at this time, A/D conversion is started.

In the TPU, a total of six TGRA input capture/compare match interrupts can be used as A/D converter conversion start sources, one for each channel.

10.8 Operation Timing

10.8.1 Input/Output Timing

(1) TCNT Count Timing

Figure 10.31 shows TCNT count timing in internal clock operation, and figure 10.32 shows TCNT count timing in external clock operation.

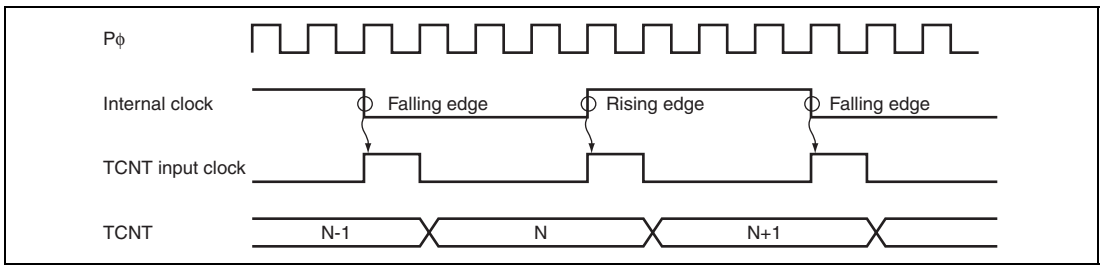


Figure 10.31 Count Timing in Internal Clock Operation

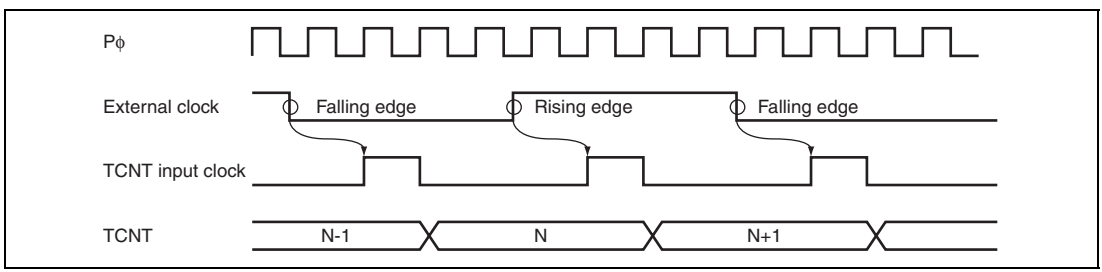


Figure 10.32 Count Timing in External Clock Operation

(2) Output Compare Output Timing

A compare match signal is generated in the final state in which TCNT and TGR match (the point at which the count value matched by TCNT is updated). When a compare match signal is generated, the output value set in TIOR is output at the output compare output pin (TIOC pin). After a match between TCNT and TGR, the compare match signal is not generated until the TCNT input clock is generated.

Figure 10.33 shows output compare output timing.

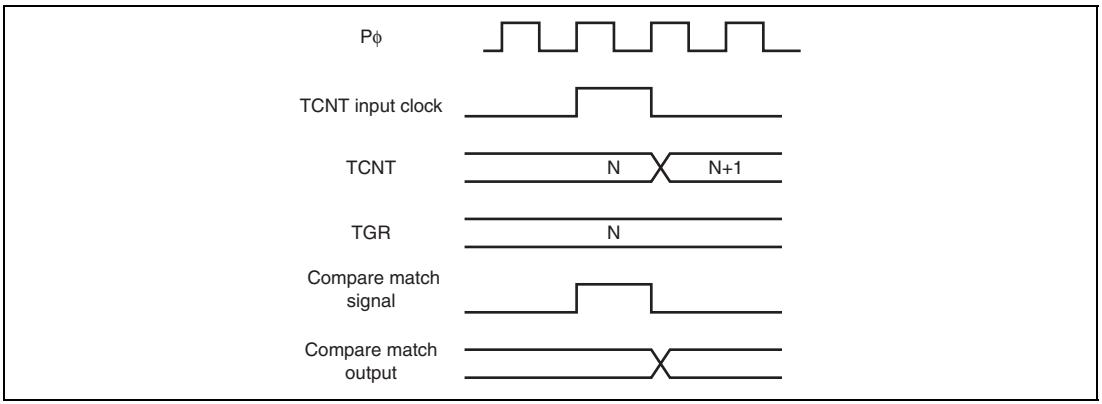


Figure 10.33 Output Compare Output Timing

(3) Input Capture Signal Timing

Figure 10.34 shows input capture signal timing.

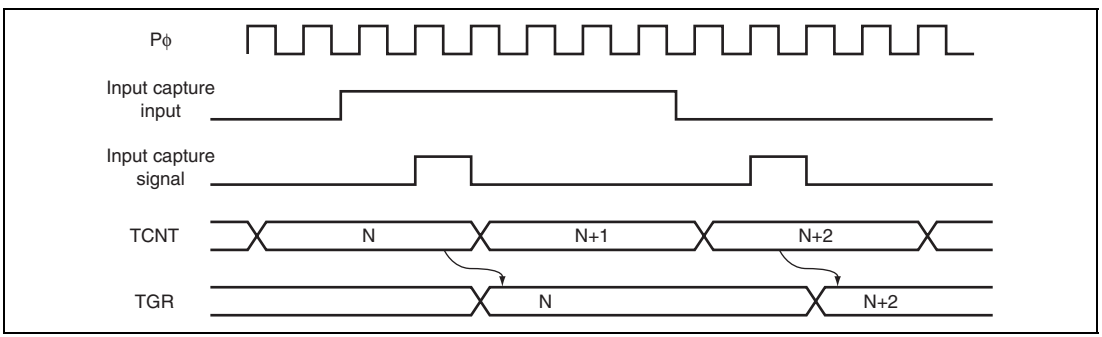


Figure 10.34 Input Capture Input Signal Timing

(4) Timing for Counter Clearing by Compare Match/Input Capture

Figure 10.35 shows the timing when counter clearing by compare match occurrence is specified, and figure 10.36 shows the timing when counter clearing by input capture occurrence is specified.

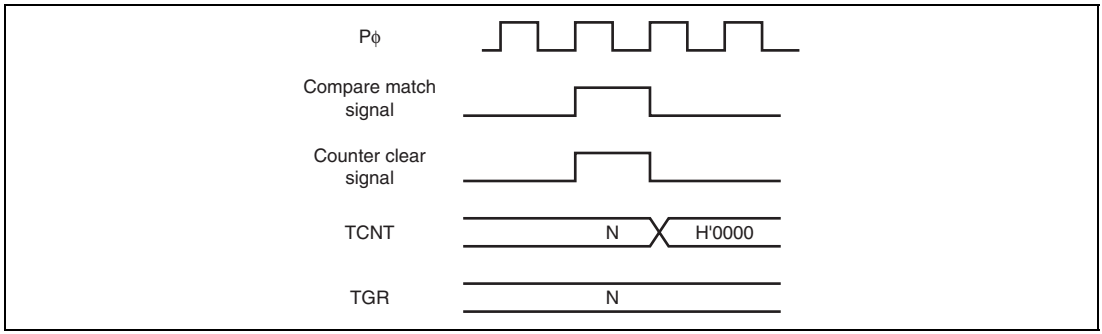


Figure 10.35 Counter Clear Timing (Compare Match)

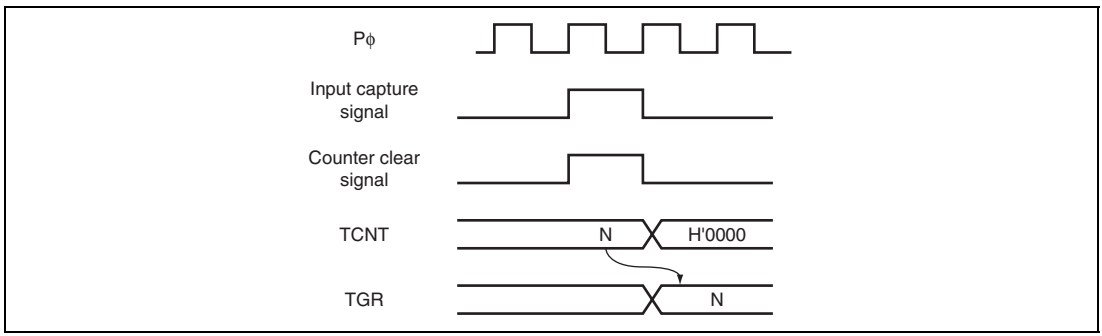


Figure 10.36 Counter Clear Timing (Input Capture)

(5) Buffer Operation Timing

Figures 10.37 and 10.38 show the timings in buffer operation.

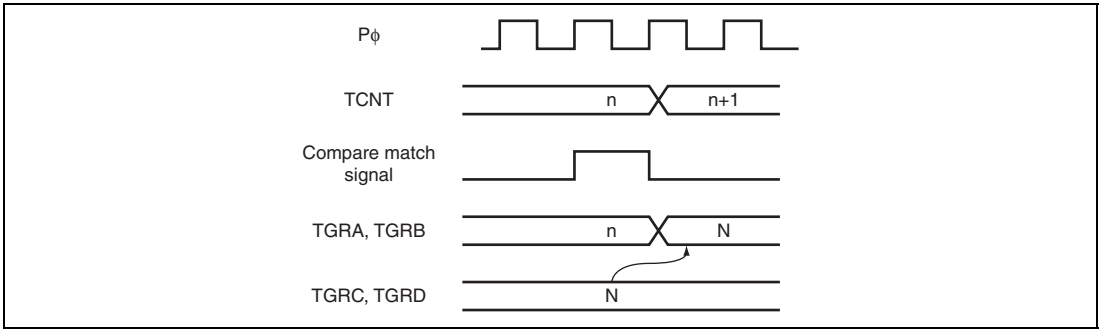


Figure 10.37 Buffer Operation Timing (Compare Match)

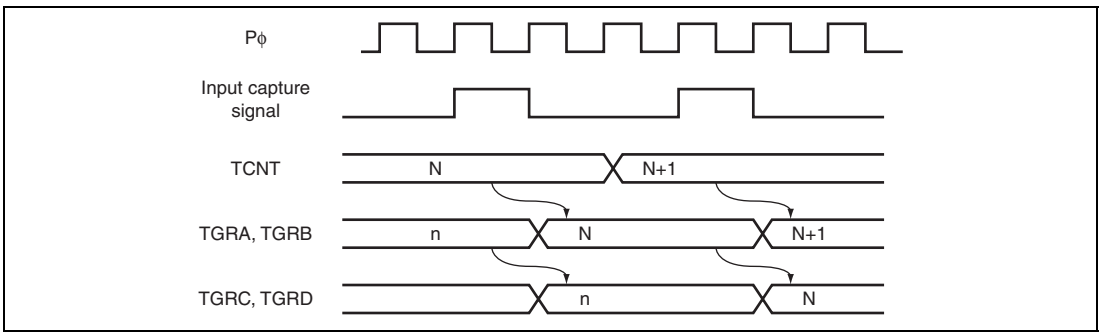


Figure 10.38 Buffer Operation Timing (Input Capture)

10.8.2 Interrupt Signal Timing

(1) TGF Flag Setting Timing in Case of Compare Match

Figure 10.39 shows the timing for setting of the TGF flag in TSR by compare match occurrence, and the TGI interrupt request signal timing.

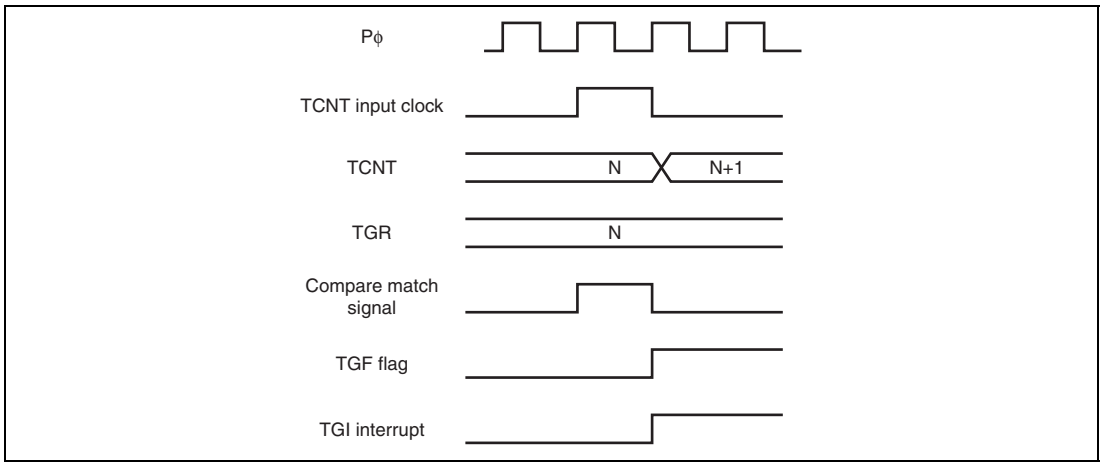


Figure 10.39 TGI Interrupt Timing (Compare Match)

(2) TGF Flag Setting Timing in Case of Input Capture

Figure 10.40 shows the timing for setting of the TGF flag in TSR by input capture occurrence, and the TGI interrupt request signal timing.

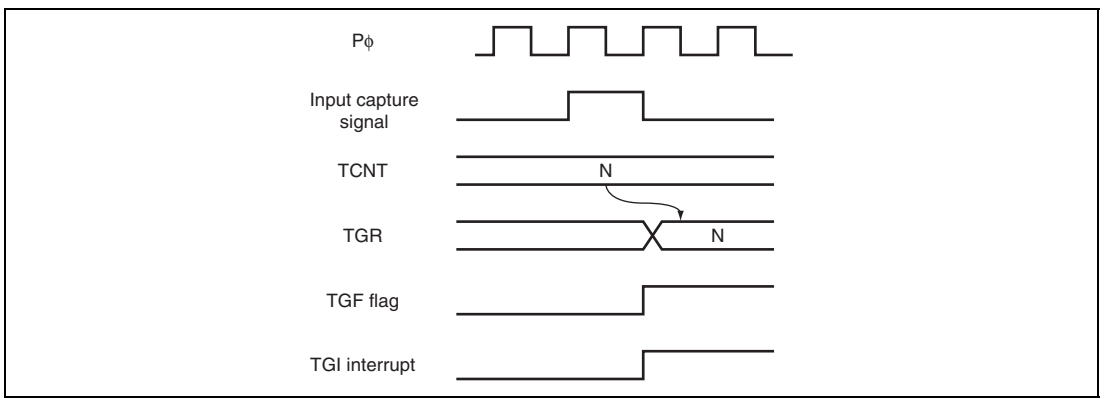


Figure 10.40 TGI Interrupt Timing (Input Capture)

(3) TCFV Flag/TCFU Flag Setting Timing

Figure 10.41 shows the timing for setting of the TCFV flag in TSR by overflow occurrence, and the TCIV interrupt request signal timing.

Figure 10.42 shows the timing for setting of the TCFU flag in TSR by underflow occurrence, and the TCIU interrupt request signal timing.

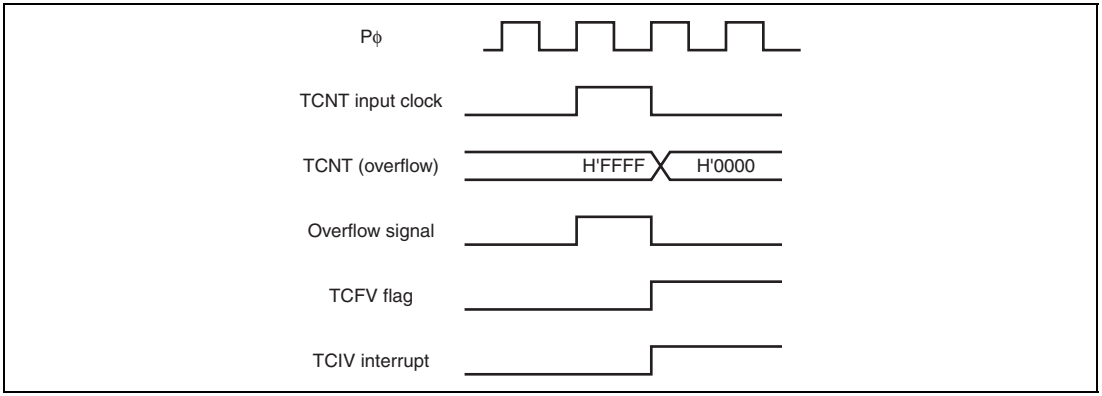


Figure 10.41 TCIV Interrupt Setting Timing

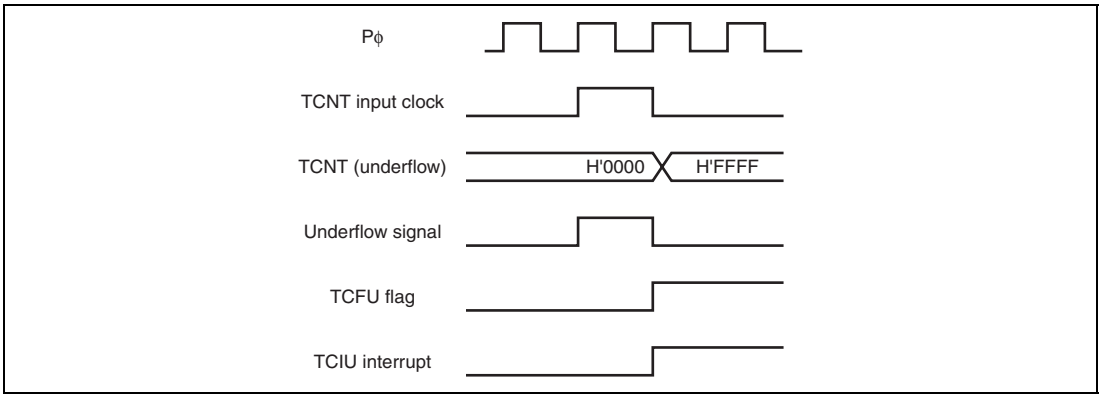


Figure 10.42 TCIU Interrupt Setting Timing

(4) Status Flag Clearing Timing

After a status flag is read as 1 by the CPU, it is cleared by writing 0 to it. When the DTC is activated, the flag is cleared automatically. Figure 10.43 shows the timing for status flag clearing by the CPU, and figures 10.44 shows the timing for status flag clearing by the DTC.

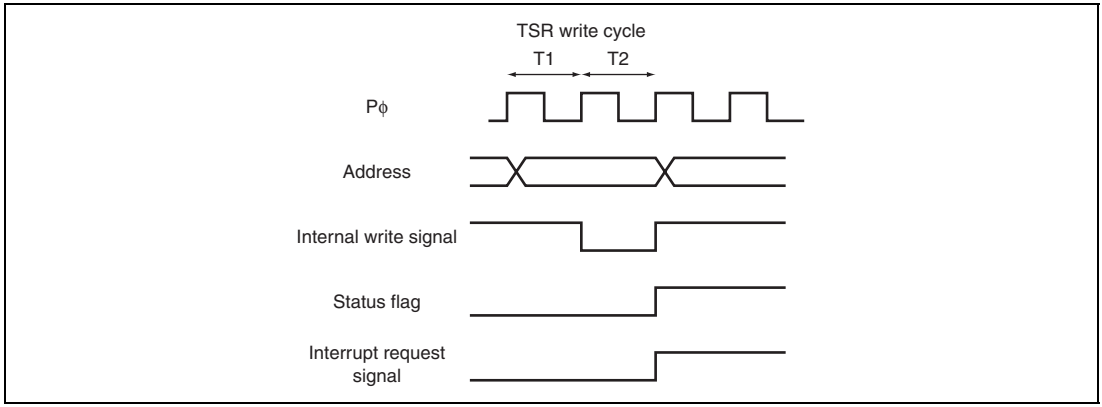


Figure 10.43 Timing for Status Flag Clearing by CPU

The status flag and interrupt request signal are cleared during the DTC transfer as shown in figure 10.44.

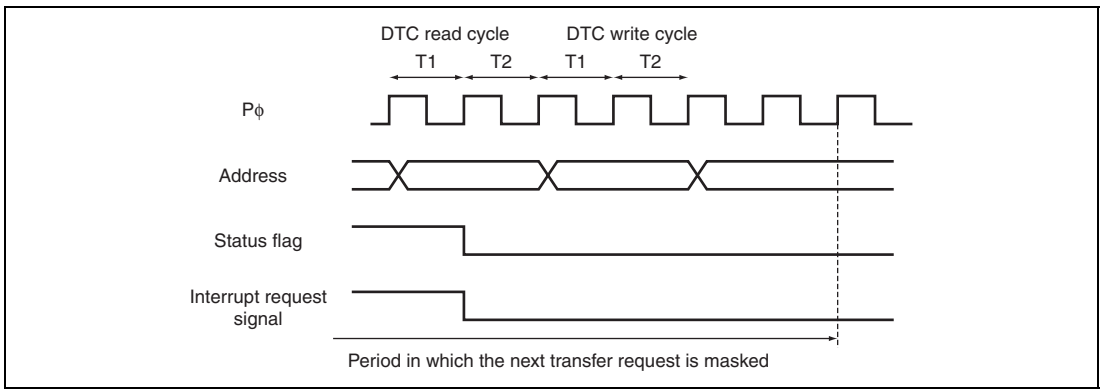


Figure 10.44 Timing for Status Flag Clearing by DTC Activation

10.9 Usage Notes

10.9.1 Module Stop Mode Setting

Operation of the TPU can be enabled or disabled using the module stop control register. The initial setting is for operation of the TPU to be halted. Register access is enabled by clearing module stop mode. For details, see section 23, Power-Down Modes.

10.9.2 Input Clock Restrictions

The input clock pulse width must be at least 1.5 states in the case of single-edge detection, and at least 2.5 states in the case of both-edge detection. The TPU will not operate properly with a narrower pulse width.

In phase counting mode, the phase difference and overlap between the two input clocks must be at least 1.5 states, and the pulse width must be at least 2.5 states. Figure 10.45 shows the input clock conditions in phase counting mode.

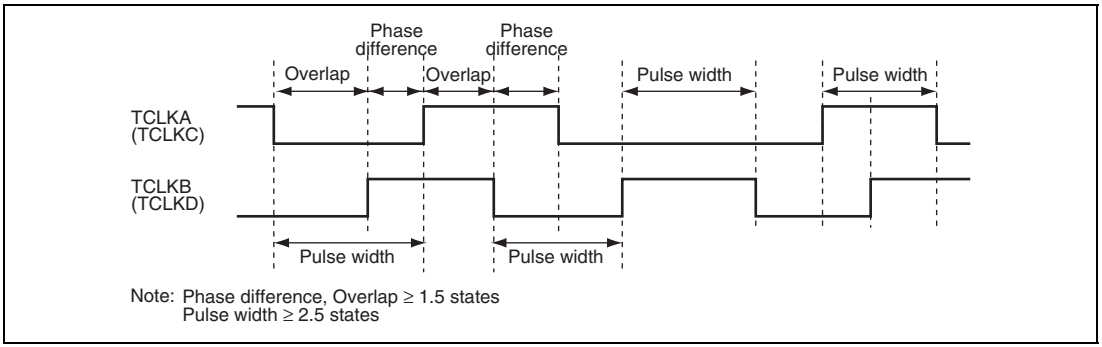


Figure 10.45 Phase Difference, Overlap, and Pulse Width in Phase Counting Mode

10.9.3 Caution on Cycle Setting

When counter clearing by compare match is set, TCNT is cleared in the final state in which it matches the TGR value (the point at which the count value matched by TCNT is updated). Consequently, the actual counter frequency is given by the following formula:

$$f = \frac{P\phi}{(N + 1)}$$

f: Counter frequency
Pϕ: Operating frequency
N: TGR set value

10.9.4 Conflict between TCNT Write and Clear Operations

If the counter clearing signal is generated in the T2 state of a TCNT write cycle, TCNT clearing takes precedence and the TCNT write is not performed. Figure 10.46 shows the timing in this case.

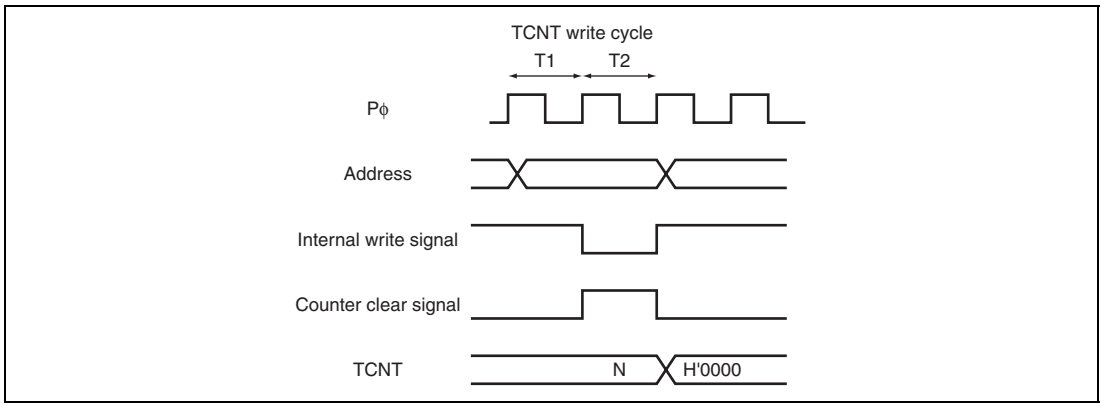


Figure 10.46 Conflict between TCNT Write and Clear Operations

10.9.5 Conflict between TCNT Write and Increment Operations

If incrementing occurs in the T2 state of a TCNT write cycle, the TCNT write takes precedence and TCNT is not incremented. Figure 10.47 shows the timing in this case.

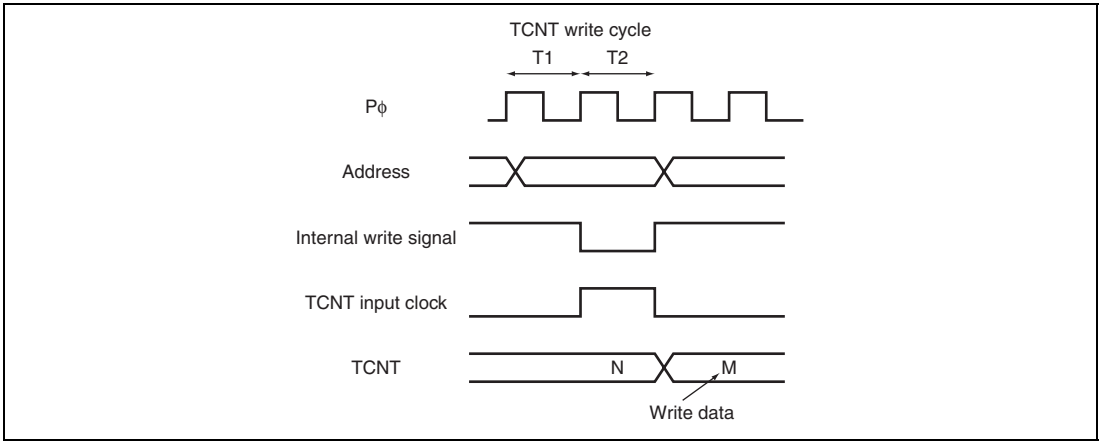


Figure 10.47 Conflict between TCNT Write and Increment Operations

10.9.6 Conflict between TGR Write and Compare Match

If a compare match occurs in the T2 state of a TGR write cycle, the TGR write takes precedence and the compare match signal is disabled. A compare match also does not occur when the same value as before is written.

Figure 10.48 shows the timing in this case.

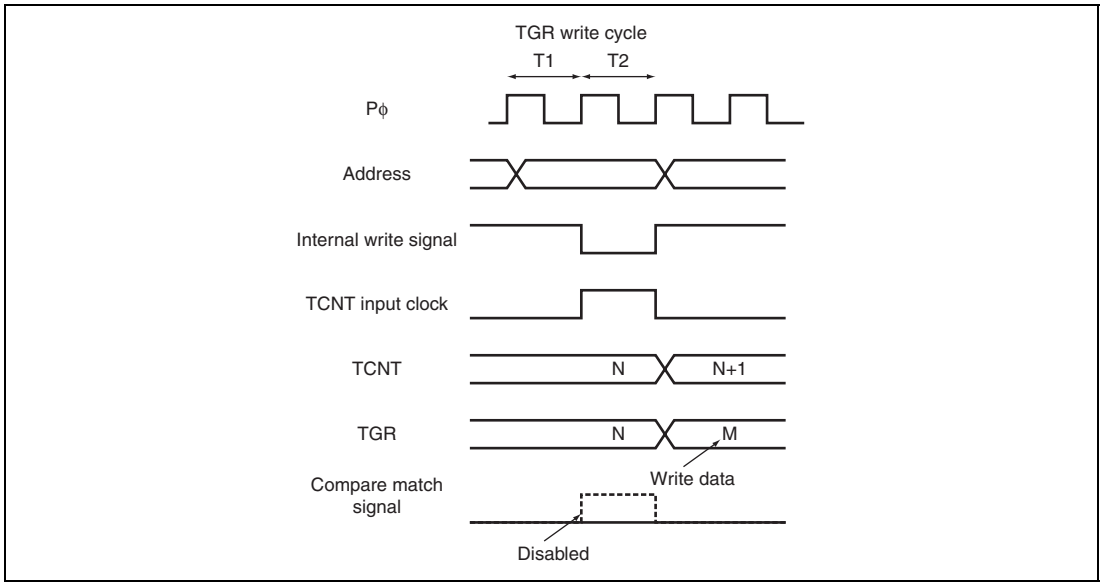


Figure 10.48 Conflict between TGR Write and Compare Match

10.9.7 Conflict between Buffer Register Write and Compare Match

If a compare match occurs in the T2 state of a TGR write cycle, the data transferred to TGR by the buffer operation will not be the write data (M).

Figure 10.49 shows the timing in this case.

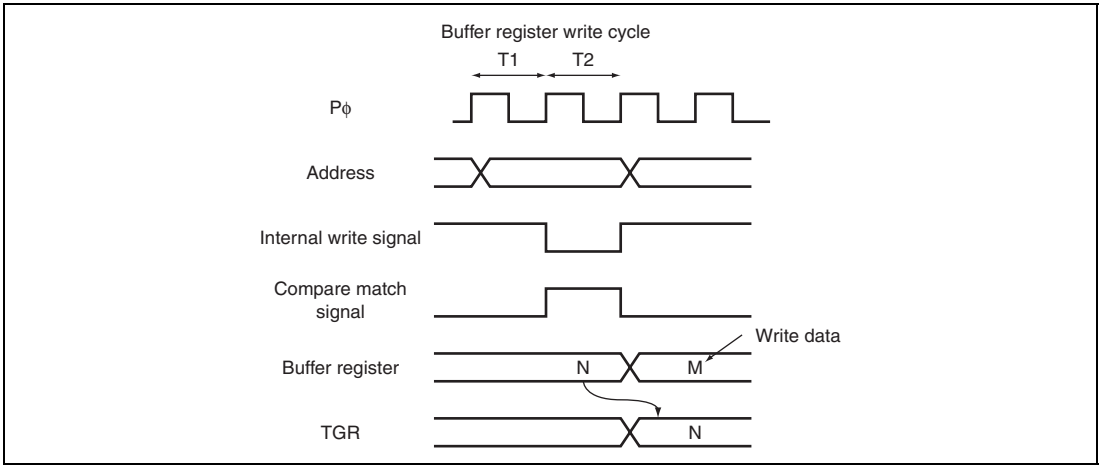


Figure 10.49 Conflict between Buffer Register Write and Compare Match

10.9.8 Conflict between TGR Read and Input Capture

If the input capture signal is generated in the T1 state of a TGR read cycle, the data that is read will be the data before input capture transfer (M).

Figure 10.50 shows the timing in this case.

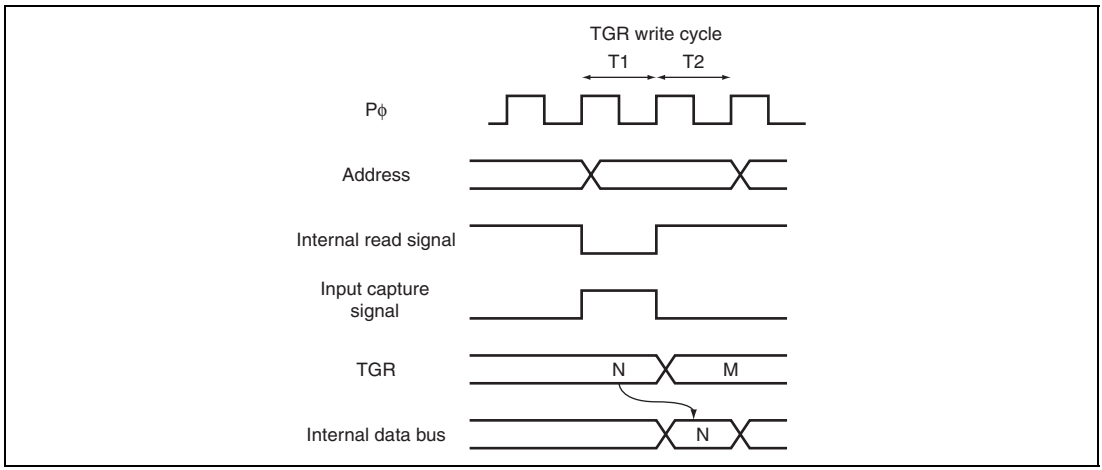


Figure 10.50 Conflict between TGR Read and Input Capture

10.9.9 Conflict between TGR Write and Input Capture

If the input capture signal is generated in the T2 state of a TGR write cycle, the input capture operation takes precedence and the write to TGR is not performed.

Figure 10.51 shows the timing in this case.

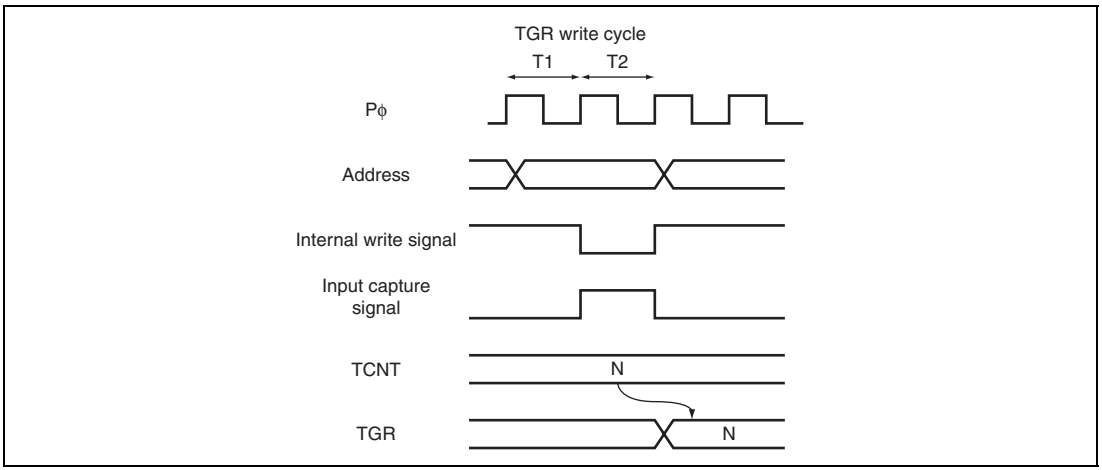


Figure 10.51 Conflict between TGR Write and Input Capture

10.9.10 Conflict between Buffer Register Write and Input Capture

If the input capture signal is generated in the T2 state of a buffer register write cycle, the buffer operation takes precedence and the write to the buffer register is not performed.

Figure 10.52 shows the timing in this case.

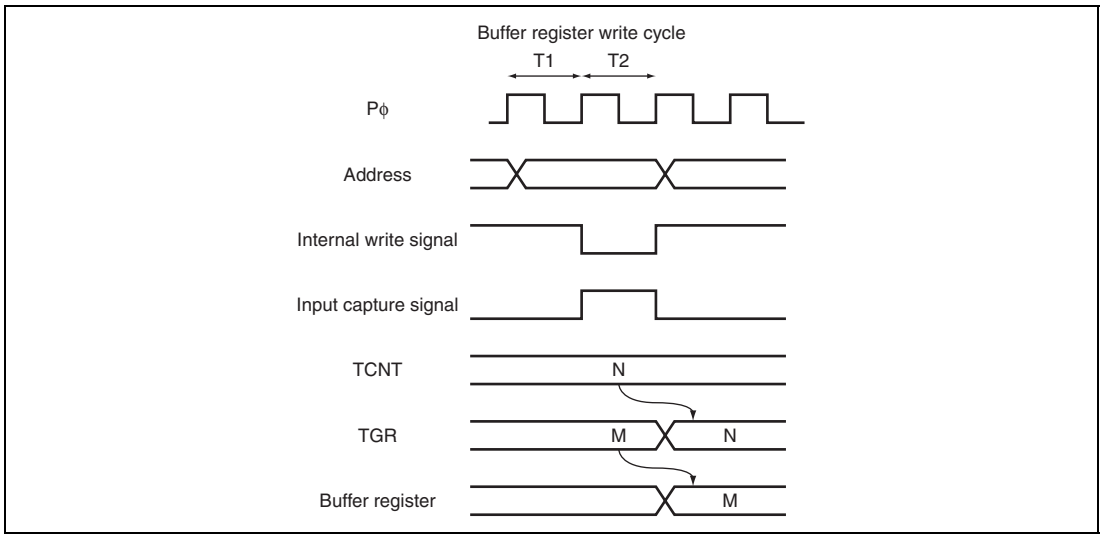


Figure 10.52 Conflict between Buffer Register Write and Input Capture

10.9.11 Conflict between Overflow/Underflow and Counter Clearing

If overflow/underflow and counter clearing occur simultaneously, the TCFV/TCFU flag in TSR is set and TCNT clearing takes precedence.

Figure 10.53 shows the operation timing when a TGR compare match is specified as the clearing source, and H'FFFF is set in TGR.

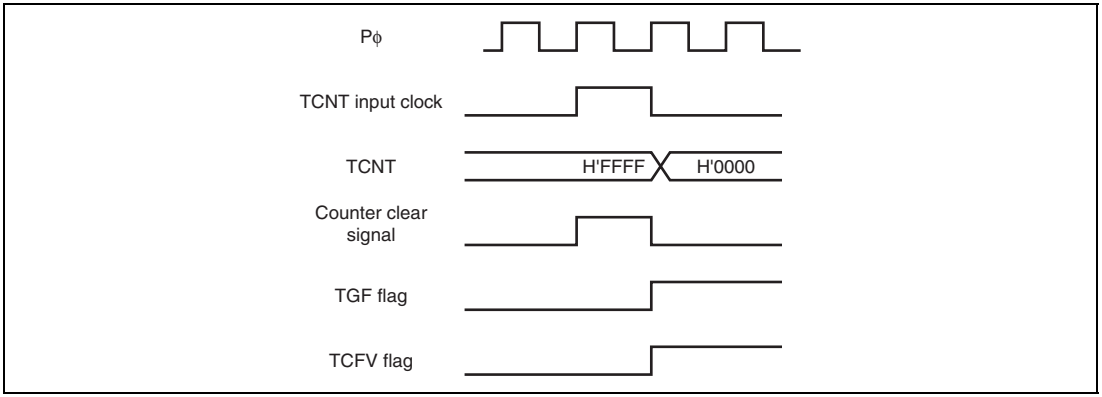


Figure 10.53 Conflict between Overflow and Counter Clearing

10.9.12 Conflict between TCNT Write and Overflow/Underflow

If an overflow/underflow occurs due to increment/decrement in the T2 state of a TCNT write cycle, the TCNT write takes precedence and the TCFV/TCFU flag in TSR is not set.

Figure 10.54 shows the operation timing when there is conflict between TCNT write and overflow.

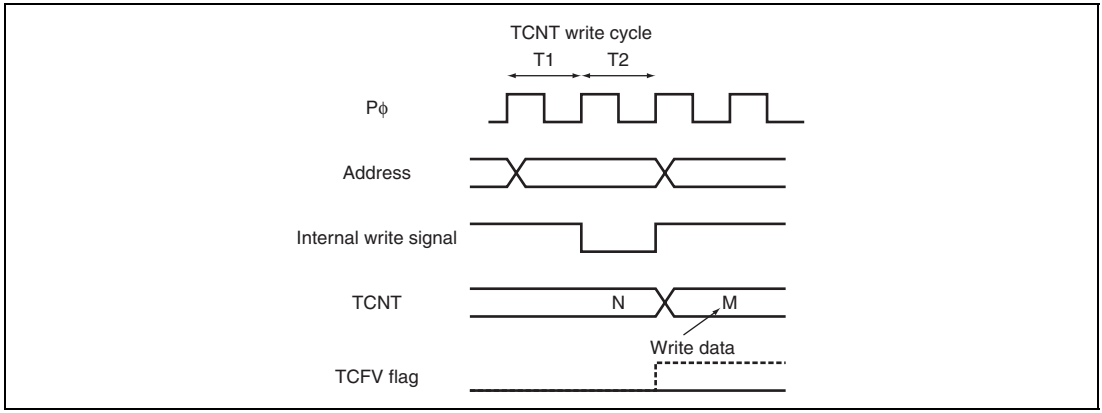


Figure 10.54 Conflict between TCNT Write and Overflow

10.9.13 Interrupts in Module Stop Mode

If module stop mode is entered when an interrupt has been requested, it will not be possible to clear the CPU interrupt source or the DTC activation source. Interrupts should therefore be disabled before entering module stop mode.

Section 11 Programmable Pulse Generator (PPG)

The programmable pulse generator (PPG) provides pulse outputs by using the 16-bit timer pulse unit (TPU) as a time base. The PPG pulse outputs are divided into 4-bit groups (groups 3 and 2) that can operate both simultaneously and independently. Figure 11.1 shows a block diagram of the PPG.

11.1 Features

- 8-bit output data
- Two output groups
- Selectable output trigger signals
- Non-overlapping mode
- Can operate together with the data transfer controller (DTC)
- Inverted output can be set
- Module stop mode can be set

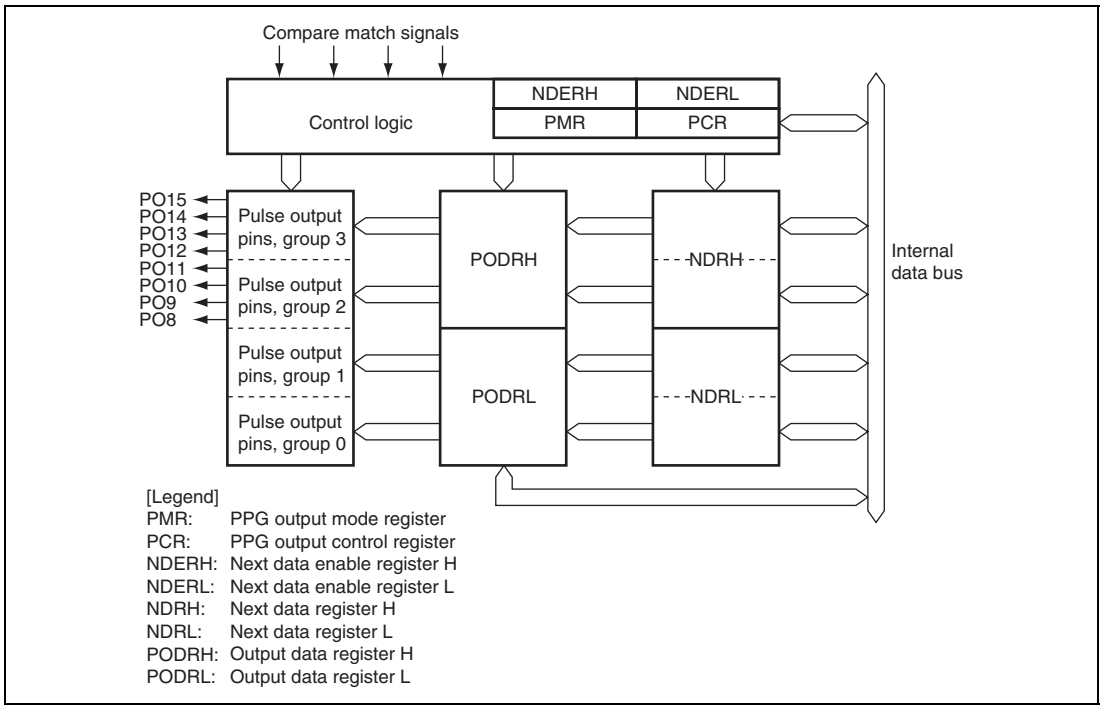


Figure 11.1 Block Diagram of PPG

11.2 Input/Output Pins

Table 11.1 shows the PPG pin configuration.

Table 11.1 Pin Configuration

Pin Name	I/O	Function
PO15	Output	Group 3 pulse output
PO14	Output	
PO13	Output	
PO12	Output	
PO11	Output	Group 2 pulse output
PO10	Output	
PO9	Output	
PO8	Output	

11.3 Register Descriptions

The PPG has the following registers.

- Next data enable register H (NDERH)
- Next data enable register L (NDERL)
- Output data register H (PODRH)
- Output data register L (PODRL)
- Next data register H (NDRH)
- Next data register L (NDRL)
- PPG output control register (PCR)
- PPG output mode register (PMR)

11.3.1 Next Data Enable Registers H, L (NDERH, NDERL)

NDERH and NDERL enable/disable pulse output on a bit-by-bit basis.

- NDERH

Bit	7	6	5	4	3	2	1	0
Bit Name	NDER15	NDER14	NDER13	NDER12	NDER11	NDER10	NDER9	NDER8
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- NDERL

Bit	7	6	5	4	3	2	1	0
Bit Name	NDER7	NDER6	NDER5	NDER4	NDER3	NDER2	NDER1	NDER0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- NDERH

Bit	Bit Name	Initial Value	R/W	Description
7	NDER15	0	R/W	Next Data Enable 15 to 8
6	NDER14	0	R/W	When a bit is set to 1, the value in the corresponding NDRH bit is transferred to the PODRH bit by the selected output trigger. Values are not transferred from NDRH to PODRH for cleared bits.
5	NDER13	0	R/W	
4	NDER12	0	R/W	
3	NDER11	0	R/W	
2	NDER10	0	R/W	
1	NDER9	0	R/W	
0	NDER8	0	R/W	

- NDERL

Bit	Bit Name	Initial Value	R/W	Description
7	NDER7	0	R/W	Next Data Enable 7 to 0
6	NDER6	0	R/W	When a bit is set to 1, the value in the corresponding NDRL bit is transferred to the PODRL bit by the selected output trigger. Values are not transferred from NDRL to PODRL for cleared bits.
5	NDER5	0	R/W	
4	NDER4	0	R/W	
3	NDER3	0	R/W	
2	NDER2	0	R/W	
1	NDER1	0	R/W	
0	NDER0	0	R/W	

11.3.2 Output Data Registers H, L (PODRH, PODRL)

PODRH and PODRL store output data for use in pulse output. A bit that has been set for pulse output by NDER is read-only and cannot be modified.

- PODRH

Bit	7	6	5	4	3	2	1	0
Bit Name	POD15	POD14	POD13	POD12	POD11	POD10	POD9	POD8
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- PODRL

Bit	7	6	5	4	3	2	1	0
Bit Name	POD7	POD6	POD5	POD4	POD3	POD2	POD2	POD0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- PODRH

Bit	Bit Name	Initial Value	R/W	Description
7	POD15	0	R/W	Output Data Register 15 to 8
6	POD14	0	R/W	For bits which have been set to pulse output by NDERH, the output trigger transfers NDRH values to this register during PPG operation. While NDERH is set to 1, the CPU cannot write to this register. While NDERH is cleared, the initial output value of the pulse can be set.
5	POD13	0	R/W	
4	POD12	0	R/W	
3	POD11	0	R/W	
2	POD10	0	R/W	
1	POD9	0	R/W	
0	POD8	0	R/W	

- PODRL

Bit	Bit Name	Initial Value	R/W	Description
7	POD7	0	R/W	Output Data Register 7 to 0
6	POD6	0	R/W	For bits which have been set to pulse output by NDERL, the output trigger transfers NDRL values to this register during PPG operation. While NDERL is set to 1, the CPU cannot write to this register. While NDERL is cleared, the initial output value of the pulse can be set.
5	POD5	0	R/W	
4	POD4	0	R/W	
3	POD3	0	R/W	
2	POD2	0	R/W	
1	POD1	0	R/W	
0	POD0	0	R/W	

11.3.3 Next Data Registers H, L (NDRH, NDRL)

NDRH and NDRL store the next data for pulse output. The NDR addresses differ depending on whether pulse output groups have the same output trigger or different output triggers.

- NDRH

Bit	7	6	5	4	3	2	1	0
Bit Name	NDR15	NDR14	NDR13	NDR12	NDR11	NDR10	NDR9	NDR8
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- NDRL

Bit	7	6	5	4	3	2	1	0
Bit Name	NDR7	NDR6	NDR5	NDR4	NDR3	NDR2	NDR1	NDR0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- NDRH

  If pulse output groups 2 and 3 have the same output trigger, all eight bits are mapped to the same address and can be accessed at one time, as shown below.

Bit	Bit Name	Initial Value	R/W	Description
7	NDR15	0	R/W	Next Data Register 15 to 8
6	NDR14	0	R/W	The register contents are transferred to the corresponding PODRH bits by the output trigger specified with PCR.
5	NDR13	0	R/W	
4	NDR12	0	R/W	
3	NDR11	0	R/W	
2	NDR10	0	R/W	
1	NDR9	0	R/W	
0	NDR8	0	R/W	

If pulse output groups 2 and 3 have different output triggers, the upper four bits and lower four bits are mapped to different addresses as shown below.

Bit	Bit Name	Initial Value	R/W	Description
7	NDR15	0	R/W	Next Data Register 15 to 12
6	NDR14	0	R/W	The register contents are transferred to the corresponding PODRH bits by the output trigger specified with PCR.
5	NDR13	0	R/W	
4	NDR12	0	R/W	
3 to 0	—	All 1	R	Reserved These bits are always read as 1 and cannot be modified.

Bit	Bit Name	Initial Value	R/W	Description
7 to 4	—	All 1	—	Reserved These bits are always read as 1 and cannot be modified.
3	NDR11	0	R/W	Next Data Register 11 to 8
2	NDR10	0	R/W	The register contents are transferred to the corresponding PODRH bits by the output trigger specified with PCR.
1	NDR9	0	R/W	
0	NDR8	0	R/W	

- NDRL

  If pulse output groups 0 and 1 have the same output trigger, all eight bits are mapped to the same address and can be accessed at one time, as shown below.

Bit	Bit Name	Initial Value	R/W	Description
7	NDR7	0	R/W	Next Data Register 7 to 0
6	NDR6	0	R/W	The register contents are transferred to the corresponding PODRL bits by the output trigger specified with PCR.
5	NDR5	0	R/W	
4	NDR4	0	R/W	
3	NDR3	0	R/W	
2	NDR2	0	R/W	
1	NDR1	0	R/W	
0	NDR0	0	R/W	

If pulse output groups 0 and 1 have different output triggers, the upper four bits and lower four bits are mapped to different addresses as shown below.

Bit	Bit Name	Initial Value	R/W	Description
7	NDR7	0	R/W	Next Data Register 7 to 4
6	NDR6	0	R/W	The register contents are transferred to the corresponding PODRL bits by the output trigger specified with PCR.
5	NDR5	0	R/W	
4	NDR4	0	R/W	
3 to 0	—	All 1	—	Reserved These bits are always read as 1 and cannot be modified.

Bit	Bit Name	Initial Value	R/W	Description
7 to 4	—	All 1	—	Reserved These bits are always read as 1 and cannot be modified.
3	NDR3	0	R/W	Next Data Register 3 to 0
2	NDR2	0	R/W	The register contents are transferred to the corresponding PODRL bits by the output trigger specified with PCR.
1	NDR1	0	R/W	
0	NDR0	0	R/W	

11.3.4 PPG Output Control Register (PCR)

PCR selects pulse output trigger signals on a group-by-group basis. For details on output trigger selection, see section 11.3.5, PPG Output Mode Register (PMR).

Bit	7	6	5	4	3	2	1	0
Bit Name	G3CMS1	G3CMS0	G2CMS1	G2CMS0	–	–	–	–
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	G3CMS1	1	R/W	Group 3 Compare Match Select 1 and 0
6	G3CMS0	1	R/W	These bits select output trigger of pulse output group 3. 00: Compare match in TPU channel 0 01: Compare match in TPU channel 1 10: Compare match in TPU channel 2 11: Compare match in TPU channel 3
5	G2CMS1	1	R/W	Group 2 Compare Match Select 1 and 0
4	G2CMS0	1	R/W	These bits select output trigger of pulse output group 2. 00: Compare match in TPU channel 0 01: Compare match in TPU channel 1 10: Compare match in TPU channel 2 11: Compare match in TPU channel 3
3 to 0	—	All 1	R/W	Reserved These bits are always read as 1. The write value should always be 1.

11.3.5 PPG Output Mode Register (PMR)

PMR selects the pulse output mode of the PPG for each group. If inverted output is selected, a low-level pulse is output when PODRH is 1 and a high-level pulse is output when PODRH is 0. If non-overlapping operation is selected, PPG updates its output values at compare match A or B of the TPU that becomes the output trigger. For details, see section 11.4.4, Non-Overlapping Pulse Output.

Bit	7	6	5	4	3	2	1	0
Bit Name	G3INV	G2INV	–	–	G3NOV	G2NOV	–	–
Initial Value	1	1	1	1	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	G3INV	1	R/W	Group 3 Inversion Selects direct output or inverted output for pulse output group 3. 0: Inverted output 1: Direct output
6	G2INV	1	R/W	Group 2 Inversion Selects direct output or inverted output for pulse output group 2. 0: Inverted output 1: Direct output
5, 4	—	All 1	R/W	Reserved These bits are always read as 1. The write value should always be 1.
3	G3NOV	0	R/W	Group 3 Non-Overlap Selects normal or non-overlapping operation for pulse output group 3. 0: Normal operation (output values updated at compare match A in the selected TPU channel) 1: Non-overlapping operation (output values updated at compare match A or B in the selected TPU channel)

Bit	Bit Name	Initial Value	R/W	Description
2	G2NOV	0	R/W	Group 2 Non-Overlap Selects normal or non-overlapping operation for pulse output group 2. 0: Normal operation (output values updated at compare match A in the selected TPU channel) 1: Non-overlapping operation (output values updated at compare match A or B in the selected TPU channel)
1, 0	—	All 0	R/W	Reserved These bits are always read as 0. The write value should always be 0.

11.4 Operation

Figure 11.2 shows a schematic diagram of the PPG. PPG pulse output is enabled when the corresponding bits in NDER are set to 1. An initial output value is determined by its corresponding PODR initial setting. When the compare match event specified by PCR occurs, the corresponding NDR bit contents are transferred to PODR to update the output values. Sequential output of data of up to 8 bits is possible by writing new output data to NDR before the next compare match.

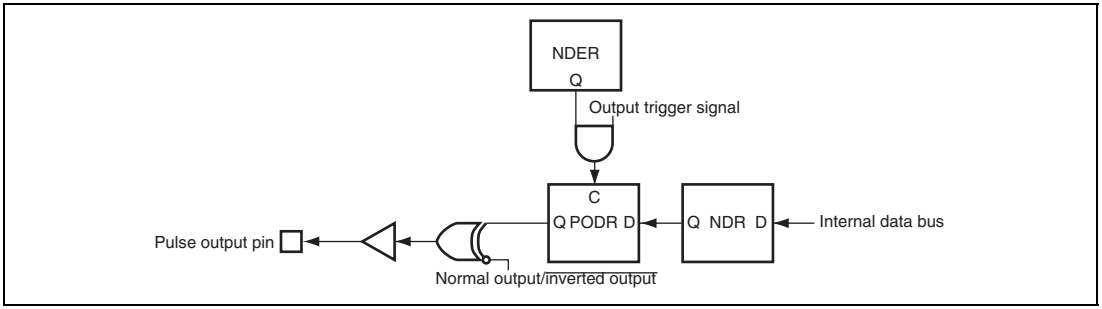


Figure 11.2 Schematic Diagram of PPG

11.4.1 Output Timing

If pulse output is enabled, the NDR contents are transferred to PODR and output when the specified compare match event occurs. Figure 11.3 shows the timing of these operations for the case of normal output in groups 2 and 3, triggered by compare match A.

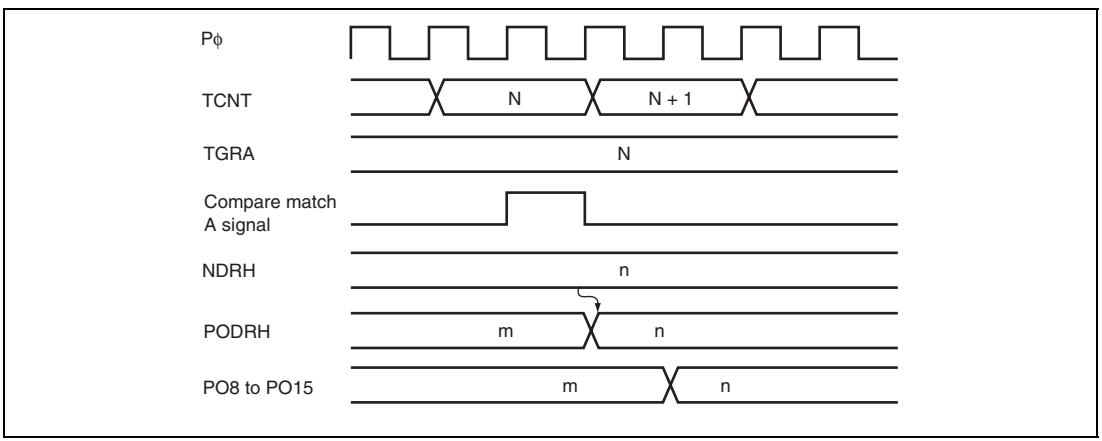


Figure 11.3 Timing of Transfer and Output of NDR Contents (Example)

11.4.2 Sample Setup Procedure for Normal Pulse Output

Figure 11.4 shows a sample procedure for setting up normal pulse output.

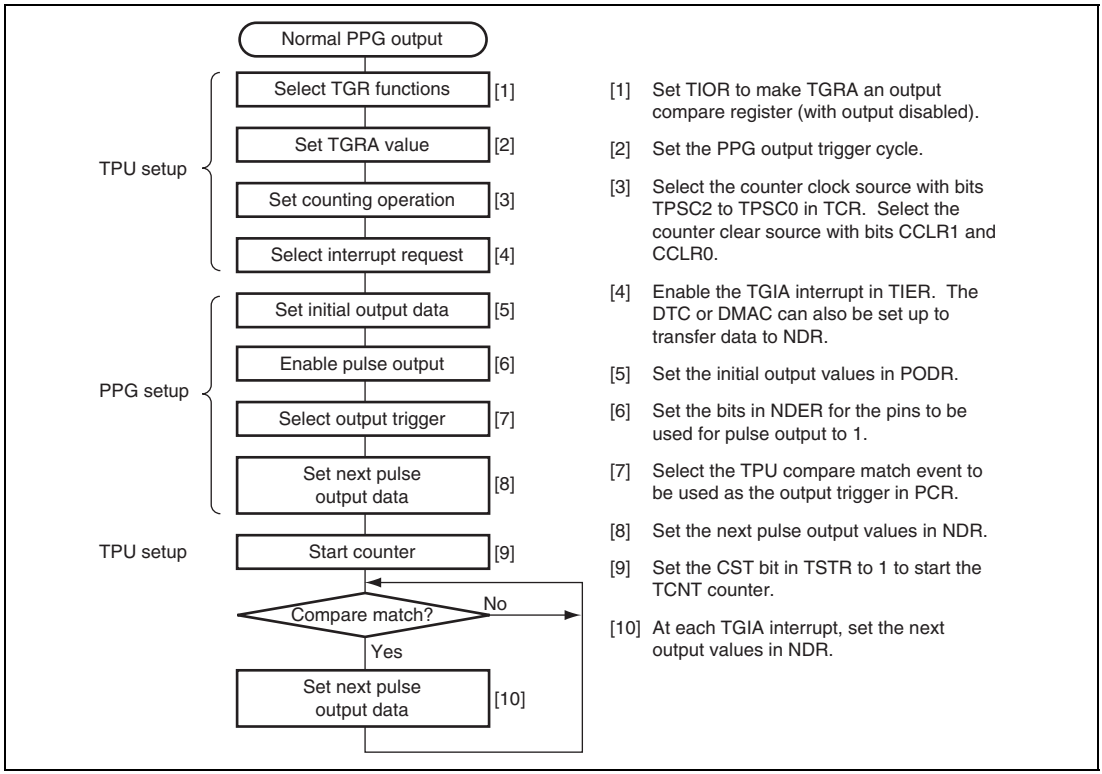


Figure 11.4 Setup Procedure for Normal Pulse Output (Example)

11.4.3 Example of Normal Pulse Output (Example of 5-Phase Pulse Output)

Figure 11.5 shows an example in which pulse output is used for cyclic 5-phase pulse output.

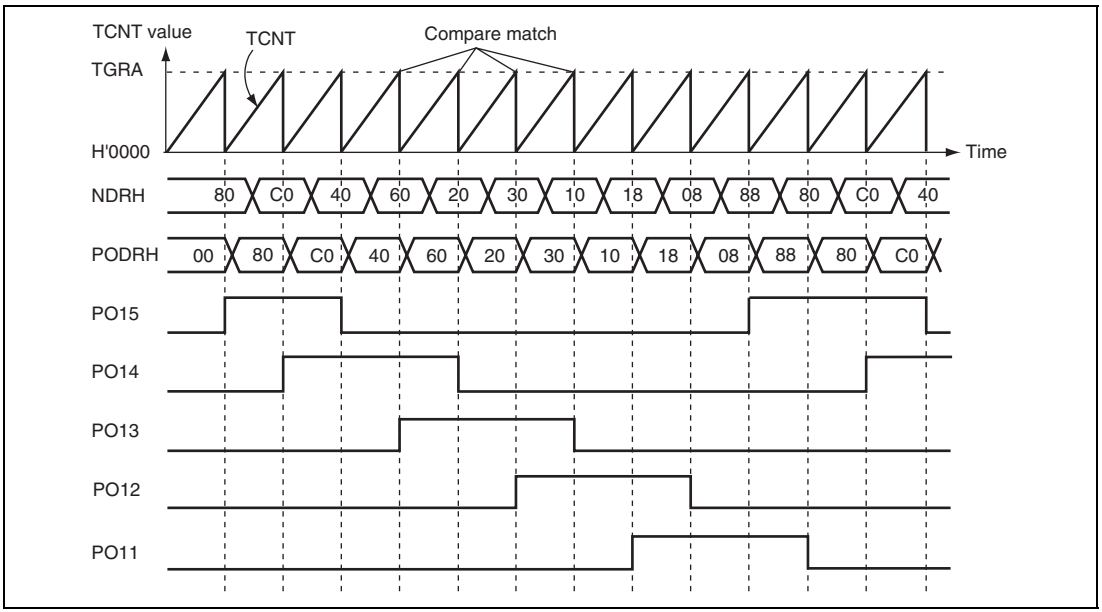


Figure 11.5 Normal Pulse Output Example (5-Phase Pulse Output)

1. Set up TGRA in TPU which is used as the output trigger to be an output compare register. Set a cycle in TGRA so the counter will be cleared by compare match A. Set the TGIEA bit in TIER to 1 to enable the compare match/input capture A (TGIA) interrupt.
2. Write H'F8 to NDERH, and set bits G3CMS1, G3CMS0, G2CMS1, and G2CMS0 in PCR to select compare match in the TPU channel set up in the previous step to be the output trigger. Write output data H'80 in NDRH.
3. The timer counter in the TPU channel starts. When compare match A occurs, the NDRH contents are transferred to PODRH and output. The TGIA interrupt handling routine writes the next output data (H'C0) in NDRH.
4. 5-phase pulse output (one or two phases active at a time) can be obtained subsequently by writing H'40, H'60, H'20, H'30, H'10, H'18, H'08, H'88... at successive TGIA interrupts.

   If the DTC is set for activation by the TGIA interrupt, pulse output can be obtained without imposing a load on the CPU.

11.4.4 Non-Overlapping Pulse Output

During non-overlapping operation, transfer from NDR to PODR is performed as follows:

- At compare match A, the NDR bits are always transferred to PODR.
- At compare match B, the NDR bits are transferred only if their value is 0. The NDR bits are not transferred if their value is 1.

Figure 11.6 illustrates the non-overlapping pulse output operation.

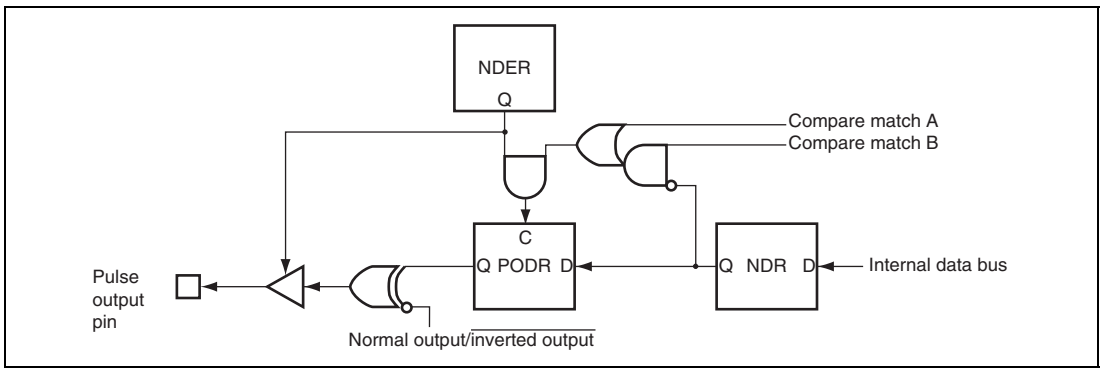


Figure 11.6 Non-Overlapping Pulse Output

Therefore, 0 data can be transferred ahead of 1 data by making compare match B occur before compare match A.

The NDR contents should not be altered during the interval from compare match B to compare match A (the non-overlapping margin).

This can be accomplished by having the TGIA interrupt handling routine write the next data in NDR, or by having the TGIA interrupt activate the DTC. Note, however, that the next data must be written before the next compare match B occurs.

Figure 11.7 shows the timing of this operation.

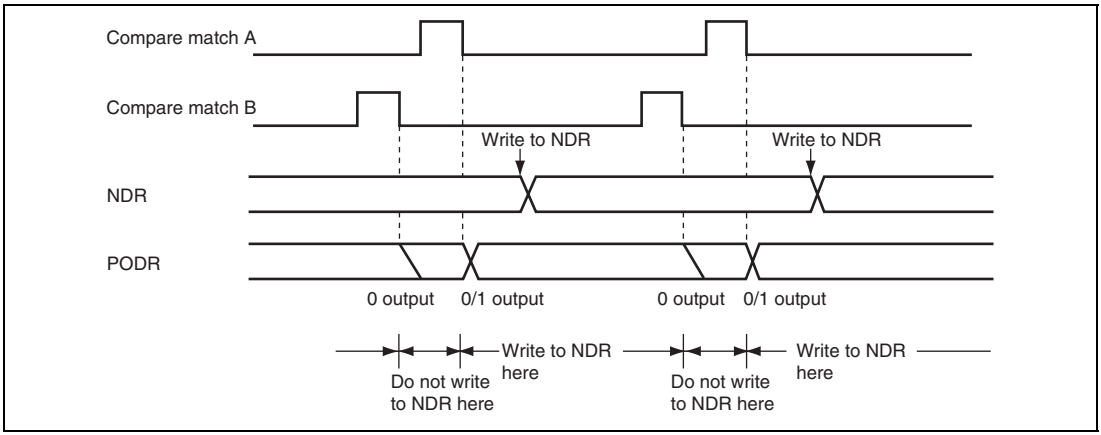


Figure 11.7 Non-Overlapping Operation and NDR Write Timing

11.4.5 Sample Setup Procedure for Non-Overlapping Pulse Output

Figure 11.8 shows a sample procedure for setting up non-overlapping pulse output.

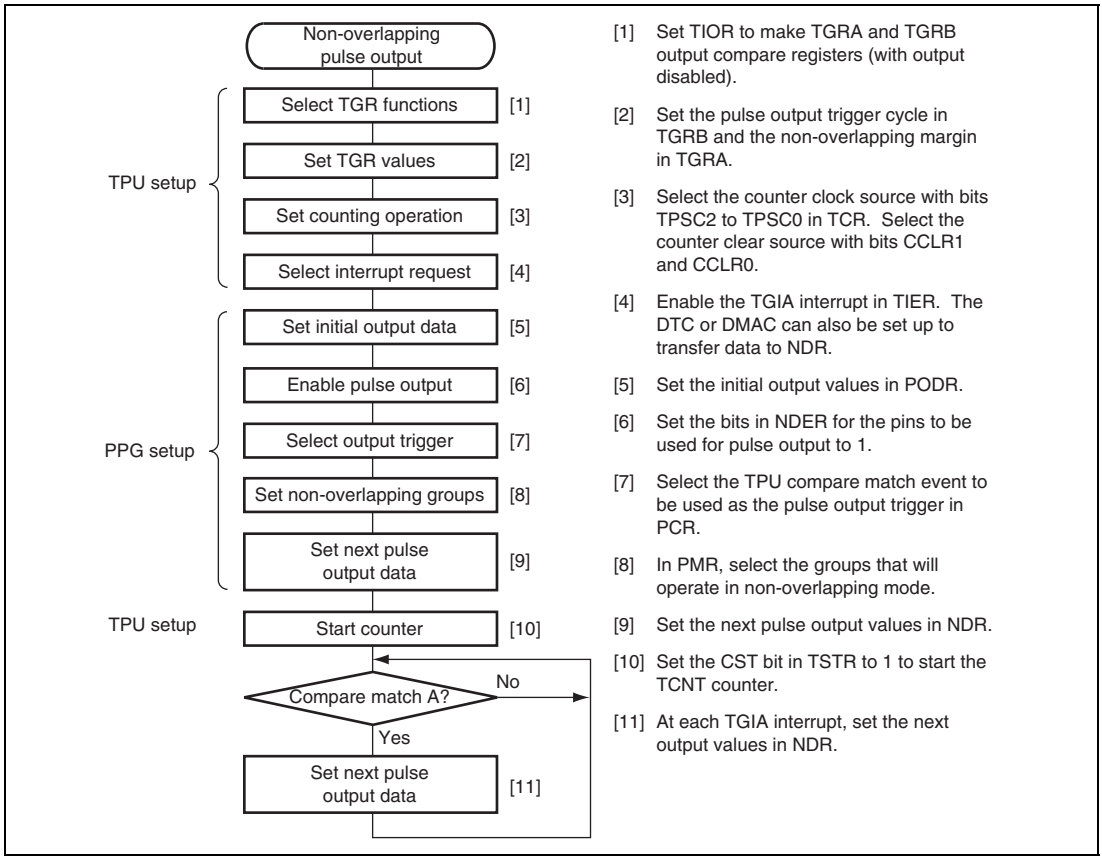


Figure 11.8 Setup Procedure for Non-Overlapping Pulse Output (Example)

11.4.6 Example of Non-Overlapping Pulse Output (Example of 4-Phase Complementary Non-Overlapping Pulse Output)

Figure 11.9 shows an example in which pulse output is used for 4-phase complementary non-overlapping pulse output.

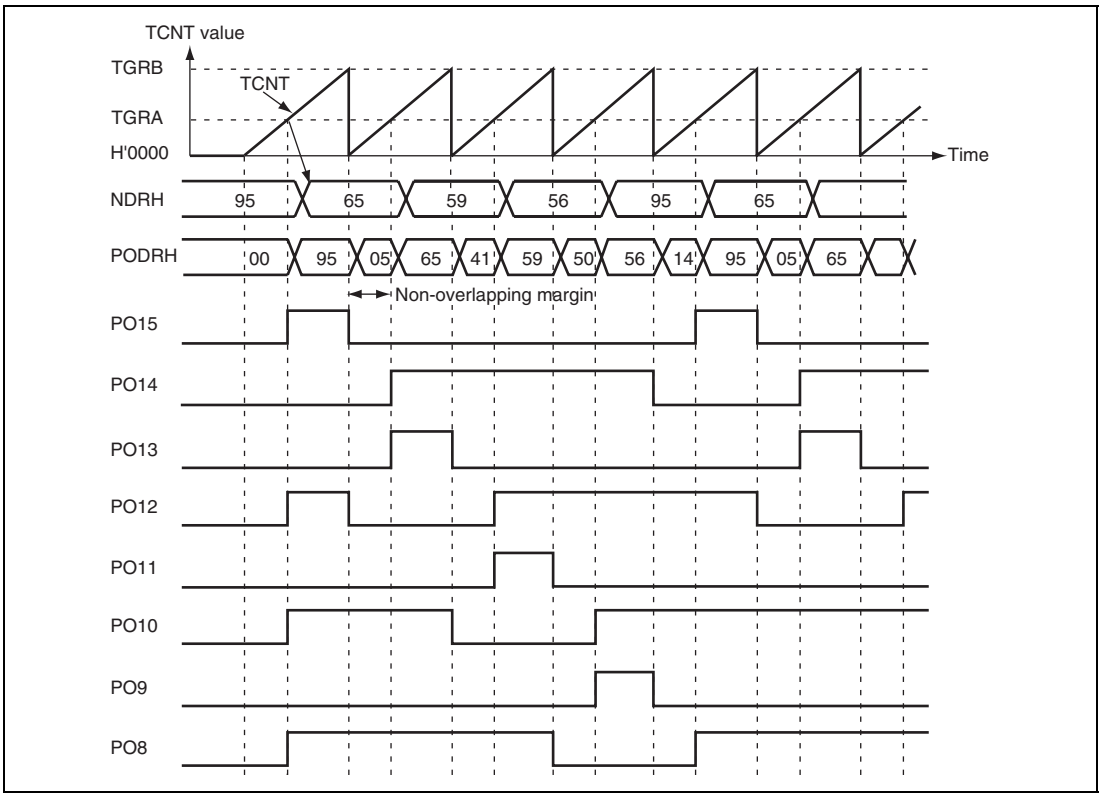


Figure 11.9 Non-Overlapping Pulse Output Example (4-Phase Complementary)

1. Set up the TPU channel to be used as the output trigger channel so that TGRA and TGRB are output compare registers. Set the cycle in TGRB and the non-overlapping margin in TGRA, and set the counter to be cleared by compare match B. Set the TGIEA bit in TIER to 1 to enable the TGIA interrupt.
2. Write H'FF to NDERH, and set bits G3CMS1, G3CMS0, G2CMS1, and G2CMS0 in PCR to select compare match in the TPU channel set up in the previous step to be the output trigger.

   Set bits G3NOV and G2NOV in PMR to 1 to select non-overlapping pulse output.

   Write output data H'95 to NDRH.
3. The timer counter in the TPU channel starts. When a compare match with TGRB occurs, outputs change from 1 to 0. When a compare match with TGRA occurs, outputs change from 0 to 1 (the change from 0 to 1 is delayed by the value set in TGRA).

   The TGIA interrupt handling routine writes the next output data (H'65) to NDRH.
4. 4-phase complementary non-overlapping pulse output can be obtained subsequently by writing H'59, H'56, H'95... at successive TGIA interrupts.

   If the DTC is set for activation by a TGIA interrupt, pulse can be output without imposing a load on the CPU.

11.4.7 Inverted Pulse Output

If the G3INV and G2INV bits in PMR are cleared to 0, values that are the inverse of the PODR contents can be output.

Figure 11.10 shows the outputs when the G3INV and G2INV bits are cleared to 0, in addition to the settings of figure 11.9.

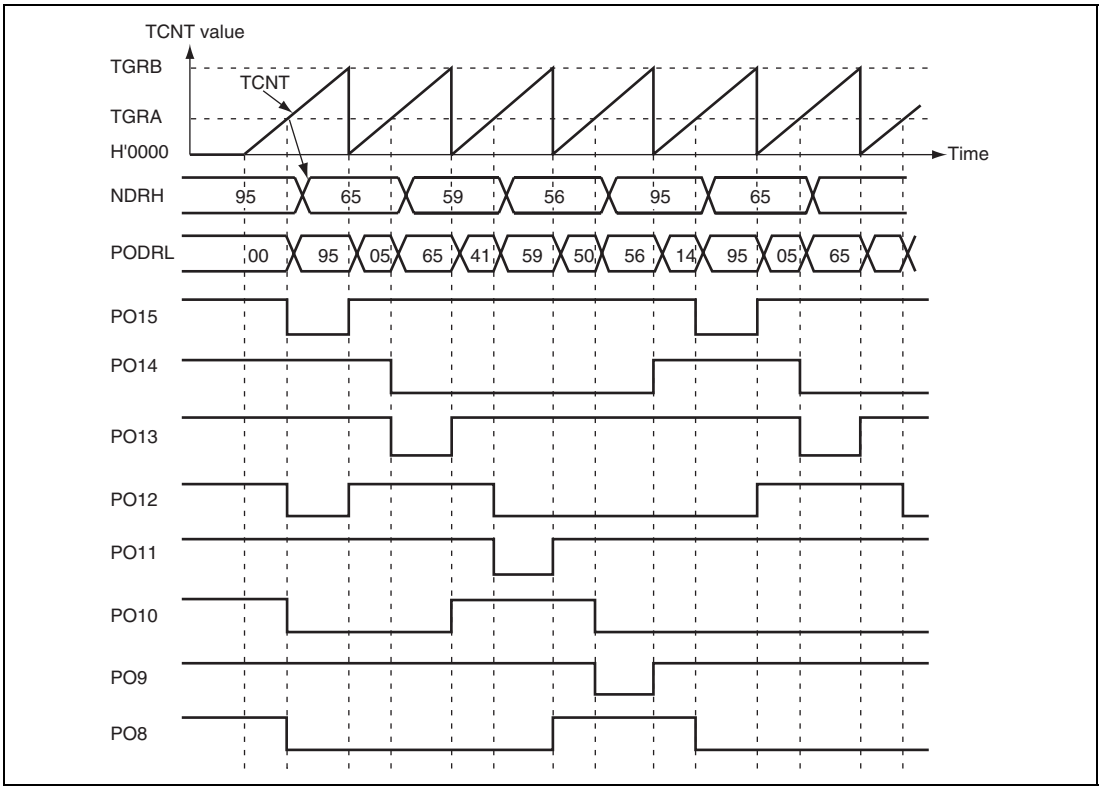


Figure 11.10 Inverted Pulse Output (Example)

11.4.8 Pulse Output Triggered by Input Capture

Pulse output can be triggered by TPU input capture as well as by compare match. If TGRA functions as an input capture register in the TPU channel selected by PCR, pulse output will be triggered by the input capture signal.

Figure 11.11 shows the timing of this output.

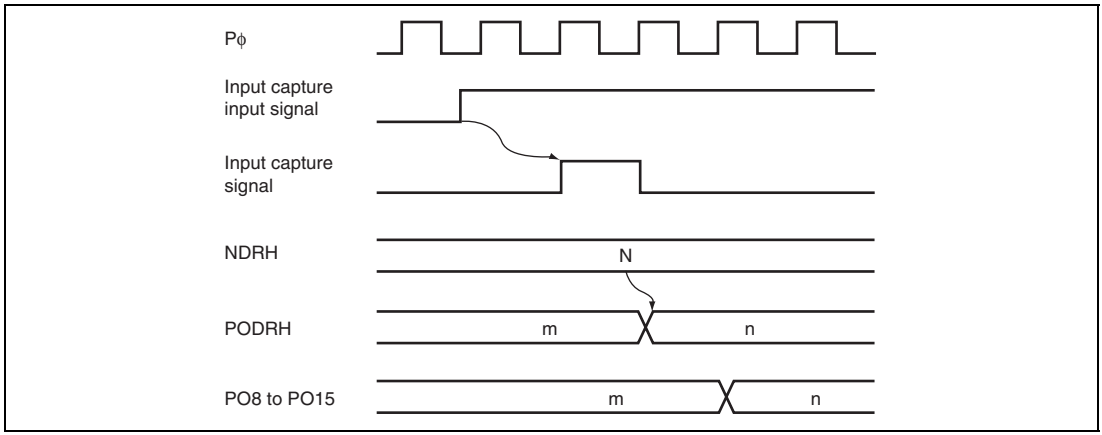


Figure 11.11 Pulse Output Triggered by Input Capture (Example)

11.5 Usage Notes

11.5.1 Module Stop Mode Setting

Operation of the PPG can be enabled or disabled using the module stop control register. The initial setting is for operation of the PPG to be halted. Register access is enabled by clearing module stop mode. For details, see section 23, Power-Down Modes.

11.5.2 Operation of Pulse Output Pins

Pin functions PO8 to PO15 are multiplexed on the same pins as pin functions for other peripheral modules such as the TPU. Pulse output from any of these pins is impossible if the output from another peripheral module is enabled. However, transfer from NDR to PODR is always possible regardless of the pin states.

When changing a pin function, ensure that trigger outputs will not be generated from these pins.

Section 12 Watchdog Timer (WDT)

The watchdog timer (WDT) is an 8-bit timer that outputs an internal reset signal on overflow of the counter when the value of the counter has not been updated because of a system runaway.

When this watchdog function is not needed, the WDT can be used as an interval timer. In interval timer operation, an interval timer interrupt is generated each time the counter overflows.

Figure 12.1 shows a block diagram of the WDT.

12.1 Features

- Selectable from eight counter input clocks
- Switchable between watchdog timer mode and interval timer mode
  - In watchdog timer mode
    If the counter overflows, this LSI can be initialized internally.
  - In interval timer mode
    If the counter overflows, the WDT generates an interval timer interrupt (WOVI).

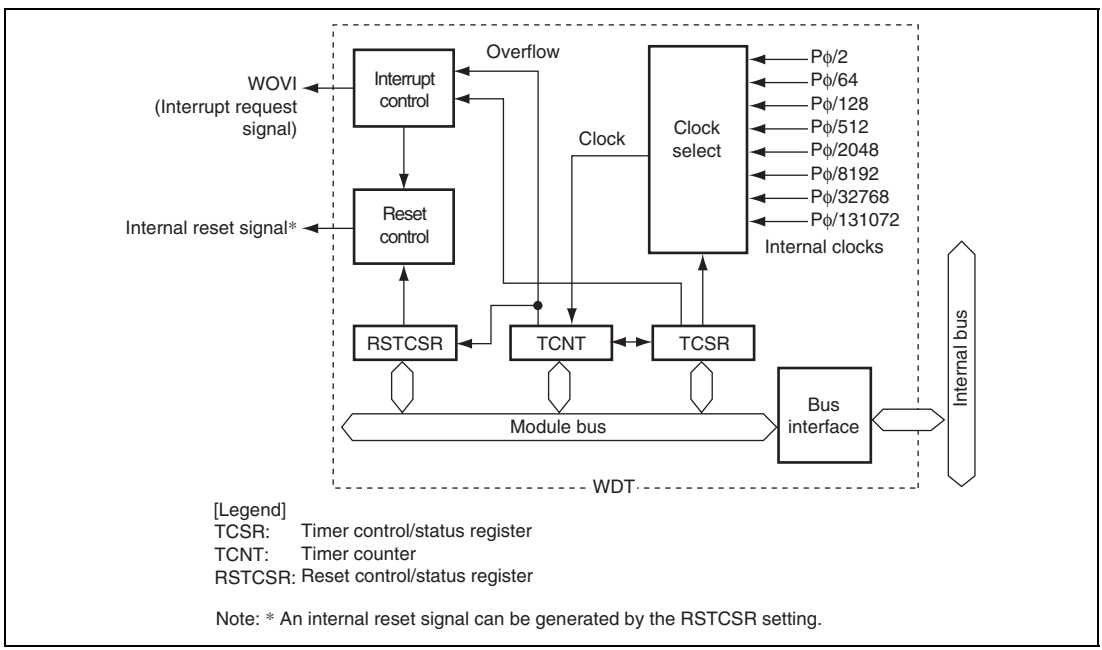


Figure 12.1 Block Diagram of WDT

12.2 Register Descriptions

The WDT has the following three registers. To prevent accidental overwriting, TCNT, TCSR, and RSTCSR have to be written to in a method different from normal registers. For details, see section 12.5.1, Notes on Register Access.

- Timer counter (TCNT)
- Timer control/status register (TCSR)
- Reset control/status register (RSTCSR)

12.2.1 Timer Counter (TCNT)

TCNT is an 8-bit readable/writable up-counter. TCNT is initialized to H'00 when the TME bit in TCSR is cleared to 0.

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

12.2.2 Timer Control/Status Register (TCSR)

TCSR selects the clock source to be input to TCNT, and the timer mode.

Bit	7	6	5	4	3	2	1	0
Bit Name	OVF	WT/$\overline{IT}$	TME	—	—	CKS2	CKS1	CKS0
Initial Value	0	0	0	1	1	0	0	0
R/W	R/(W)*	R/W	R/W	R	R	R/W	R/W	R/W

Note: * Only 0 can be written to clear the flag.

Bit	Bit Name	Initial Value	R/W	Description
7	OVF	0	R/(W)*	Overflow Flag Indicates that TCNT has overflowed in interval timer mode. Only 0 can be written to this bit to clear the flag. [Setting condition] When TCNT overflows in interval timer mode (changes from H'FF to H'00) When internal reset request generation is selected in watchdog timer mode, OVF is cleared automatically by the internal reset. [Clearing condition] Cleared by reading TCSR when OVF = 1, then writing 0 to OVF (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.)
6	WT/$\overline{IT}$	0	R/W	Timer Mode Select Selects whether the WDT is used as a watchdog timer or interval timer. 0: Interval timer mode When TCNT overflows, an interval timer interrupt (WOVI) is requested. 1: Watchdog timer mode When TCNT overflows while RSTE = 1, this LSI is initialized internally.
5	TME	0	R/W	Timer Enable When this bit is set to 1, TCNT starts counting. When this bit is cleared, TCNT stops counting and is initialized to H'00.
4, 3	—	All 1	R	Reserved These are read-only bits and cannot be modified.

Bit	Bit Name	Initial Value	R/W	Description
2	CKS2	0	R/W	Clock Select 2 to 0
1	CKS1	0	R/W	Select the clock source to be input to TCNT. The overflow cycle for Pϕ = 20 MHz is indicated in parentheses.
0	CKS0	0	R/W	000: Clock Pϕ/2 (cycle: 25.6 μs)
				001: Clock Pϕ/64 (cycle: 819.2 μs)
				010: Clock Pϕ/128 (cycle: 1.6 ms)
				011: Clock Pϕ/512 (cycle: 6.6 ms)
				100: Clock Pϕ/2048 (cycle: 26.2 ms)
				101: Clock Pϕ/8192 (cycle: 104.9 ms)
				110: Clock Pϕ/32768 (cycle: 419.4 ms)
				111: Clock Pϕ/131072 (cycle: 1.68 s)

Note: * Only 0 can be written to clear the flag.

12.2.3 Reset Control/Status Register (RSTCSR)

RSTCSR controls the generation of the internal reset signal when TCNT overflows, and selects the type of internal reset signal. RSTCSR is initialized to H'1F by a reset signal from the $\overline{\text{RES}}$ pin, but not by the WDT internal reset signal caused by WDT overflows.

Bit	7	6	5	4	3	2	1	0
Bit Name	WOVF	RSTE	—	—	—	—	—	—
Initial Value	0	0	0	1	1	1	1	1
R/W	R/(W)*	R/W	R/W	R	R	R	R	R

Note: * Only 0 can be written to clear the flag.

Bit	Bit Name	Initial Value	R/W	Description
7	WOVF	0	R/(W)*	Watchdog Timer Overflow Flag This bit is set when TCNT overflows in watchdog timer mode. This bit cannot be set in interval timer mode, and only 0 can be written. [Setting condition] When TCNT overflows (changed from H'FF to H'00) in watchdog timer mode [Clearing condition] Reading RSTCSR when WOVF = 1, and then writing 0 to WOVF (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.)
6	RSTE	0	R/W	Reset Enable Specifies whether or not this LSI is internally reset if TCNT overflows during watchdog timer operation. 0: LSI is not reset even if TCNT overflows (Though this LSI is not reset, TCNT and TCSR in WDT are reset) 1: LSI is reset if TCNT overflows
5	—	0	R/W	Reserved This bit is always read as 0. The write value should always be 0.
4 to 0	—	All 1	R	Reserved These are read-only bits and cannot be modified.

Note: * Only 0 can be written to clear the flag.

12.3 Operation

12.3.1 Watchdog Timer Mode

To use the WDT in watchdog timer mode, set both the WT/$\overline{IT}$ and TME bits in TCSR to 1.

When TCNT overflows in watchdog timer mode, the WOVF bit in RSTCSR is set to 1.

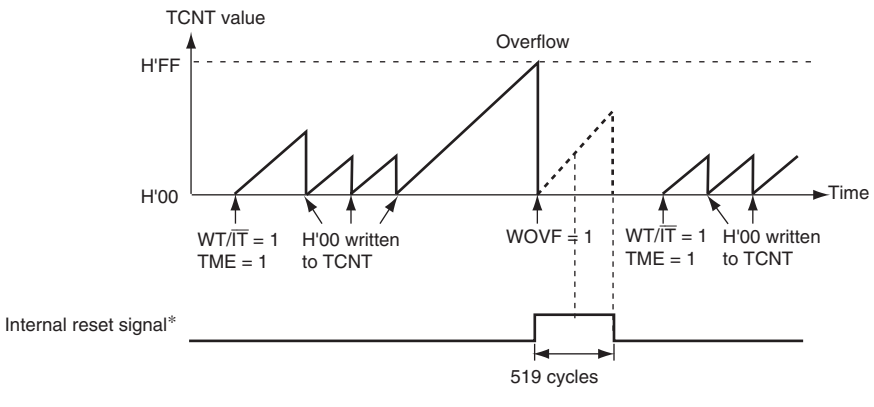
When the watchdog timer mode is selected and the RSTE bit in RSTCSR is set to 1, if TCNT overflows without being rewritten because of a system crash or other error, this LSI is initialized internally. This ensures that TCNT does not overflow while the system is operating normally. Software must prevent TCNT overflows by rewriting the TCNT value (normally H'00 is written) before overflow occurs.

If a reset caused by a signal input to the $\overline{RES}$ pin occurs at the same time as a reset caused by a WDT overflow (TCNT has overflowed), the $\overline{RES}$ pin reset has priority and the WOVF bit in RSTCSR is cleared to 0.

The internal reset signal is output for 519 cycles of Pϕ.

When RSTE = 1, a signal to initialize this LSI internally is generated. Since this signal initializes the system clock control register (SCKCR), the multiplication ratio of Pϕ clock is also initialized.

When RSTE = 0, the signal is not generated, meaning that the SCKCR value and multiplication ratio of Pϕ clock remain unchanged.



Note: * If TCNT overflows when the RSTE bit is set to 1, an internal reset signal is generated.

Figure 12.2 Operation in Watchdog Timer Mode

12.3.2 Interval Timer Mode

To use the WDT as an interval timer, set the WT/$\overline{IT}$ bit to 0 and the TME bit to 1 in TCSR.

When the WDT is used as an interval timer, an interval timer interrupt (WOVI) is generated each time the TCNT overflows. Therefore, an interrupt can be generated at intervals.

When the TCNT overflows in interval timer mode, an interval timer interrupt (WOVI) is requested at the same time the OVF bit in the TCSR is set to 1.

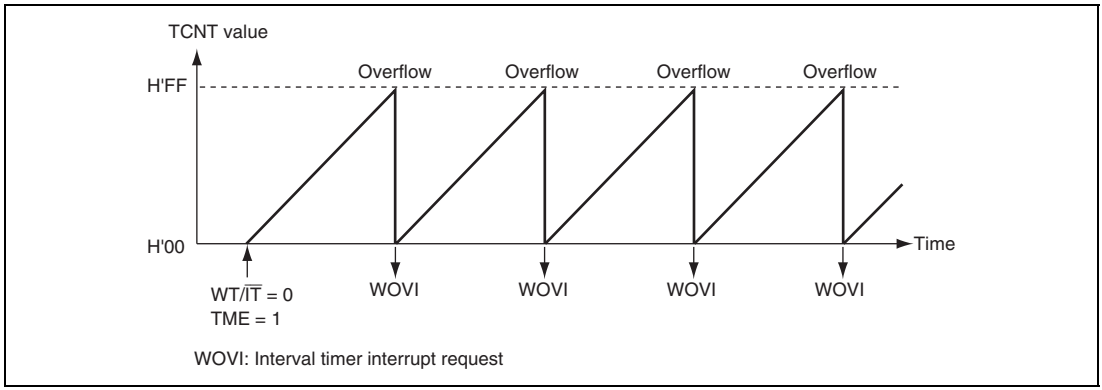


Figure 12.3 Operation in Interval Timer Mode

12.4 Interrupt Source

During interval timer mode operation, an overflow generates an interval timer interrupt (WOVI). The interval timer interrupt is requested whenever the OVF flag is set to 1 in TCSR. The OVF flag must be cleared to 0 in the interrupt handling routine.

Table 12.1 WDT Interrupt Source

Name	Interrupt Source	Interrupt Flag
WOVI	TCNT overflow	OVF

12.5 Usage Notes

12.5.1 Notes on Register Access

To avoid erroneous rewriting, the way of writing to TCNT, TCSR, and RSTCSR registers is different from that to other general registers. The procedures for writing to and reading these registers are given below.

(1) Writing to TCNT, TCSR, and RSTCSR

TCNT and TCSR must be written to by a word transfer instruction. They cannot be written to by a byte transfer instruction.

For writing, TCNT and TCSR are assigned to the same address. Accordingly, perform data transfer as shown in figure 12.4. The transfer instruction writes the lower byte data to TCNT or TCSR.

To write to RSTCSR, execute a word transfer instruction for address H'FFA6. A byte transfer instruction cannot be used to write to RSTCSR.

The method of writing 0 to the WOVF bit in RSTCSR differs from that of writing to the RSTE bit in RSTCSR. Perform data transfer as shown in figure 12.4.

At data transfer, the transfer instruction clears the WOVF bit to 0, but has no effect on the RSTE bit. To write to the RSTE bit, perform data transfer as shown in figure 12.4. In this case, the transfer instruction writes the value in bit 6 of the lower byte to the RSTE bit, but has no effect on the WOVF bit.

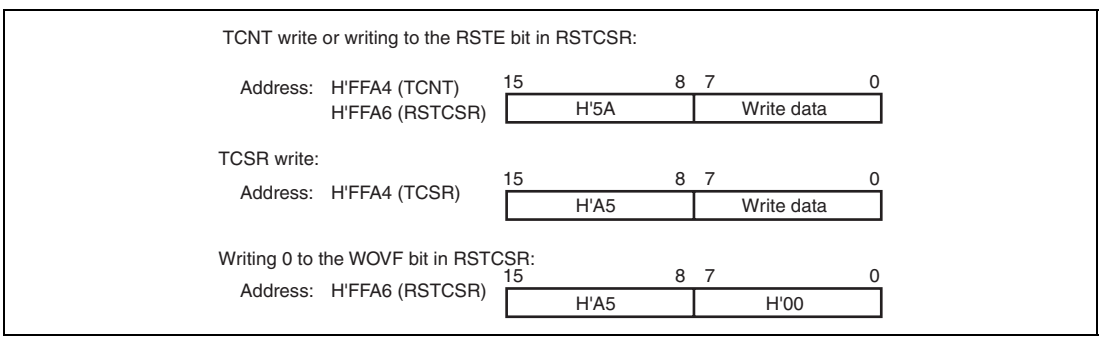


Figure 12.4 Writing to TCNT, TCSR, and RSTCSR

(2) Reading from TCNT, TCSR, and RSTCSR

These registers can be read from in the same way as other registers. For reading, TCSR is assigned to address H'FFA4, TCNT to address H'FFA5, and RSTCSR to address H'FFA7.

12.5.2 Conflict between Timer Counter (TCNT) Write and Increment

If a TCNT clock pulse is generated during the T2 state of a TCNT write cycle, the write takes priority and the timer counter is not incremented. Figure 12.5 shows this operation.

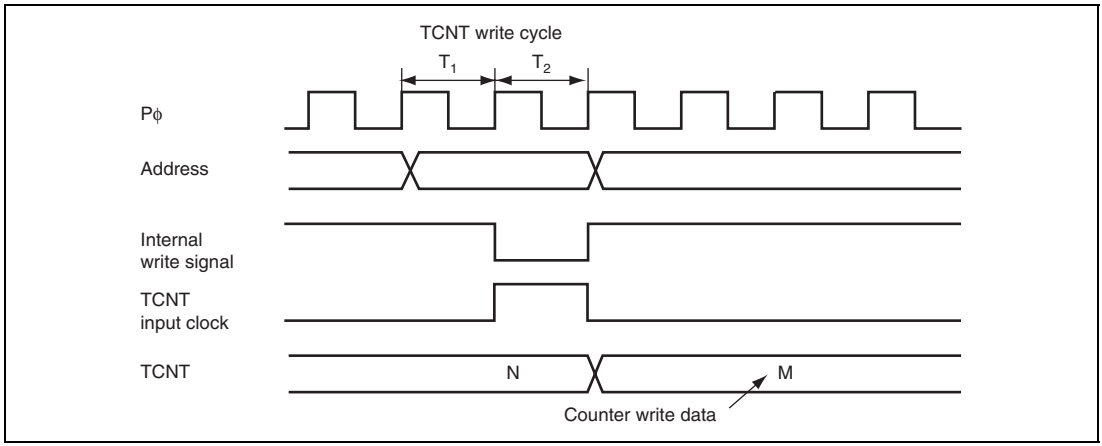


Figure 12.5 Conflict between TCNT Write and Increment

12.5.3 Changing Values of Bits CKS2 to CKS0

If bits CKS2 to CKS0 in TCSR are written to while the WDT is operating, errors could occur in the incrementation. The watchdog timer must be stopped (by clearing the TME bit to 0) before the values of bits CKS2 to CKS0 are changed.

12.5.4 Switching between Watchdog Timer Mode and Interval Timer Mode

If the timer mode is switched from watchdog timer mode to interval timer mode while the WDT is operating, errors could occur in the incrementation. The watchdog timer must be stopped (by clearing the TME bit to 0) before switching the timer mode.

12.5.5 Transition to Watchdog Timer Mode or Software Standby Mode

When the WDT operates in watchdog timer mode, a transition to software standby mode is not made even when the SLEEP instruction is executed when the SSBY bit in SBYCR is set to 1. Instead, a transition to sleep mode is made.

To transit to software standby mode, the SLEEP instruction must be executed after halting the WDT (clearing the TME bit to 0).

When the WDT operates in interval timer mode, a transition to software standby mode is made through execution of the SLEEP instruction when the SSBY bit in SBYCR is set to 1.

Section 13 Serial Communication Interface (SCI)

This LSI has two independent serial communication interface (SCI) channels. The SCI can handle both asynchronous and clocked synchronous serial communication. Asynchronous serial data communication can be carried out with standard asynchronous communication chips such as a Universal Asynchronous Receiver/Transmitter (UART) or Asynchronous Communication Interface Adapter (ACIA). In asynchronous serial communication mode, a function is also provided for serial communication between processors (multiprocessor communication function).

13.1 Features

- Choice of asynchronous or clocked synchronous serial communication mode
- Full-duplex communication capability

  The transmitter and receiver are mutually independent, enabling transmission and reception to be executed simultaneously. Double-buffering is used in both the transmitter and the receiver, enabling continuous transmission and continuous reception of serial data.
- On-chip baud rate generator allows selection of the desired bit rate

  The external clock can be selected as a transfer clock source.
- Four interrupt sources

  The interrupt sources are transmit-end, transmit-data-empty, receive-data-full, and receive error. The transmit-data-empty and receive-data-full interrupt sources can activate the DTC or DMAC.
- Module stop mode can be set

Asynchronous Mode:

- Data length: 7 or 8 bits
- Stop bit length: 1 or 2 bits
- Parity: Even, odd, or none
- Detected error reception: Parity, overrun, and framing errors
- Break detection: Break can be detected by reading the RxD pin level directly in case of a framing error

Clocked Synchronous Mode:

- Data length: 8 bits
- Detected error reception: Overrun errors

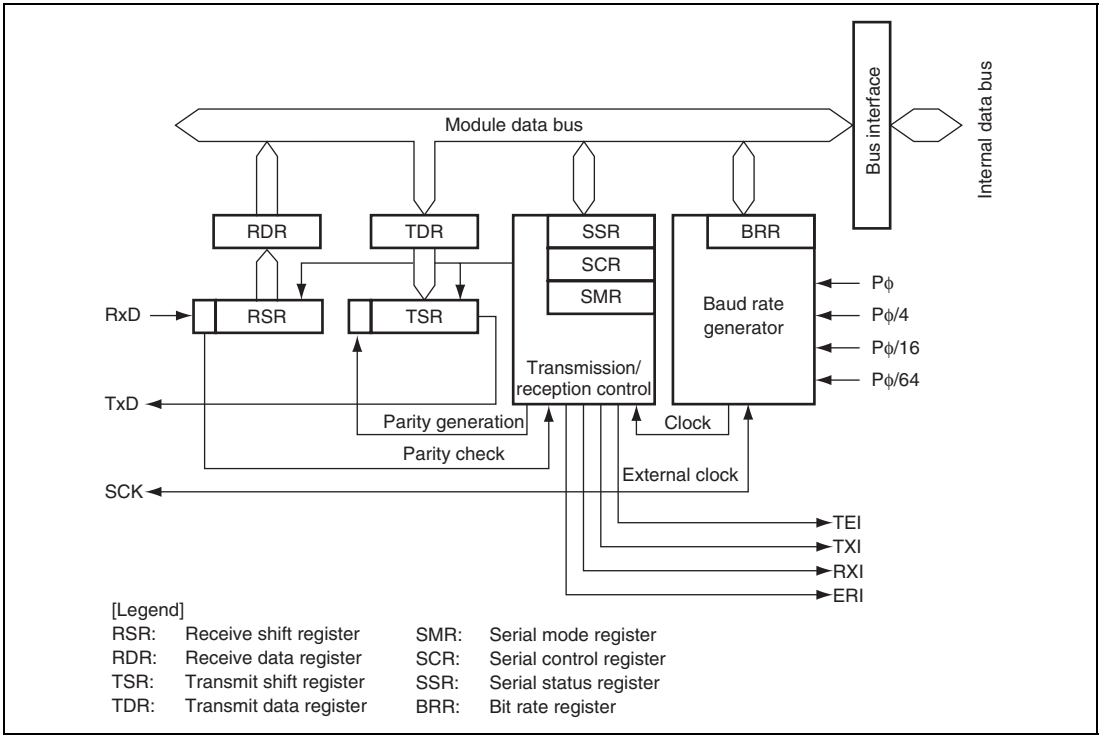


Figure 13.1 Block Diagram of SCI

13.2 Input/Output Pins

Table 13.1 lists the pin configuration of the SCI.

Table 13.1 Pin Configuration

Channel	Pin Name*	I/O	Function
3	SCK3	I/O	Channel 3 clock input/output
	RxD3	Input	Channel 3 receive data input
	TxD3	Output	Channel 3 transmit data output
4	SCK4	I/O	Channel 4 clock input/output
	RxD4	Input	Channel 4 receive data input
	TxD4	Output	Channel 4 transmit data output

Note: * Pin names SCK, RxD, and TxD are used in the text for all channels, omitting the channel designation.

13.3 Register Descriptions

The SCI module has the following registers.

- Channel 3:
  Receive shift register_3 (RSR_3)
  Transmit shift register_3 (TSR_3)
  Receive data register_3 (RDR_3)
  Transmit data register_3 (TDR_3)
  Serial mode register_3 (SMR_3)
  Serial control register_3 (SCR_3)
  Serial status register_3 (SSR_3)
  Bit rate register_3 (BRR_3)

- Channel 4:
  Receive shift register_4 (RSR_4)
  Transmit shift register_4 (TSR_4)
  Receive data register_4 (RDR_4)
  Transmit data register_4 (TDR_4)
  Serial mode register_4 (SMR_4)
  Serial control register_4 (SCR_4)
  Serial status register_4 (SSR_4)
  Bit rate register_4 (BRR_4)

13.3.1 Receive Shift Register (RSR)

RSR is a shift register which is used to receive serial data input from the RxD pin and converts it into parallel data. When one frame of data has been received, it is transferred to RDR automatically. RSR cannot be directly accessed by the CPU.

13.3.2 Receive Data Register (RDR)

RDR is an 8-bit register that stores receive data. When the SCI has received one frame of serial data, it transfers the received serial data from RSR to RDR where it is stored. This allows RSR to receive the next data. Since RSR and RDR function as a double buffer in this way, continuous receive operations can be performed. After confirming that the RDRF bit in SSR is set to 1, read RDR. RDR cannot be written to by the CPU.

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R

13.3.3 Transmit Data Register (TDR)

TDR is an 8-bit register that stores transmit data. When the SCI detects that TSR is empty, it transfers the transmit data written in TDR to TSR and starts transmission. The double-buffered structures of TDR and TSR enable continuous serial transmission. If the next transmit data has already been written to TDR when one frame of data is transmitted, the SCI transfers the written data to TSR to continue transmission. TDR can be read from or written to by the CPU at all times. Write transmit data to TDR only after confirming that the TDRE bit in SSR is set to 1.

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

13.3.4 Transmit Shift Register (TSR)

TSR is a shift register that transmits serial data. To perform serial data transmission, the SCI first automatically transfers transmit data from TDR to TSR, then sends the data to the TxD pin. TSR cannot be directly accessed by the CPU.

13.3.5 Serial Mode Register (SMR)

SMR is used to set the SCI's serial transfer format and select the clock source for the baud rate generator.

Bit	7	6	5	4	3	2	1	0
Bit Name	C/$\overline{A}$	CHR	PE	O/$\overline{E}$	STOP	MP	CKS1	CKS0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*

Note: * These bits can be written to only when TE = RE = 0.

Bit	Bit Name	Initial Value	R/W	Description
7	C/$\overline{A}$	0	R/W*	Communications Mode 0: Asynchronous 1: Clocked synchronous
6	CHR	0	R/W*	Character Length (only valid in asynchronous mode) 0: The data length is 8 bits. 1: The data length is 7 bits. The MSB (bit 7) in TDR is not transmitted in transmission. In clocked synchronous mode, a fixed data length of 8 bits is used.
5	PE	0	R/W*	Parity Enable (only valid in asynchronous mode) When this bit is set to 1, the parity bit is appended to transmit data before transmission, and the parity bit is checked in reception. For a multiprocessor format, parity bit addition and checking are not performed regardless of the PE bit setting.

Bit	Bit Name	Initial Value	R/W	Description
4	O/$\overline{E}$	0	R/W*	Parity Mode (only valid when the PE bit is 1 in asynchronous mode) 0: Selects even parity for transmission/reception. 1: Selects odd parity for transmission/reception. If even parity is selected, the parity bit is determined to make an even number of 1s in the transmitted/received character and the parity bit. Likewise, if odd parity is selected, the parity bit is determined to make an odd number of 1s in the transmitted/received character and the parity bit.
3	STOP	0	R/W*	Stop Bit Length (only valid in asynchronous mode) 0: 1 stop bit in transmission 1: 2 stop bits in transmission In reception, only the first stop bit is checked. If the second stop bit is 0, it is treated as the start bit of the next transmit frame.
2	MP	0	R/W*	Multiprocessor Mode (only valid in asynchronous mode) When this bit is set to 1, the multiprocessor function is enabled. The PE bit and O/$\overline{E}$ bit settings are invalid in multiprocessor mode.
1	CKS1	0	R/W*	Clock Select 1, 0
0	CKS0	0	R/W*	These bits select the clock source for the baud rate generator. 00: Pϕ clock (n = 0) 01: Pϕ/4 clock (n = 1) 10: Pϕ/16 clock (n = 2) 11: Pϕ/64 clock (n = 3) For the relation between the settings of these bits and the baud rate, see section 13.3.8, Bit Rate Register (BRR). n is the decimal notation of the value of n in BRR (see section 13.3.8, Bit Rate Register (BRR)).

Note: * These bits can be written to only when TE = RE = 0.

13.3.6 Serial Control Register (SCR)

SCR is a register that enables/disables the following SCI transfer operations and interrupt requests, and selects the transfer clock source. For details on interrupt requests, see section 13.7, Interrupt Sources.

Bit	7	6	5	4	3	2	1	0
Bit Name	TIE	RIE	TE	RE	MPIE	TEIE	CKE1	CKE0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W*1	R/W*1	R/W	R/W	R/W*2	R/W*2

Bit	Bit Name	Initial Value	R/W	Description
7	TIE	0	R/W	Transmit Interrupt Enable When this bit is set to 1, a TXI interrupt request is enabled. A TXI interrupt request can be cancelled by reading 1 from the TDRE flag and then clearing the flag to 0, or by clearing the TIE bit to 0.
6	RIE	0	R/W	Receive Interrupt Enable When this bit is set to 1, RXI and ERI interrupt requests are enabled. RXI and ERI interrupt requests can be cancelled by reading 1 from the RDRF, FER, PER, or ORER flag and then clearing the flag to 0, or by clearing the RIE bit to 0.
5	TE	0	R/W*1	Transmit Enable When this bit is set to 1, transmission is enabled. Under this condition, serial transmission is started by writing transmit data to TDR, and clearing the TDRE flag in SSR to 0. Note that SMR should be set prior to setting the TE bit to 1 in order to designate the transmission format. If transmission is halted by clearing this bit to 0, the TDRE flag in SSR is fixed 1.

Bit	Bit Name	Initial Value	R/W	Description
4	RE	0	R/W*[1]	Receive Enable When this bit is set to 1, reception is enabled. Under this condition, serial reception is started by detecting the start bit in asynchronous mode or the synchronous clock input in clocked synchronous mode. Note that SMR should be set prior to setting the RE bit to 1 in order to designate the reception format. Even if reception is halted by clearing this bit to 0, the RDRF, FER, PER, and ORER flags are not affected and the previous value is retained.
3	MPIE	0	R/W	Multiprocessor Interrupt Enable (only valid when the MP bit in SMR is 1 in asynchronous mode) When this bit is set to 1, receive data in which the multiprocessor bit is 0 is skipped, and setting of the RDRF, FER, and ORER status flags in SSR is disabled. On receiving data in which the multiprocessor bit is 1, this bit is automatically cleared and normal reception is resumed. For details, see section 13.5, Multiprocessor Communication Function. When receive data including MPB = 0 in SSR is being received, transfer of the received data from RSR to RDR, detection of reception errors, and the settings of RDRF, FER, and ORER flags in SSR are not performed. When receive data including MPB = 1 is received, the MPB bit in SSR is set to 1, the MPIE bit is automatically cleared to 0, and RXI and ERI interrupt requests (in the case where the RIE bit in SCR is set to 1) and setting of the FER and ORER flags are enabled. When the multiprocessor communication function is not to be used, write 0 to this bit.
2	TEIE	0	R/W	Transmit End Interrupt Enable When this bit is set to 1, a TEI interrupt request is enabled. A TEI interrupt request can be cancelled by reading 1 from the TDRE flag and then clearing the flag to 0 in order to clear the TEND flag to 0, or by clearing the TEIE bit to 0.

Bit	Bit Name	Initial Value	R/W	Description
1	CKE1	0	R/W*[2]	Clock Enable 1, 0
0	CKE0	0	R/W*[2]	These bits select the clock source and SCK pin function. • Asynchronous mode 00: On-chip baud rate generator (SCK pin functions as I/O port.) 01: On-chip baud rate generator (Outputs a clock with the same frequency as the bit rate from the SCK pin.) 1X: External clock (Inputs a clock with a frequency 16 times the bit rate from the SCK pin.) • Clocked synchronous mode 0X: Internal clock (SCK pin functions as clock output.) 1X: External clock (SCK pin functions as clock input.)

[Legend] X: Don't care

Notes: 1. Only when TE = RE = 0, 1 can be written to these bits. Once either the TE or RE bit has been set to 1, only writing of TE = RE = 0 is possible.
2. These bits can be written to only when TE = RE = 0.

13.3.7 Serial Status Register (SSR)

SSR is a register containing status flags of the SCI and multiprocessor bits for transfer. TDRE, RDRF, ORER, PER, and FER can only be cleared.

Bit	7	6	5	4	3	2	1	0
Bit Name	TDRE	RDRF	ORER	FER	PER	TEND	MPB	MPBT
Initial Value	1	0	0	0	0	1	0	0
R/W	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R	R	R/W

Note: * Only 0 can be written to clear the flag.

Bit	Bit Name	Initial Value	R/W	Description
7	TDRE	1	R/(W)*	Transmit Data Register Empty Indicates whether TDR contains transmit data. [Setting conditions] • The TE bit in SCR is 0. • Data is transferred from TDR to TSR and TDR has become ready to be written with data. [Clearing conditions] • 0 is written to TDRE after reading TDRE = 1 (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.) • A TXI interrupt request is issued allowing DMAC or DTC to write transmit data to TDR
6	RDRF	0	R/(W)*	Receive Data Register Full Indicates whether receive data is stored in RDR. [Setting condition] • Serial reception ends normally and receive data is transferred from RSR to RDR. [Clearing conditions] • 0 is written to RDRF after reading RDRF = 1. (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.) • An RXI interrupt request is issued allowing DTC or DMAC to read data from RDR. The RDRF flag is not affected and retains its previous value when the RE bit in SCR is cleared to 0. Note that when the next serial reception is completed while the RDRF flag is being set to 1, an overrun error occurs and the received data is lost.

Bit	Bit Name	Initial Value	R/W	Description
5	ORER	0	R/(W)*	Overrun Error Indicates that an overrun error has occurred during reception and the reception ends abnormally. [Setting condition] • The next serial reception is completed while RDRF = 1. In RDR, receive data prior to an overrun error occurrence is retained, but data received after the overrun error occurrence is lost. When the ORER flag is set to 1, subsequent serial reception cannot be performed. Note that in clocked synchronous mode, serial transmission also cannot continue. [Clearing condition] • 0 is written to ORER after reading ORER = 1. (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.) Even when the RE bit in SCR is cleared, the ORER flag is not affected and retains its previous value.
4	FER	0	R/(W)*	Framing Error Indicates that a framing error has occurred during reception in asynchronous mode and the reception ends abnormally. [Setting condition] • The stop bit is 0. In 2-stop-bit mode, only the first stop bit is checked whether it is 1 but the second stop bit is not checked. Note that receive data when the framing error occurs is transferred to RDR, however, the RDRF flag is not set. In addition, when the FER flag is being set to 1, the subsequent serial reception cannot be performed. In clocked synchronous mode, serial transmission also cannot continue. [Clearing condition] • 0 is written to FER after reading FER = 1. (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.) Even when the RE bit in SCR is cleared, the FER flag is not affected and retains its previous value.

Bit	Bit Name	Initial Value	R/W	Description
3	PER	0	R/(W)*	Parity Error Indicates that a parity error has occurred during reception in asynchronous mode and the reception ends abnormally. [Setting condition] • A parity error is detected during reception. Receive data when the parity error occurs is transferred to RDR, however, the RDRF flag is not set. Note that when the PER flag is being set to 1, the subsequent serial reception cannot be performed. In clocked synchronous mode, serial transmission also cannot continue. [Clearing condition] • 0 is written to PER after reading PER = 1. (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.) Even when the RE bit in SCR is cleared, the PER bit is not affected and retains its previous value.
2	TEND	1	R	Transmit End [Setting conditions] • The TE bit in SCR is 0. • TDRE = 1 at transmission of the last bit of a transmit character. [Clearing conditions] • 0 is written to TDRE flag after reading TDRE = 1. • A TXI interrupt request is issued allowing DTC or DMAC to write data to TDR.
1	MPB	0	R	Multiprocessor Bit Stores the multiprocessor bit value in the receive frame. When the RE bit in SCR is cleared to 0, its previous state is retained.
0	MPBT	0	R/W	Multiprocessor Bit Transfer Sets the multiprocessor bit value to be added to the transmit frame.

Note: * Only 0 can be written to clear the flag.

13.3.8 Bit Rate Register (BRR)

BRR is an 8-bit register that adjusts the bit rate. As the SCI performs baud rate generator control independently for each channel, different bit rates can be set for each channel. Table 13.2 shows the relationships between the N setting in BRR and bit rate B for normal asynchronous mode and clocked synchronous mode. The initial value of BRR is H'FF. BRR can always be read from the CPU but can only be written to when TE = RE = 0.

Table 13.2 Relationships between N Setting in BRR and Bit Rate B

Mode	Bit Rate	Error
Asynchronous mode	$B = \frac{\phi \times 10^6}{64 \times 2^{2n-1} \times (N+1)}$	$\text{Error}(\%) = \{\frac{\phi \times 10^6}{B \times 64 \times 2^{2n-1} \times (N+1)} - 1\} \times 100$
Clocked synchronous mode	$B = \frac{\phi \times 10^6}{8 \times 2^{2n-1} \times (N+1)}$	

[Legend]

B: Bit rate (bit/s)

N: BRR setting for baud rate generator ($0 \leq N \leq 255$)

ϕ: Operating frequency (MHz)

n: Determined by the SMR settings shown in the following table.

SMR Setting		
CKS1	**CKS0**	**n**
0	0	0
0	1	1
1	0	2
1	1	3

Table 13.3 shows sample N settings in BRR in normal asynchronous mode. Table 13.4 shows the maximum bit rate specifiable for each operating frequency. Table 13.6 shows sample N settings in BRR in clocked synchronous mode. Tables 13.5 and 13.7 show the maximum bit rates with external clock input.

Table 13.3 Examples of BRR Settings for Various Bit Rates (Asynchronous Mode) (1)

	Operating Frequency φ (MHz)											
	9.8304			10			12			12.288		
Bit Rate (bit/s)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	2	174	–0.26	2	177	–0.25	2	212	0.03	2	217	0.08
150	2	127	0.00	2	129	0.16	2	155	0.16	2	159	0.00
300	1	255	0.00	2	64	0.16	2	77	0.16	2	79	0.00
600	1	127	0.00	1	129	0.16	1	155	0.16	1	159	0.00
1200	0	255	0.00	1	64	0.16	1	77	0.16	1	79	0.00
2400	0	127	0.00	0	129	0.16	0	155	0.16	0	159	0.00
4800	0	63	0.00	0	64	0.16	0	77	0.16	0	79	0.00
9600	0	31	0.00	0	32	–1.36	0	38	0.16	0	39	0.00
19200	0	15	0.00	0	15	1.73	0	19	–2.34	0	19	0.00
31250	0	9	–1.70	0	9	0.00	0	11	0.00	0	11	2.40
38400	0	7	0.00	0	7	1.73	0	9	–2.34	0	9	0.00

	Operating Frequency φ (MHz)											
	14			14.7456			16			17.2032		
Bit Rate (bit/s)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	2	248	–0.17	3	64	0.70	3	70	0.03	3	75	0.48
150	2	181	0.16	2	191	0.00	2	207	0.16	2	223	0.00
300	2	90	0.16	2	95	0.00	2	103	0.16	2	111	0.00
600	1	181	0.16	1	191	0.00	1	207	0.16	1	223	0.00
1200	1	90	0.16	1	95	0.00	1	103	0.16	1	111	0.00
2400	0	181	0.16	0	191	0.00	0	207	0.16	0	223	0.00
4800	0	90	0.16	0	95	0.00	0	103	0.16	0	111	0.00
9600	0	45	–0.93	0	47	0.00	0	51	0.16	0	55	0.00
19200	0	22	–0.93	0	23	0.00	0	25	0.16	0	27	0.00
31250	0	13	0.00	0	14	–1.70	0	15	0.00	0	16	1.20
38400	—	—	—	0	11	0.00	0	12	0.16	0	13	0.00

Table 13.3 Examples of BRR Settings for Various Bit Rates (Asynchronous Mode) (2)

	Operating Frequency ϕ (MHz)											
	18			19.6608			20			25		
Bit Rate (bit/s)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	3	79	–0.12	3	86	0.31	3	88	–0.25	3	110	–0.02
150	2	233	0.16	2	255	0.00	3	64	0.16	3	80	–0.47
300	2	116	0.16	2	127	0.00	2	129	0.16	2	162	0.15
600	1	233	0.16	1	255	0.00	2	64	0.16	2	80	–0.47
1200	1	116	0.16	1	127	0.00	1	129	0.16	1	162	0.15
2400	0	233	0.16	0	255	0.00	1	64	0.16	1	80	–0.47
4800	0	116	0.16	0	127	0.00	0	129	0.16	0	162	0.15
9600	0	58	–0.69	0	63	0.00	0	64	0.16	0	80	–0.47
19200	0	28	1.02	0	31	0.00	0	32	–1.36	0	40	–0.76
31250	0	17	0.00	0	19	–1.70	0	19	0.00	0	24	0.00
38400	0	14	–2.34	0	15	0.00	0	15	1.73	0	19	1.73

	Operating Frequency ϕ (MHz)					
	30			33		
Bit Rate (bit/s)	n	N	Error (%)	n	N	Error (%)
110	3	132	0.13	3	145	0.33
150	3	97	–0.35	3	106	0.39
300	2	194	0.16	2	214	–0.07
600	2	97	–0.35	2	106	0.39
1200	1	194	0.16	1	214	–0.07
2400	1	97	–0.35	1	106	0.39
4800	0	194	0.16	0	214	–0.07
9600	0	97	–0.35	0	106	0.39
19200	0	48	–0.35	0	53	–0.54
31250	0	29	0	0	32	0
38400	0	23	1.73	0	26	–0.54

Table 13.4 Maximum Bit Rate for Each Operating Frequency (Asynchronous Mode)

Pϕ (MHz)	Maximum Bit Rate (bit/s)	n	N	Pϕ (MHz)	Maximum Bit Rate (bit/s)	n	N
8	250000	0	0	17.2032	537600	0	0
9.8304	307200	0	0	18	562500	0	0
10	312500	0	0	19.6608	614400	0	0
12	375000	0	0	20	625000	0	0
12.288	384000	0	0	25	781250	0	0
14	437500	0	0	30	937500	0	0
14.7456	460800	0	0	33	1031250	0	0
16	500000	0	0	35	1093750	0	0

Table 13.5 Maximum Bit Rate with External Clock Input (Asynchronous Mode)

Pϕ (MHz)	External Input Clock (MHz)	Maximum Bit Rate (bit/s)	Pϕ (MHz)	External Input Clock (MHz)	Maximum Bit Rate (bit/s)
8	2.0000	125000	17.2032	4.3008	268800
9.8304	2.4576	153600	18	4.5000	281250
10	2.5000	156250	19.6608	4.9152	307200
12	3.0000	187500	20	5.0000	312500
12.288	3.0720	192000	25	6.2500	390625
14	3.5000	218750	30	7.5000	468750
14.7456	3.6864	230400	33	8.2500	515625
16	4.0000	250000	35	8.7500	546875

Table 13.6 Examples of BRR Settings for Various Bit Rates (Clocked Synchronous Mode)

Bit Rate (bit/s)	Operating Frequency Pϕ (MHz)							
	8		10		16		20	
	n	N	n	N	n	N	n	N
110								
250	3	124	—	—	3	249		
500	2	249	—	—	3	124	—	—
1k	2	124	—	—	2	249	—	—
2.5k	1	199	1	249	2	99	2	124
5k	1	99	1	124	1	199	1	249
10k	0	199	0	249	1	99	1	124
25k	0	79	0	99	0	159	0	199
50k	0	39	0	49	0	79	0	99
100k	0	19	0	24	0	39	0	49
250k	0	7	0	9	0	15	0	19
500k	0	3	0	4	0	7	0	9
1M	0	1			0	3	0	4
2.5M			0	0*			0	1
5M							0	0*

Bit Rate (bit/s)	Operating Frequency Pϕ (MHz)							
	25		30		33		35	
	n	N	n	N	n	N	n	N
110								
250								
500			3	233				
1k	3	97	3	116	3	128	3	136
2.5k	2	155	2	187	2	205	2	218
5k	2	77	2	93	2	102	2	108
10k	1	155	1	187	1	205	1	218
25k	0	249	1	74	1	82	1	87
50k	0	124	0	149	0	164	0	174
100k	0	62	0	74	0	82	0	87
250k	0	24	0	29	0	32	0	34
500k	—	—	0	14	—	—	—	—
1M	—	—	—	—	—	—	—	—
2.5M	—	—	0	2	—	—	—	—
5M	—	—	—	—	—	—	—	—

[Legend]

Blank space : Setting prohibited.

— : Can be set, but there will be error.

* : Continuous transmission or reception is not possible.

Table 13.7 Maximum Bit Rate with External Clock Input (Clocked Synchronous Mode)

Pϕ (MHz)	External Input Clock (MHz)	Maximum Bit Rate (bit/s)	Pϕ (MHz)	External Input Clock (MHz)	Maximum Bit Rate (bit/s)
8	1.3333	1333333.3	20	3.3333	3333333.3
10	1.6667	1666666.7	25	4.1667	4166666.7
12	2.0000	2000000.0	30	5.0000	5000000.0
14	2.3333	2333333.3	33	5.5000	5500000.0
16	2.6667	2666666.7	35	5.8336	5833625.0
18	3.0000	3000000.0			

13.4 Operation in Asynchronous Mode

Figure 13.2 shows the general format for asynchronous serial communication. One frame consists of a start bit (low level), followed by transmit/receive data, a parity bit, and finally stop bits (high level). In asynchronous serial communication, the communication line is usually held in the mark state (high level). The SCI monitors the communication line, and when it goes to the space state (low level), recognizes it as a start bit and starts serial communication. Inside the SCI, the transmitter and receiver are independent units, enabling full-duplex communication. Both the transmitter and the receiver also have a double-buffered structure, so that data can be read or written during transmission or reception, enabling continuous data transmission and reception.

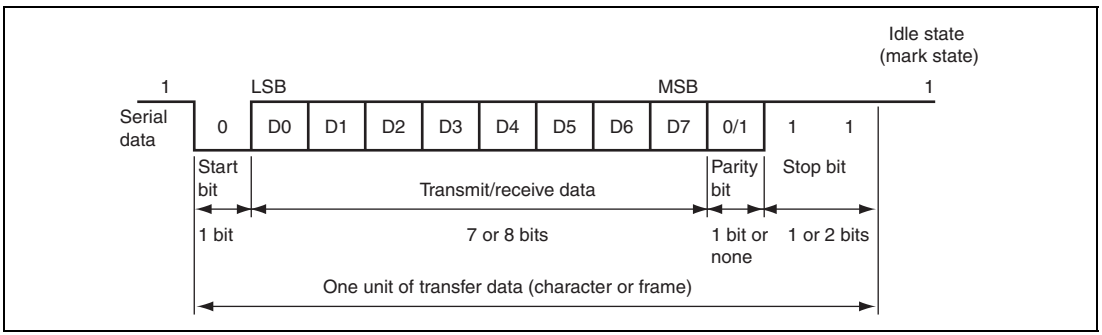


Figure 13.2 Data Format in Asynchronous Communication (Example with 8-Bit Data, Parity, Two Stop Bits)

13.4.1 Data Transfer Format

Table 13.8 shows the data transfer formats that can be used in asynchronous mode. Any of 12 transfer formats can be selected according to the SMR setting. For details on the multiprocessor bit, see section 13.5, Multiprocessor Communication Function.

Table 13.8 Serial Transfer Formats (Asynchronous Mode)

SMR Settings				Serial Transmit/Receive Format and Frame Length
CHR	PE	MP	STOP	1 2 3 4 5 6 7 8 9 10 11 12
0	0	0	0	S 8-bit data STOP
0	0	0	1	S 8-bit data STOP STOP
0	1	0	0	S 8-bit data P STOP
0	1	0	1	S 8-bit data P STOP STOP
1	0	0	0	S 7-bit data STOP
1	0	0	1	S 7-bit data STOP STOP
1	1	0	0	S 7-bit data P STOP
1	1	0	1	S 7-bit data P STOP STOP
0	—	1	0	S 8-bit data MPB STOP
0	—	1	1	S 8-bit data MPB STOP STOP
1	—	1	0	S 7-bit data MPB STOP
1	—	1	1	S 7-bit data MPB STOP STOP

[Legend]
S: Start bit
STOP: Stop bit
P: Parity bit
MPB: Multiprocessor bit

13.4.2 Receive Data Sampling Timing and Reception Margin in Asynchronous Mode

In asynchronous mode, the SCI operates on a basic clock with a frequency of 16 times the bit rate. In reception, the SCI samples the falling edge of the start bit using the basic clock, and performs internal synchronization. Since receive data is sampled at the rising edge of the 8th pulse of the basic clock, data is latched at the middle of each bit, as shown in figure 13.3. Thus the reception margin in asynchronous mode is determined by formula (1) below.

$$M = \{ (0.5 - \frac{1}{2N}) - (L - 0.5) F - \frac{| D - 0.5 |}{N} (1 + F) \} \times 100 \quad [\%] \quad \cdots \quad \text{Formula (1)}$$

M: Reception margin
N: Ratio of bit rate to clock (N = 16)
D: Duty cycle of clock (D = 0.5 to 1.0)
L: Frame length (L = 9 to 12)
F: Absolute value of clock frequency deviation

Assuming values of F = 0 and D = 0.5 in formula (1), the reception margin is determined by the formula below.

$$M = \{ 0.5 - \frac{1}{2 \times 16} \} \times 100 \quad [\%] = 46.875\%$$

However, this is only the computed value, and a margin of 20% to 30% should be allowed in system design.

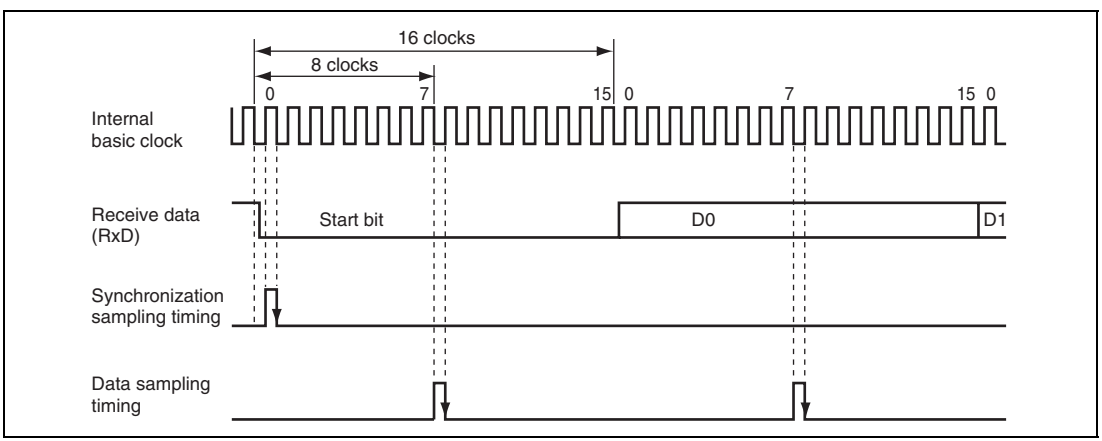


Figure 13.3 Receive Data Sampling Timing in Asynchronous Mode

13.4.3 Clock

Either an internal clock generated by the on-chip baud rate generator or an external clock input to the SCK pin can be selected as the SCI's transfer clock, according to the setting of the C/A bit in SMR and the CKE1 and CKE0 bits in SCR. When an external clock is input to the SCK pin, the clock frequency should be 16 times the bit rate used.

When the SCI is operated on an internal clock, the clock can be output from the SCK pin. The frequency of the clock output in this case is equal to the bit rate, and the phase is such that the rising edge of the clock is in the middle of the transmit data, as shown in figure 13.4.

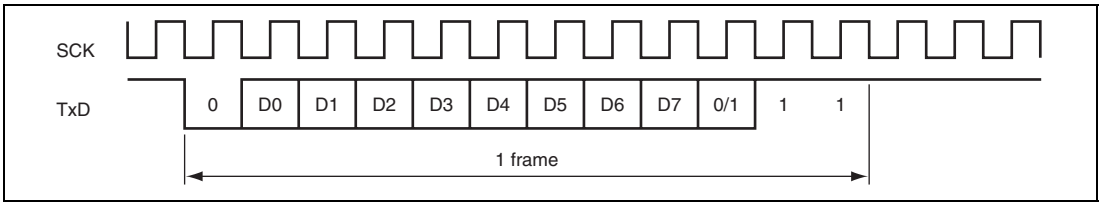
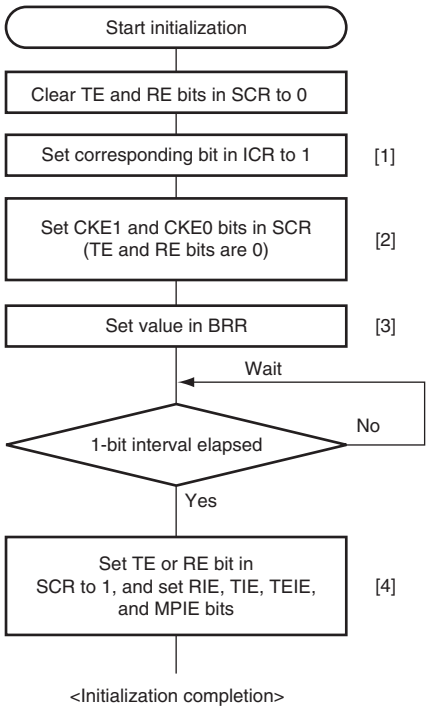


Figure 13.4 Phase Relation between Output Clock and Transmit Data (Asynchronous Mode)

13.4.4 SCI Initialization (Asynchronous Mode)

Before transmitting and receiving data, first clear the TE and RE bits in SCR to 0, then initialize the SCI as described in a sample flowchart in figure 13.5. When the operating mode, transfer format, etc., are changed, both of the TE and RE bits must be cleared to 0 before making the change. When the TE bit is cleared to 0, the TDRE flag is set to 1. Note that clearing the RE bit to 0 does not initialize the RDRF, PER, FER, and ORER flags, or RDR. When the external clock is used in asynchronous mode, the clock must be supplied even during initialization.



[1] Set the bit in ICR for the corresponding pin when receiving data or using an external clock.

[2] Set the clock selection in SCR. Be sure to clear bits RIE, TIE, TEIE, and MPIE, and bits TE and RE, to 0.

When the clock output is selected in asynchronous mode, the clock is output immediately after SCR settings are made.

[3] Write a value corresponding to the bit rate to BRR. This step is not necessary if an external clock is used.

[4] Wait at least one bit interval, then set the TE bit or RE bit in SCR to 1. Also set the RIE, TIE, TEIE, and MPIE bits. Setting the TE and RE bits enables the TxD and RxD pins to be used.

Figure 13.5 Sample SCI Initialization Flowchart

13.4.5 Serial Data Transmission (Asynchronous Mode)

Figure 13.6 shows an example of the operation for transmission in asynchronous mode. In transmission, the SCI operates as described below.

1. The SCI monitors the TDRE flag in SSR, and if it is cleared to 0, recognizes that data has been written to TDR, and transfers the data from TDR to TSR.
2. After transferring data from TDR to TSR, the SCI sets the TDRE flag to 1 and starts transmission. If the TIE bit in SCR is set to 1 at this time, a TXI interrupt request is generated. Because the TXI interrupt processing routine writes the next transmit data to TDR before transmission of the current transmit data has finished, continuous transmission is enabled.
3. Data is sent from the TxD pin in the following order: start bit, transmit data, parity bit or multiprocessor bit (may be omitted depending on the format), and stop bit.
4. The SCI checks the TDRE flag at the timing for sending the stop bit.
5. If the TDRE flag is 0, the next transmit data is transferred from TDR to TSR, the stop bit is sent, and then serial transmission of the next frame is started.
6. If the TDRE flag is 1, the TEND flag in SSR is set to 1, the stop bit is sent, and then the mark state is entered in which 1 is output. If the TEIE bit in SCR is set to 1 at this time, a TEI interrupt request is generated.

Figure 13.7 shows a sample flowchart for transmission in asynchronous mode.

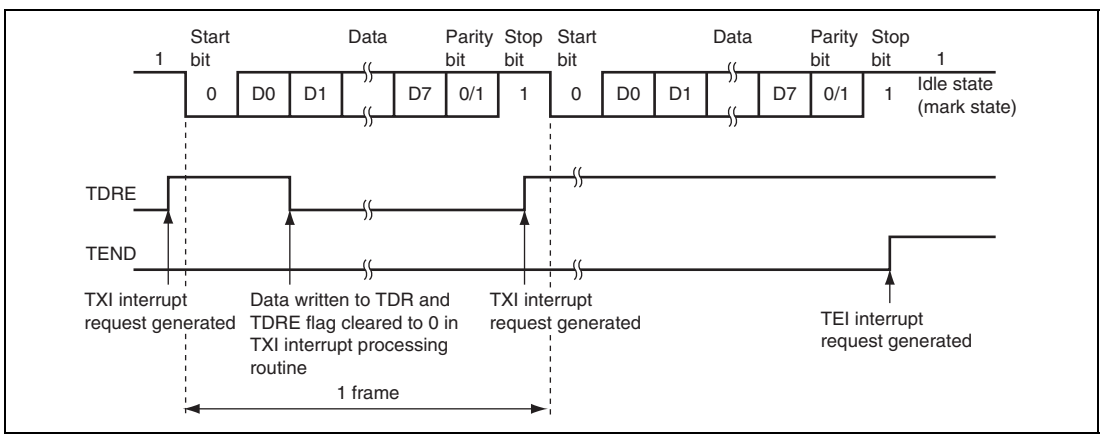
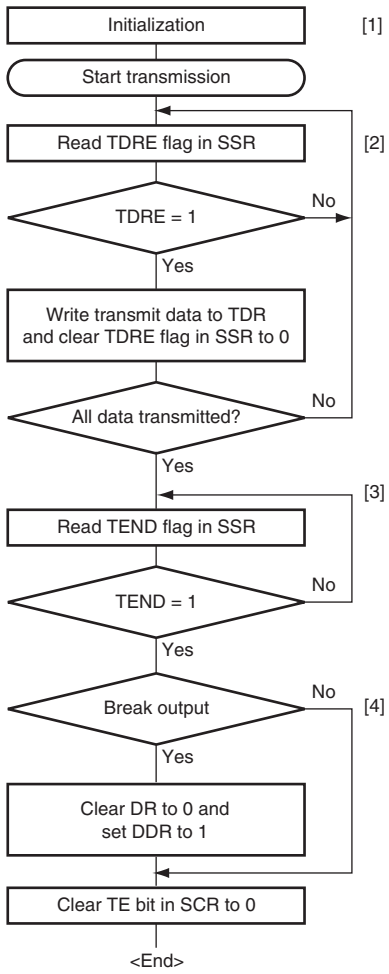


Figure 13.6 Example of Operation for Transmission in Asynchronous Mode (Example with 8-Bit Data, Parity, One Stop Bit)



[1] SCI initialization:
The TxD pin is automatically designated as the transmit data output pin. After the TE bit is set to 1, a 1 is output for a frame, and transmission is enabled.

[2] SCI state check and transmit data write:
Read SSR and check that the TDRE flag is set to 1, then write transmit data to TDR and clear the TDRE flag to 0.

[3] Serial transmission continuation procedure:

To continue serial transmission, read 1 from the TDRE flag to confirm that writing is possible, then write data to TDR, and clear the TDRE flag to 0. However, the TDRE flag is checked and cleared automatically when the DMAC or DTC is initiated by a transmit data empty interrupt (TXI) request and writes data to TDR.

[4] Break output at the end of serial transmission:
To output a break in serial transmission, clear DR corresponding to the TxD pin to 0, set DDR to 1, then clear the TE bit in SCR to 0.

Figure 13.7 Sample Serial Transmission Flowchart

13.4.6 Serial Data Reception (Asynchronous Mode)

Figure 13.8 shows an example of the operation for reception in asynchronous mode. In serial reception, the SCI operates as described below.

1. The SCI monitors the communication line, and if a start bit is detected, performs internal synchronization, stores receive data in RSR, and checks the parity bit and stop bit.
2. If an overrun error (when reception of the next data is completed while the RDRF flag in SSR is still set to 1) occurs, the ORER bit in SSR is set to 1. If the RIE bit in SCR is set to 1 at this time, an ERI interrupt request is generated. Receive data is not transferred to RDR. The RDRF flag remains to be set to 1.
3. If a parity error is detected, the PER bit in SSR is set to 1 and receive data is transferred to RDR. If the RIE bit in SCR is set to 1 at this time, an ERI interrupt request is generated.
4. If a framing error (when the stop bit is 0) is detected, the FER bit in SSR is set to 1 and receive data is transferred to RDR. If the RIE bit in SCR is set to 1 at this time, an ERI interrupt request is generated.
5. If reception finishes successfully, the RDRF bit in SSR is set to 1, and receive data is transferred to RDR. If the RIE bit in SCR is set to 1 at this time, an RXI interrupt request is generated. Because the RXI interrupt processing routine reads the receive data transferred to RDR before reception of the next receive data has finished, continuous reception can be enabled.

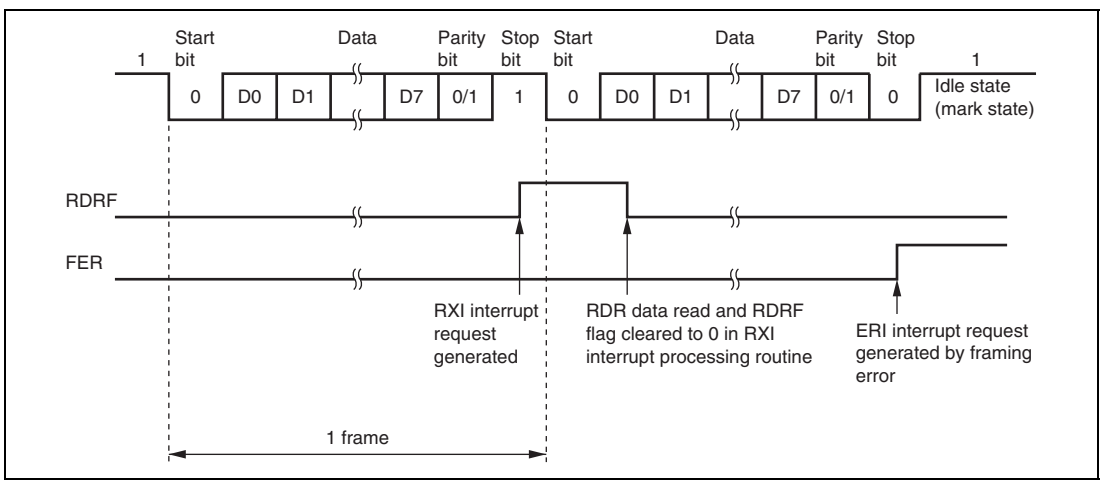


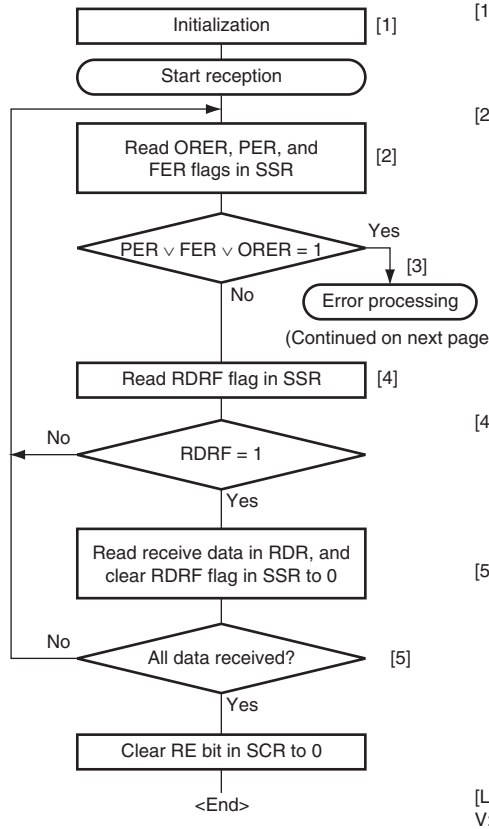
Figure 13.8 Example of SCI Operation for Reception (Example with 8-Bit Data, Parity, One Stop Bit)

Table 13.9 shows the states of the SSR status flags and receive data handling when a receive error is detected. If a receive error is detected, the RDRF flag retains its state before receiving data. Reception cannot be resumed while a receive error flag is set to 1. Accordingly, clear the ORER, FER, PER, and RDRF bits to 0 before resuming reception. Figure 13.9 shows a sample flowchart for serial data reception.

Table 13.9 SSR Status Flags and Receive Data Handling

SSR Status Flag					
RDRF*	**ORER**	**FER**	**PER**	**Receive Data**	**Receive Error Type**
1	1	0	0	Lost	Overrun error
0	0	1	0	Transferred to RDR	Framing error
0	0	0	1	Transferred to RDR	Parity error
1	1	1	0	Lost	Overrun error + framing error
1	1	0	1	Lost	Overrun error + parity error
0	0	1	1	Transferred to RDR	Framing error + parity error
1	1	1	1	Lost	Overrun error + framing error + parity error

Note: * The RDRF flag retains the state it had before data reception.



[1] SCI initialization:
The RxD pin is automatically designated as the receive data input pin.

[2] [3] Receive error processing and break detection:
If a receive error occurs, read the ORER, PER, and FER flags in SSR to identify the error. After performing the appropriate error processing, ensure that the ORER, PER, and FER flags are all cleared to 0. Reception cannot be resumed if any of these flags are set to 1. In the case of a framing error, a break can be detected by reading the value of the input port corresponding to the RxD pin.

[4] SCI state check and receive data read:
Read SSR and check that RDRF = 1, then read the receive data in RDR and clear the RDRF flag to 0. Transition of the RDRF flag from 0 to 1 can also be identified by an RXI interrupt.

[5] Serial reception continuation procedure:
To continue serial reception, before the stop bit for the current frame is received, read the RDRF flag and RDR, and clear the RDRF flag to 0. However, the RDRF flag is cleared automatically when the DMAC or DTC is initiated by an RXI interrupt and reads data from RDR.

[Legend]
V: Logical OR

Figure 13.9 Sample Serial Reception Flowchart (1)

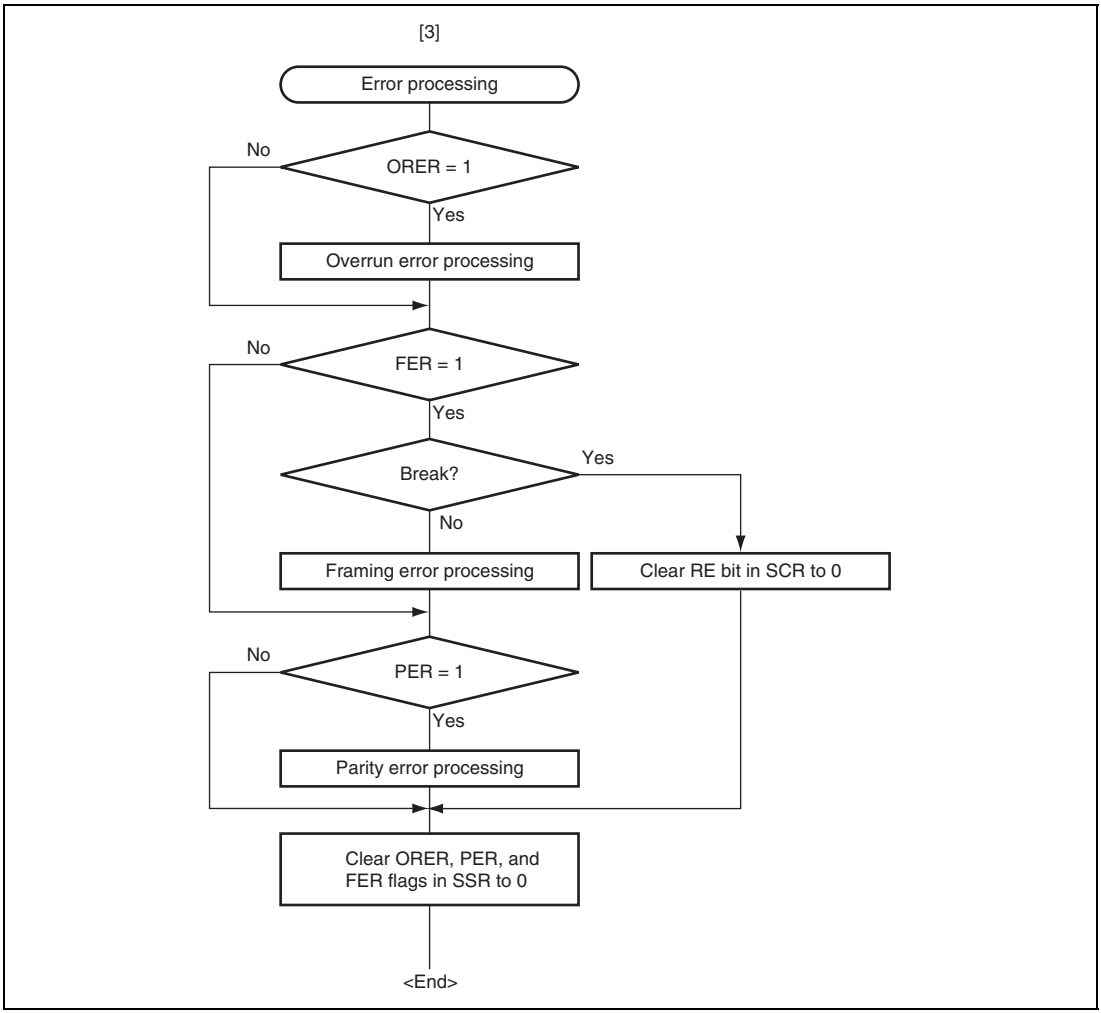


Figure 13.9 Sample Serial Reception Flowchart (2)

13.5 Multiprocessor Communication Function

Use of the multiprocessor communication function enables data transfer to be performed among a number of processors sharing communication lines by means of asynchronous serial communication using the multiprocessor format, in which a multiprocessor bit is added to the transfer data. When multiprocessor communication is carried out, each receiving station is addressed by a unique ID code. The serial communication cycle consists of two component cycles: an ID transmission cycle which specifies the receiving station, and a data transmission cycle for the specified receiving station. The multiprocessor bit is used to differentiate between the ID transmission cycle and the data transmission cycle. If the multiprocessor bit is 1, the cycle is an ID transmission cycle, and if the multiprocessor bit is 0, the cycle is a data transmission cycle. Figure 13.10 shows an example of inter-processor communication using the multiprocessor format. The transmitting station first sends data which includes the ID code of the receiving station and a multiprocessor bit set to 1. It then transmits transmit data added with the multiprocessor bit cleared to 0. The receiving station skips data until data with the multiprocessor bit set to 1 is sent. When data with the multiprocessor bit set to 1 is received, the receiving station compares that data with its own ID. The station whose ID matches then receives the data sent next. Stations whose ID does not match continue to skip data until data with the multiprocessor bit set to 1 is again received.

The SCI uses the MPIE bit in SCR to implement this function. When the MPIE bit is set to 1, transfer of receive data from RSR to RDR, error flag detection, and setting the SSR status flags, RDRF, FER, and ORER in SSR to 1 are prohibited until data with the multiprocessor bit set to 1 is received. On reception of a receive character with the multiprocessor bit set to 1, the MPB bit in SSR is set to 1 and the MPIE bit is automatically cleared, thus normal reception is resumed. If the RIE bit in SCR is set to 1 at this time, an RXI interrupt is generated. If the MPIE bit is cleared to 0, reception is performed regardless of the value of the multiprocessor bit. The multiprocessor bit is stored in the MPB bit in SSR.

When the multiprocessor format is selected, the parity bit setting is invalid. All other bit settings are the same as those in normal asynchronous mode. The clock used for multiprocessor communication is the same as that in normal asynchronous mode.

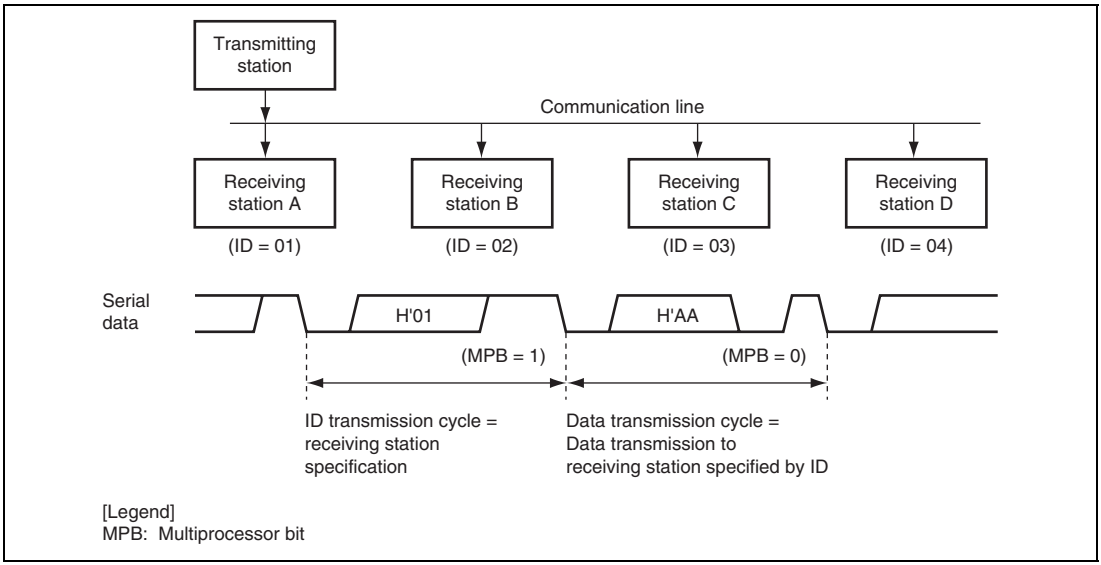


Figure 13.10 Example of Communication Using Multiprocessor Format (Transmission of Data H'AA to Receiving Station A)

13.5.1 Multiprocessor Serial Data Transmission

Figure 13.11 shows a sample flowchart for multiprocessor serial data transmission. For an ID transmission cycle, set the MPBT bit in SSR to 1 before transmission. For a data transmission cycle, clear the MPBT bit in SSR to 0 before transmission. All other SCI operations are the same as those in asynchronous mode.

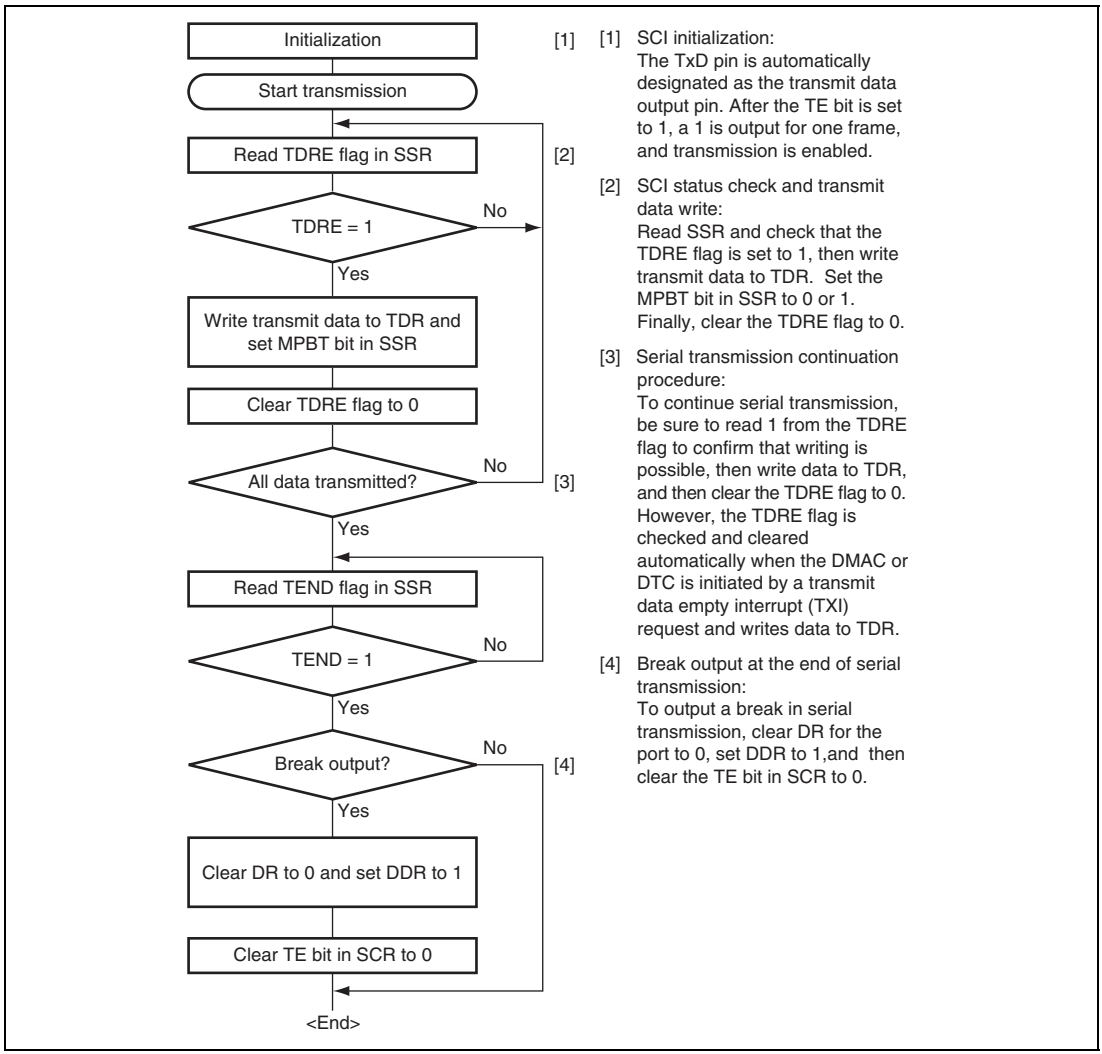


Figure 13.11 Sample Multiprocessor Serial Transmission Flowchart

13.5.2 Multiprocessor Serial Data Reception

Figure 13.13 shows a sample flowchart for multiprocessor serial data reception. If the MPIE bit in SCR is set to 1, data is skipped until data with a 1 multiprocessor bit is sent. On receiving data with a 1 multiprocessor bit, the receive data is transferred to RDR. An RXI interrupt request is generated at this time. All other SCI operations are the same as in asynchronous mode. Figure 13.12 shows an example of SCI operation for multiprocessor format reception.

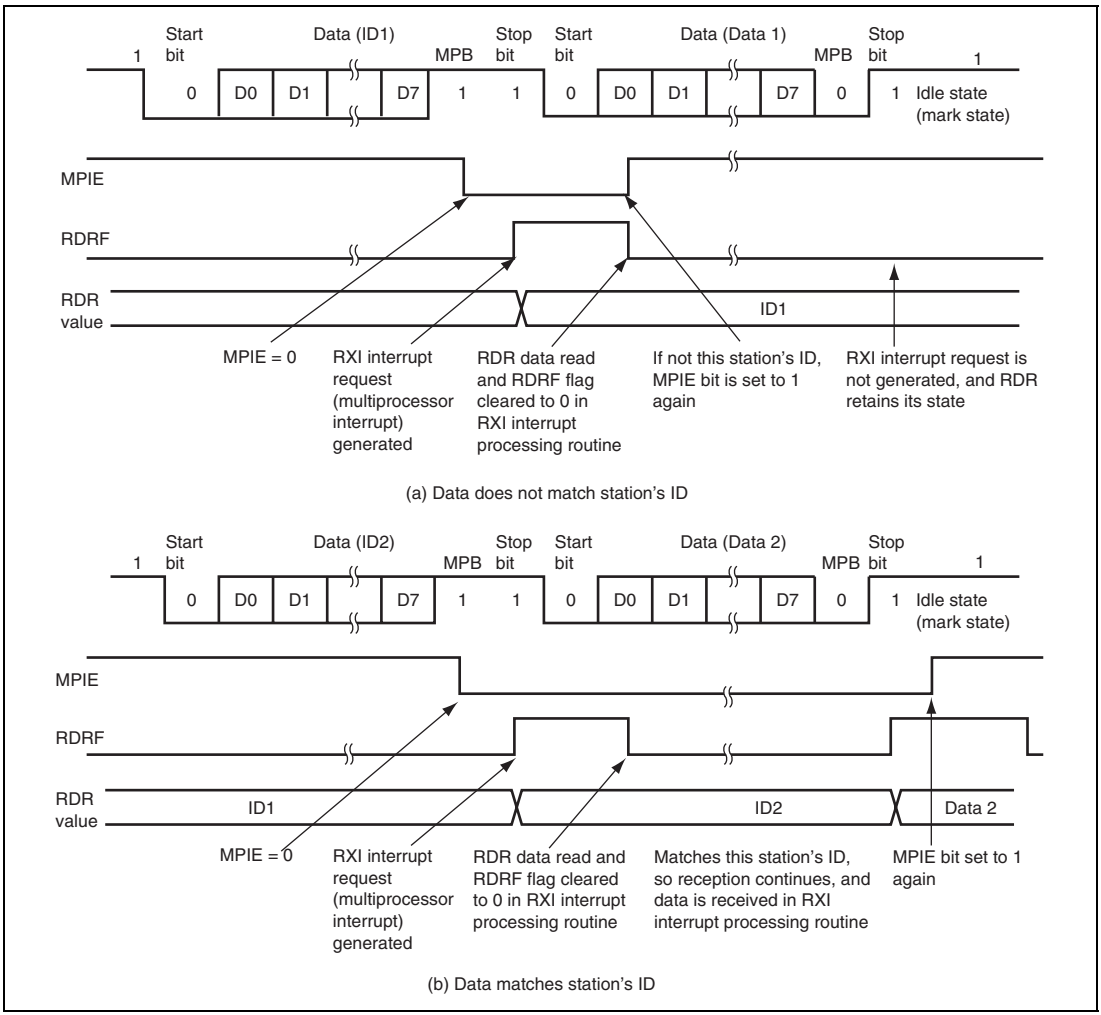


Figure 13.12 Example of SCI Operation for Reception (Example with 8-Bit Data, Multiprocessor Bit, One Stop Bit)

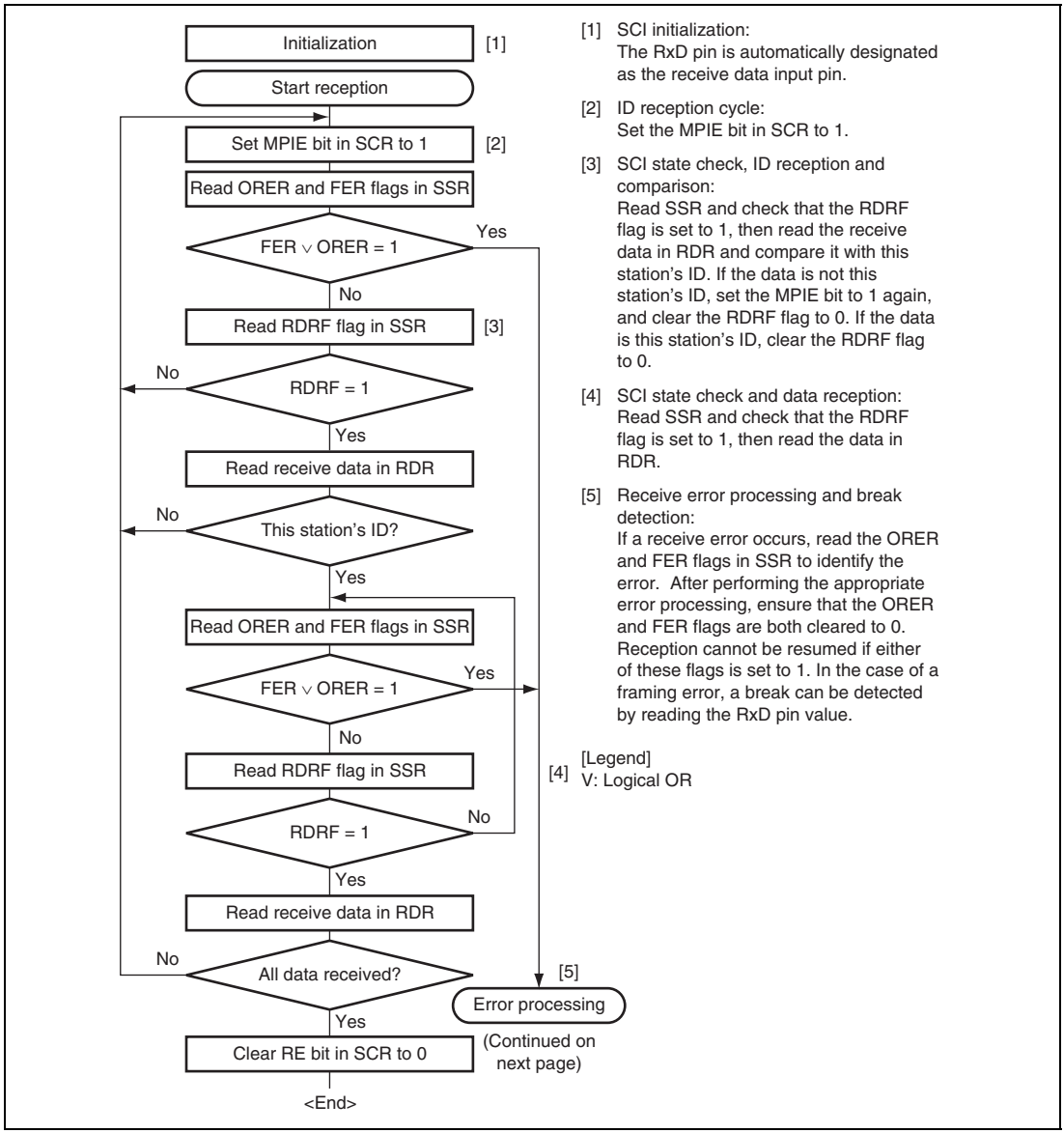


Figure 13.13 Sample Multiprocessor Serial Reception Flowchart (1)

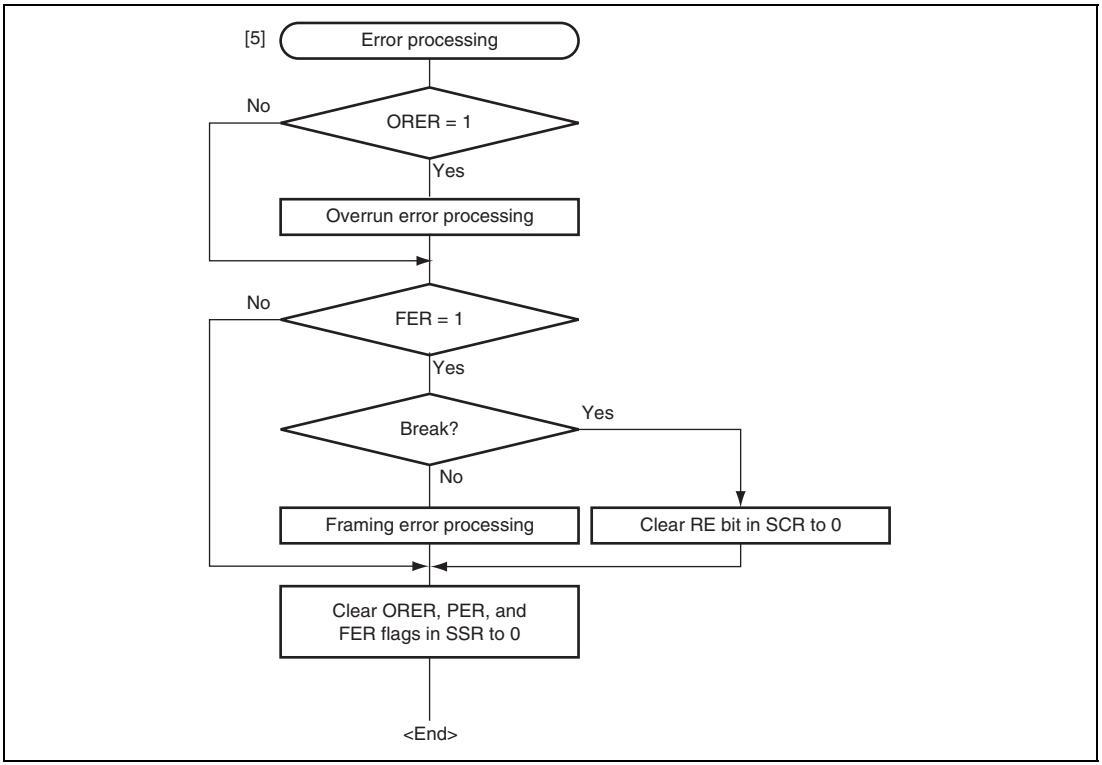


Figure 13.13 Sample Multiprocessor Serial Reception Flowchart (2)

13.6 Operation in Clocked Synchronous Mode

Figure 13.14 shows the general format for clocked synchronous communication. In clocked synchronous mode, data is transmitted or received in synchronization with clock pulses. One character in transfer data consists of 8-bit data. In data transmission, the SCI outputs data from one falling edge of the synchronization clock to the next. In data reception, the SCI receives data in synchronization with the rising edge of the synchronization clock. After 8-bit data is output, the transmission line holds the MSB output state. In clocked synchronous mode, no parity bit or multiprocessor bit is added. Inside the SCI, the transmitter and receiver are independent units, enabling full-duplex communication by use of a common clock. Both the transmitter and the receiver also have a double-buffered structure, so that the next transmit data can be written during transmission or the previous receive data can be read during reception, enabling continuous data transfer.

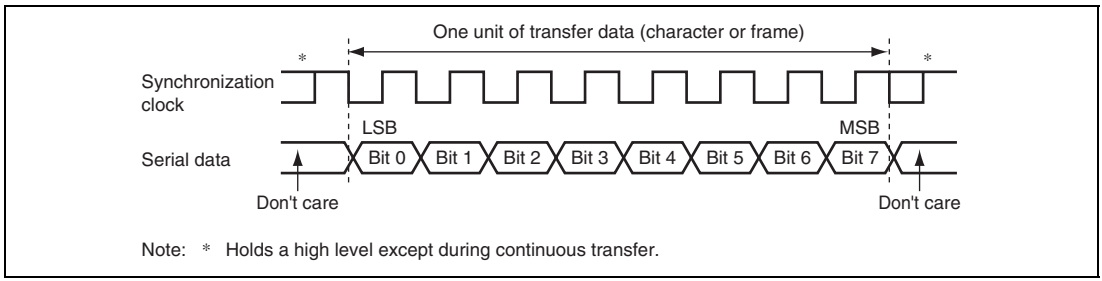


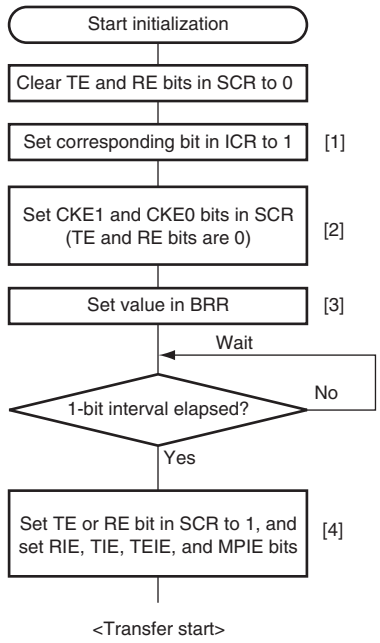
Figure 13.14 Data Format in Clocked Synchronous Communication (LSB-First)

13.6.1 Clock

Either an internal clock generated by the on-chip baud rate generator or an external synchronization clock input at the SCK pin can be selected, according to the setting of the CKE1 and CKE0 bits in SCR. When the SCI is operated on an internal clock, the synchronization clock is output from the SCK pin. Eight synchronization clock pulses are output in the transfer of one character, and when no transfer is performed, the clock is fixed high.

13.6.2 SCI Initialization (Clocked Synchronous Mode)

Before transmitting and receiving data, first clear both of the TE and RE bits in SCR to 0, then initialize the SCI as described in a sample flowchart in figure 13.15. When the operating mode, transfer format, etc., are changed, both of the TE and RE bits must be cleared to 0 before making the change. When the TE bit is cleared to 0, the TDRE flag is set to 1. However, clearing the RE bit to 0 does not initialize the RDRF, PER, FER, and ORER flags, or RDR.



[1] Set the bit in ICR for the corresponding pin when receiving data or using an external clock.

[2] Set the clock selection in SCR. Be sure to clear bits RIE, TIE, TEIE, and MPIE, and bits TE and RE, to 0.

[3] Write a value corresponding to the bit rate to BRR. This step is not necessary if an external clock is used.

[4] Wait at least one bit interval, then set the TE bit or RE bit in SCR to 1. Also set the RIE, TIE TEIE, and MPIE bits. Setting the TE and RE bits enables the TxD and RxD pins to be used.

Note: In simultaneous transmit and receive operations, the TE and RE bits should both be cleared to 0 or set to 1 simultaneously.

Figure 13.15 Sample SCI Initialization Flowchart

13.6.3 Serial Data Transmission (Clocked Synchronous Mode)

Figure 13.16 shows an example of the operation for transmission in clocked synchronous mode. In transmission, the SCI operates as described below.

1. The SCI monitors the TDRE flag in SSR, and if it is 0, recognizes that data has been written to TDR, and transfers the data from TDR to TSR.
2. After transferring data from TDR to TSR, the SCI sets the TDRE flag to 1 and starts transmission. If the TIE bit in SCR is set to 1 at this time, a TXI interrupt request is generated. Because the TXI interrupt processing routine writes the next transmit data to TDR before transmission of the current transmit data has finished, continuous transmission is enabled.
3. 8-bit data is sent from the TxD pin synchronized with the output clock when clock output mode has been specified and synchronized with the input clock when use of an external clock has been specified.
4. The SCI checks the TDRE flag at the timing for sending the last bit.
5. If the TDRE flag is cleared to 0, the next transmit data is transferred from TDR to TSR, and serial transmission of the next frame is started.
6. If the TDRE flag is set to 1, the TEND flag in SSR is set to 1, and the TxD pin retains the output state of the last bit. If the TEIE bit in SCR is set to 1 at this time, a TEI interrupt request is generated. The SCK pin is fixed high.

Figure 13.17 shows a sample flowchart for serial data transmission. Even if the TDRE flag is cleared to 0, transmission will not start while a receive error flag (ORER, FER, or PER) is set to 1. Make sure to clear the receive error flags to 0 before starting transmission. Note that clearing the RE bit to 0 does not clear the receive error flags.

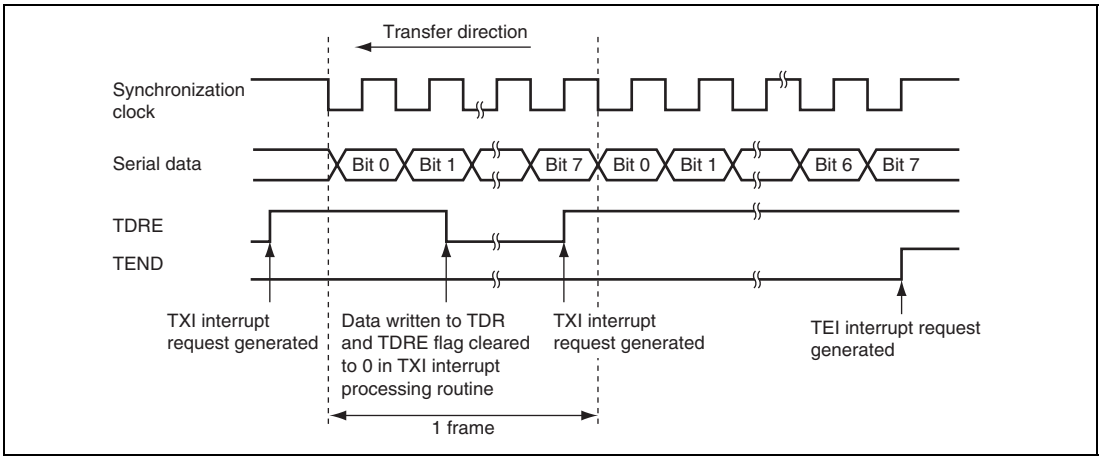


Figure 13.16 Example of Operation for Transmission in Clocked Synchronous Mode

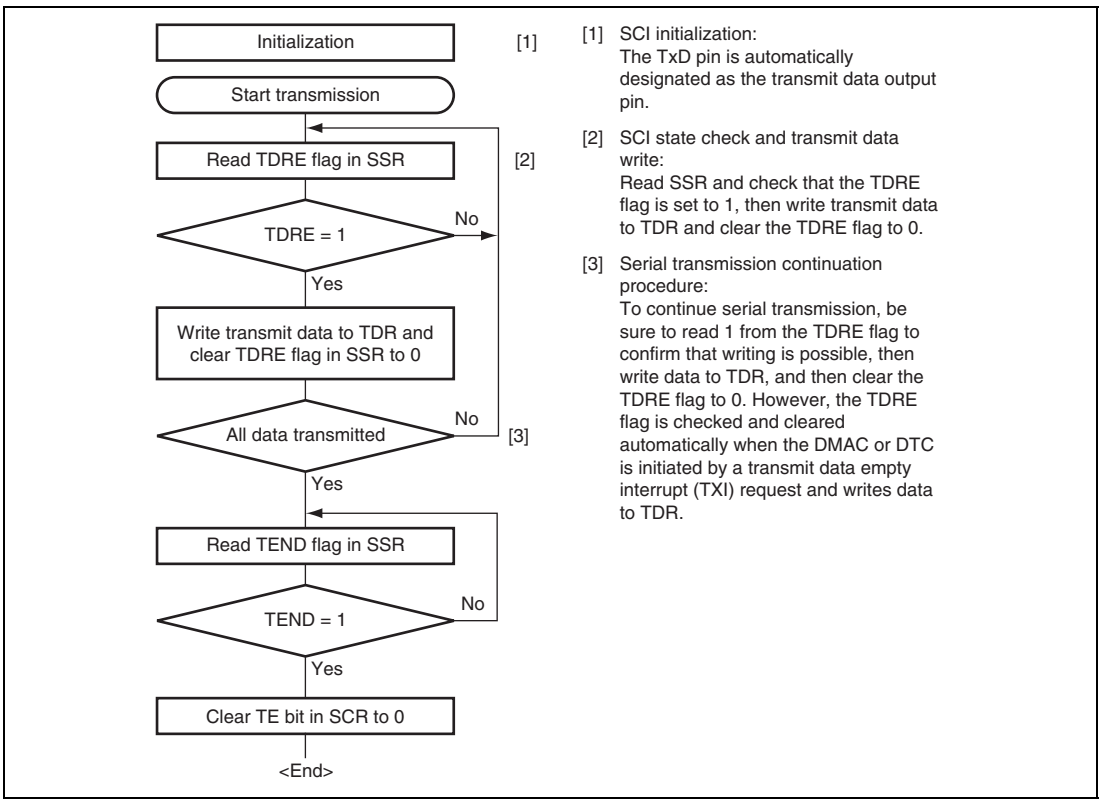


Figure 13.17 Sample Serial Transmission Flowchart

13.6.4 Serial Data Reception (Clocked Synchronous Mode)

Figure 13.18 shows an example of SCI operation for reception in clocked synchronous mode. In serial reception, the SCI operates as described below.

1. The SCI performs internal initialization in synchronization with a synchronization clock input or output, starts receiving data, and stores the receive data in RSR.
2. If an overrun error (when reception of the next data is completed while the RDRF flag in SSR is still set to 1) occurs, the ORER bit in SSR is set to 1. If the RIE bit in SCR is set to 1 at this time, an ERI interrupt request is generated. Receive data is not transferred to RDR. The RDRF flag remains to be set to 1.
3. If reception finishes successfully, the RDRF bit in SSR is set to 1, and receive data is transferred to RDR. If the RIE bit in SCR is set to 1 at this time, an RXI interrupt request is generated. Because the RXI interrupt processing routine reads the receive data transferred to RDR before reception of the next receive data has finished, continuous reception is enabled.

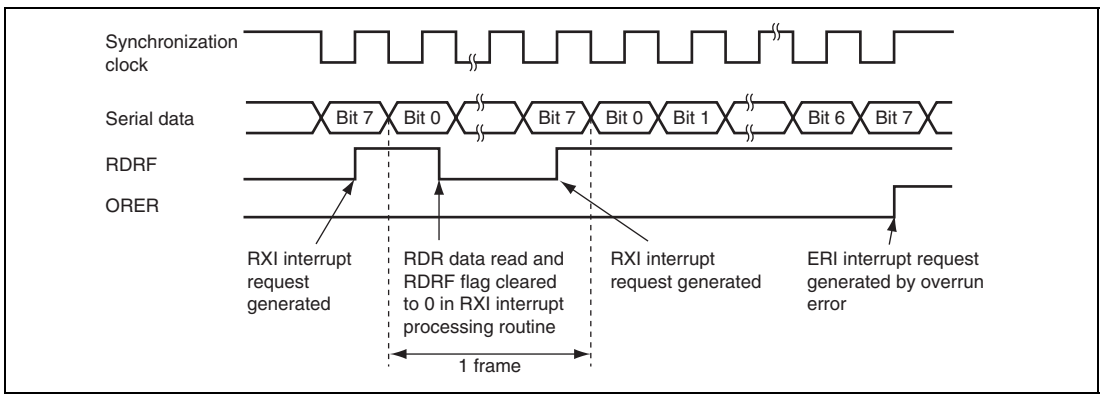


Figure 13.18 Example of Operation for Reception in Clocked Synchronous Mode

Transfer cannot be resumed while a receive error flag is set to 1. Accordingly, clear the ORER, FER, PER, and RDRF bits to 0 before resuming reception. Figure 13.19 shows a sample flowchart for serial data reception.

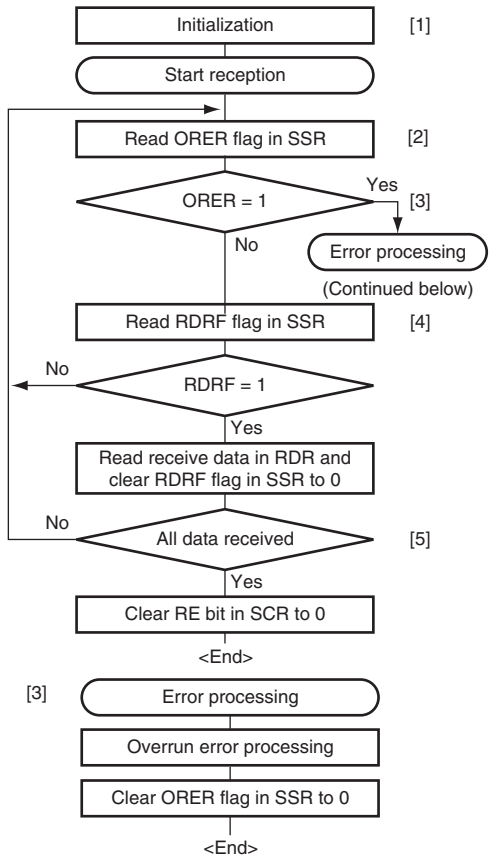


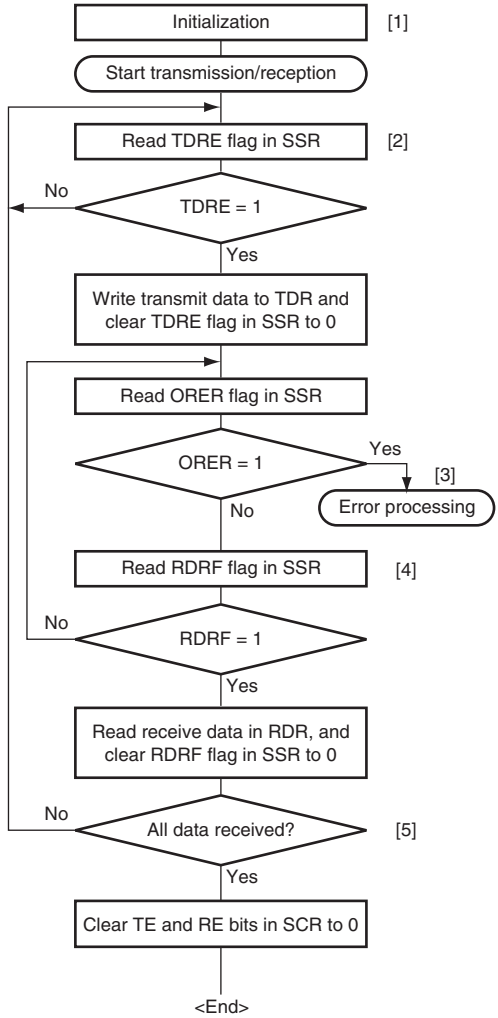
Figure 13.19 Sample Serial Reception Flowchart

13.6.5 Simultaneous Serial Data Transmission and Reception (Clocked Synchronous Mode)

Figure 13.20 shows a sample flowchart for simultaneous serial transmit and receive operations. After initializing the SCI, the following procedure should be used for simultaneous serial data transmit and receive operations.

To switch from transmit mode to simultaneous transmit and receive mode, after checking that the SCI has finished transmission and the TDRE and TEND flags are set to 1, clear the TE bit to 0. Then simultaneously set both the TE and RE bits to 1 with a single instruction.

To switch from receive mode to simultaneous transmit and receive mode, after checking that the SCI has finished reception, clear the RE bit to 0. Then after checking that the RDRF bit and receive error flags (ORER, FER, and PER) are cleared to 0, simultaneously set both the TE and RE bits to 1 with a single instruction.



Note: When switching from transmit or receive operation to simultaneous transmit and receive operations, first clear the TE bit and RE bit to 0, then set both these bits to 1 simultaneously.

[1] SCI initialization:
The TxD pin is designated as the transmit data output pin, and the RxD pin is designated as the receive data input pin, enabling simultaneous transmit and receive operations.

[2] SCI state check and transmit data write:
Read SSR and check that the TDRE flag is set to 1, then write transmit data to TDR and clear the TDRE flag to 0. Transition of the TDRE flag from 0 to 1 can also be identified by a TXI interrupt.

[3] Receive error processing:
If a receive error occurs, read the ORER flag in SSR, and after performing the appropriate error processing, clear the ORER flag to 0. Reception cannot be resumed if the ORER flag is set to 1.

[4] SCI state check and receive data read:
Read SSR and check that the RDRF flag is set to 1, then read the receive data in RDR and clear the RDRF flag to 0. Transition of the RDRF flag from 0 to 1 can also be identified by an RXI interrupt.

[5] Serial transmission/reception continuation procedure:
To continue serial transmission/reception, before the MSB (bit 7) of the current frame is received, finish reading the RDRF flag, reading RDR, and clearing the RDRF flag to 0. Also, before the MSB (bit 7) of the current frame is transmitted, read 1 from the TDRE flag to confirm that writing is possible. Then write data to TDR and clear the TDRE flag to 0.
However, the TDRE flag is checked and cleared automatically when the DMAC or DTC is initiated by a transmit data empty interrupt (TXI) request and writes data to TDR. Similarly, the RDRF flag is cleared automatically when the DMAC or DTC is initiated by a receive data full interrupt (RXI) and reads data from RDR.

Figure 13.20 Sample Flowchart of Simultaneous Serial Transmission and Reception

13.7 Interrupt Sources

13.7.1 Interrupts in Normal Serial Communications Interface Mode

Table 13.10 shows the interrupt sources in normal serial communications interface mode. Individual interrupt sources output independent interrupt request signals (a different interrupt vector can be assigned to each interrupt source). These interrupt sources can be enabled or disabled using the enable bits in SCR.

When the TDRE flag in SSR is set to 1, a TXI interrupt request is generated. When the TEND flag in SSR is set to 1, a TEI interrupt request is generated. A TXI interrupt request can activate the DTC or DMAC to allow data transfer. The TDRE flag is automatically cleared to 0 at data transfer by the DTC or DMAC.

When the RDRF flag in SSR is set to 1, an RXI interrupt request is generated. When the ORER, PER, or FER flag in SSR is set to 1, an ERI interrupt request is generated. An RXI interrupt can activate the DTC or DMAC to allow data transfer. The RDRF flag is automatically cleared to 0 at data transfer by the DTC or DMAC.

A TEI interrupt is requested when the TEND flag is set to 1 while the TEIE bit is set to 1.

Table 13.10 SCI Interrupt Sources

Name	Interrupt Source	Interrupt Flag	DTC/DMAC Activation	Priority
ERI	Receive error	ORER, FER, or PER	Not possible	High
RXI	Receive data full	RDRF	Possible	↑
TXI	Transmit data empty	TDRE	Possible	
TEI	Transmit end	TEND	Not possible	Low

13.8 Usage Notes

13.8.1 Module Stop Mode Setting

Operation of the SCI can be disabled or enabled using the module stop control register. The initial setting is for operation of the SCI to be halted. Register access is enabled by clearing module stop mode. For details, refer to section 23, Power-Down Modes.

13.8.2 Break Detection and Processing

When framing error detection is performed, a break can be detected by reading the RxD pin value directly. In a break, the input from the RxD pin becomes all 0s, and so the FER flag is set, and the PER flag may also be set. Note that the SCI continues the receive operation even after receiving a break. Therefore, even if the FER flag is cleared to 0, it will be set to 1 again.

13.8.3 Mark State and Break Detection

When the TE bit is 0, the TxD pin is used as an I/O port whose direction (input or output) and level are determined by DR and DDR. This can be used to set the TxD pin to mark state (high level) or send a break during serial data transmission. To maintain the communication line in mark state (the state of 1) until TE is set to 1, set both PCR and PDR to 1. Since the TE bit is cleared to 0 at this point, the TxD pin becomes an I/O port, and 1 is output from the TxD pin. To send a break during serial transmission, first set PCR to 1 and PDR to 0, and then clear the TE bit to 0. When the TE bit is cleared to 0, the transmitter is initialized regardless of the current transmission state, the TxD pin becomes an I/O port, and 0 is output from the TxD pin.

13.8.4 Receive Error Flags and Transmit Operations (Clocked Synchronous Mode Only)

In simultaneous clocked synchronous transmission and reception, transmission cannot be started when a receive error flag (ORER) is set to 1, even if the TDRE flag is cleared to 0. Be sure to clear the receive error flags to 0 before starting transmission. Note also that the receive error flags cannot be cleared to 0 even if the RE bit is cleared to 0.

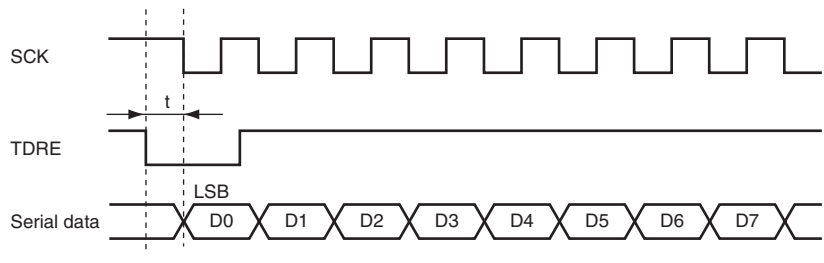
13.8.5 Relation between Writing to TDR and TDRE Flag

The TDRE flag in SSR is a status flag which indicates that transmit data has been transferred from TDR to TSR. When the SCI transfers data from TDR to TSR, the TDRE flag is set to 1.

Data can be written to TDR irrespective of the TDRE flag status. However, if new data is written to TDR when the TDRE flag is 0, that is, when the previous data has not been transferred to TSR yet, the previous data in TDR is lost. Be sure to write transmit data to TDR after verifying that the TDRE flag is set to 1.

13.8.6 Restrictions on Using DMAC or DTC

- When the external clock source is used as a synchronization clock, update TDR by the DMAC or DTC and wait for at least five Pϕ clock cycles before allowing the transmit clock to be input. If the transmit clock is input within four clock cycles after TDR modification, the SCI may malfunction (figure 13.21).
- When using the DMAC or DTC to read RDR, be sure to set the receive end interrupt (RXI) as the activation source.



Note: When external clock is supplied, t must be more than four clock cycles.

Figure 13.21 Sample Transmission using DMAC or DTC in Clocked Synchronous Mode

13.8.7 SCI Operations during Mode Transitions

(1) Transmission

Before making a transition to module stop mode or software standby mode, stop the transmit operations (TE = TIE = TEIE = 0). TSR, TDR, and SSR are reset by clearing the TE bit to 0. The states of the output pins during module stop mode or software standby mode depend on the port settings. If the transition is made during data transmission, the data being transmitted will be undefined. The output pins output a high-level signal after cancellation of these modes. To transmit data in the same transmission mode, set the TE bit to 1, write transmit data to TDR, clear the TDRE flag to 0, and then start transmission. To transmit data in a different transmission mode, initialize the SCI first. Figure 13.22 shows a sample flowchart for mode transition during transmission. Figures 13.23 and 13.24 show the port pin states during mode transition.

Before making a transition to module top mode or software standby mode from transmission by the DTC transfer, stop the transmit operations (TE = TIE = TEIE = 0). To transmit data by the DTC after mode cancellation, set bits TE and TIE to 1 to generate a TXI interrupt, then transmission by the DTC starts.

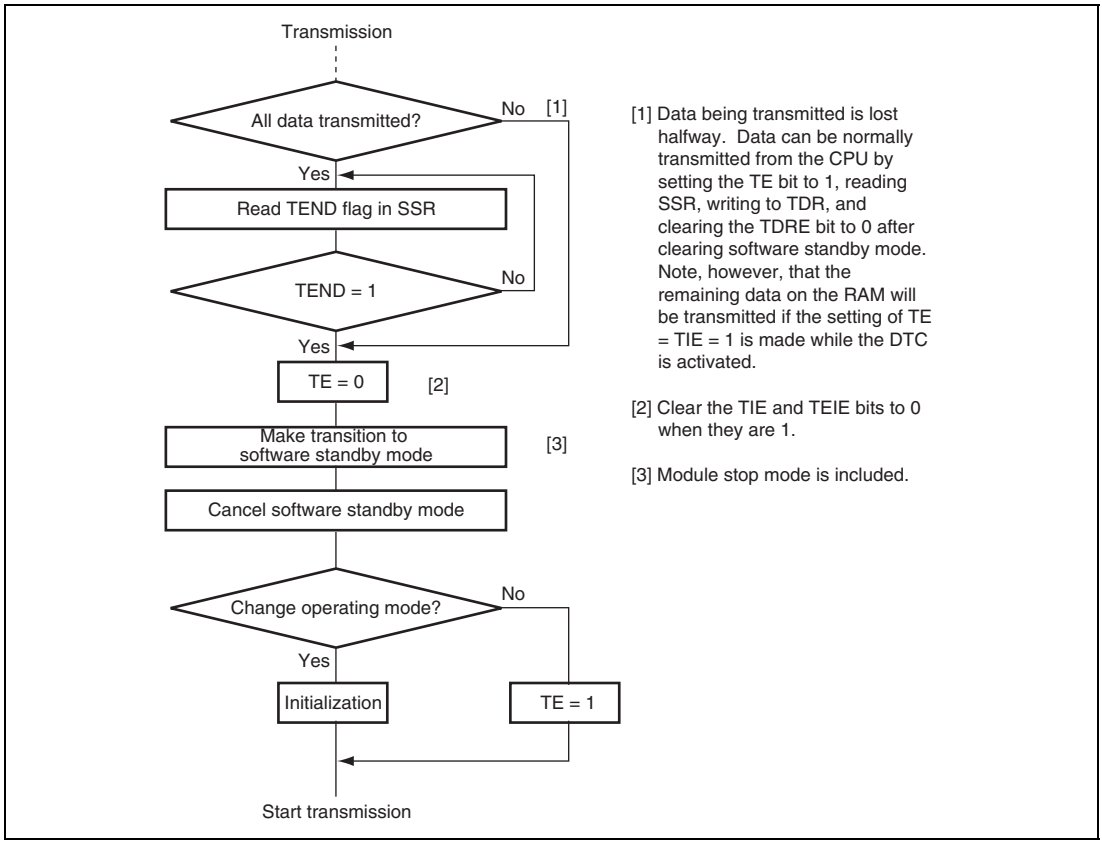


Figure 13.22 Sample Flowchart for Mode Transition during Transmission

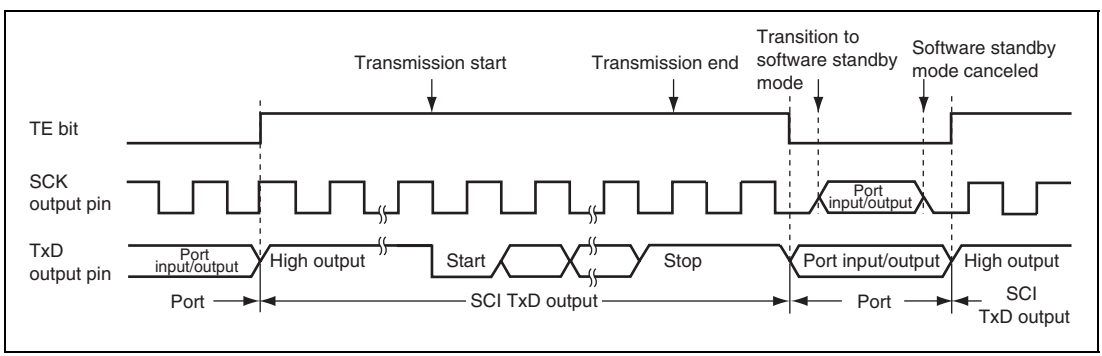


Figure 13.23 Port Pin States during Mode Transition (Internal Clock, Asynchronous Transmission)

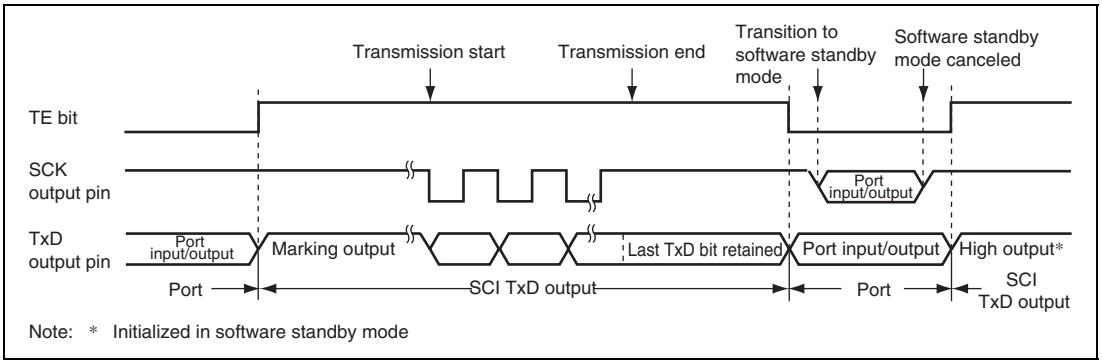


Figure 13.24 Port Pin States during Mode Transition (Internal Clock, Clocked Synchronous Transmission)

(2) Reception

Before making a transition to module stop mode or software standby mode, stop the receive operations (RE = 0). RSR, RDR, and SSR are reset by clearing the RE bit to 0. If a transition is made during data reception, the data being received will be invalid.

To receive data in the same reception mode after cancellation of these modes, set the RE bit to 1, and then start reception. To receive data in a different reception mode, initialize the SCI first. Figure 13.25 shows a sample flowchart for mode transition during reception.

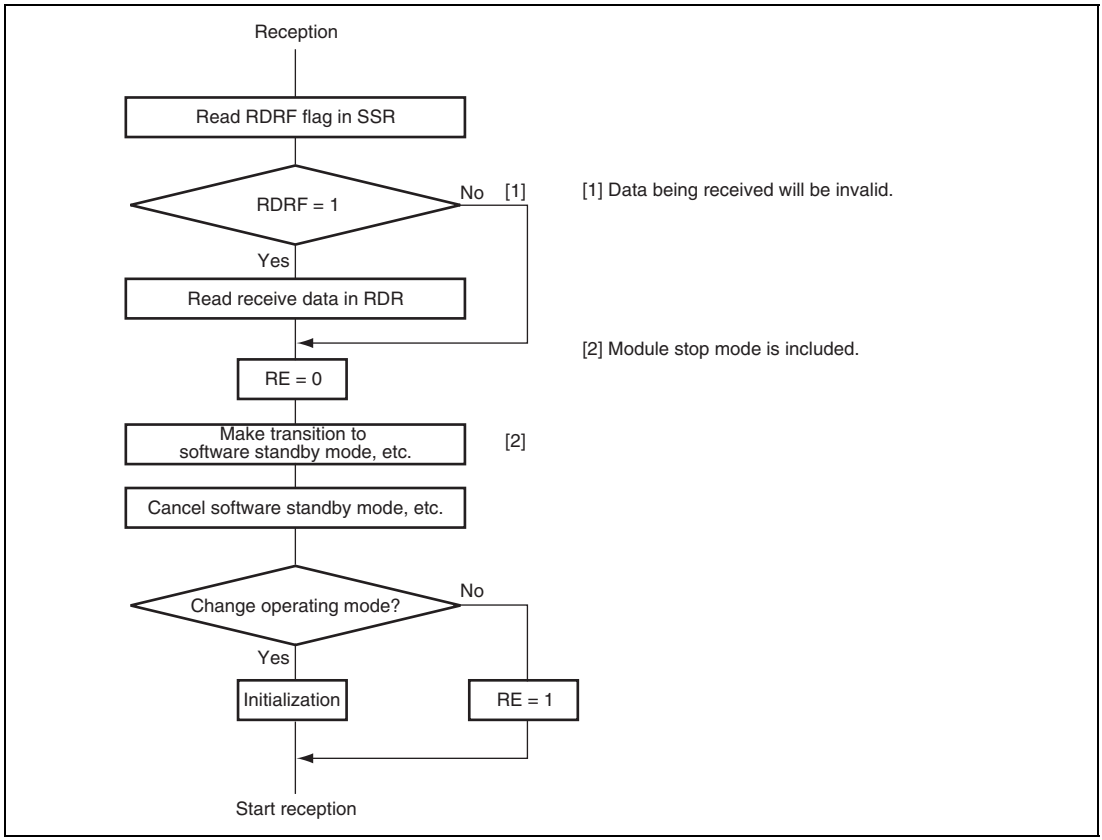


Figure 13.25 Sample Flowchart for Mode Transition during Reception

13.8.8 External Clock Input in Clocked Synchronous Mode

In clocked synchronous mode, the high pulse period and low pulse period of the externally input clock, SCK, should be specified as two or more clocks, and the cycle should be specified as six or more clocks.

Section 14 Controller Area Network (RCAN-TL1)

14.1 Summary

14.1.1 Overview

This document primarily describes the programming interface for the RCAN-TL1 (Renesas CAN Time Trigger Level 1) module. It serves to facilitate the hardware/software interface so that engineers involved in the RCAN-TL1 implementation can ensure the design is successful.

14.1.2 Scope

The CAN Data Link Controller function is not described in this document. It is the responsibility of the reader to investigate the CAN Specification Document (see references). The interfaces from the CAN Controller are described, in so far as they pertain to the connection with the User Interface.

The programming model is described in some detail. It is not the intention of this document to describe the implementation of the programming interface, but to simply present the interface to the underlying CAN functionality.

The document places no constraints upon the implementation of the RCAN-TL1 module in terms of process, packaging or power supply criteria. These issues are resolved where appropriate in implementation specifications.

14.1.3 Audience

In particular this document provides the design reference for software authors who are responsible for creating a CAN application using this module.

In the creation of the RCAN-TL1 user interface LSI engineers must use this document to understand the hardware requirements.

14.1.4 References

1. CAN Specification Version 2.0 part A, Robert Bosch GmbH, 1991
2. CAN Specification Version 2.0 part B, Robert Bosch GmbH, 1991
3. Implementation Guide for the CAN Protocol, CAN Specification 2.0 Addendum, CAN In Automation, Erlangen, Germany, 1997

4. Road vehicles - Controller area network (CAN): Part 1: Data link layer and physical signalling (ISO-11898-1, 2003)
5. Road vehicles - Controller area network (CAN): Part 4: Time triggered communication (ISO-11898-4, 2004)

14.1.5 Features

- Supports CAN specification 2.0B
- Bit timing compliant with ISO-11898-1
- 32 Mailbox version
- Clock 16 to 40 MHz
- 31 programmable Mailboxes for transmit / receive + 1 receive-only mailbox
- Sleep mode for low power consumption and automatic recovery from sleep mode by detecting CAN bus activity
- Programmable receive filter mask (standard and extended identifier) supported by all Mailboxes
- Programmable CAN data rate up to 1MBit/s
- Transmit message queuing with internal priority sorting mechanism against the problem of priority inversion for real-time applications
- Data buffer access without SW handshake requirement in reception
- Flexible micro-controller interface
- Flexible interrupt structure
- 16-bit free running timer with flexible clock sources and pre-scaler, 3 Timer Compare Match Registers
- 6-bit Basic Cycle Counter for Time Trigger Transmission
- Timer Compare Match Registers with interrupt generation
- Timer counter clear / set capability
- Registers for Time-Trigger: Local_Time, Cycle_time, Ref_Mark, Tx_Enable Window, Ref_Trigger_Offset
- Flexible TimeStamp at SOF for both transmission and reception supported
- Time-Trigger Transmission, Periodic Transmission supported (on top of Event Trigger Transmission)
- Basic Cycle value can be embedded into a CAN frame and transmitted

14.2 Architecture

The RCAN-TL1 device offers a flexible and sophisticated way to organise and control CAN frames, providing the compliance to CAN2.0B Active and ISO-11898-1. The module is formed from 5 different functional entities. These are the Micro Processor Interface (MPI), Mailbox, Mailbox Control, Timer, and CAN Interface. The figure below shows the block diagram of the RCAN-TL1 Module. The bus interface timing is designed according to the peripheral bus I/F required for each product.

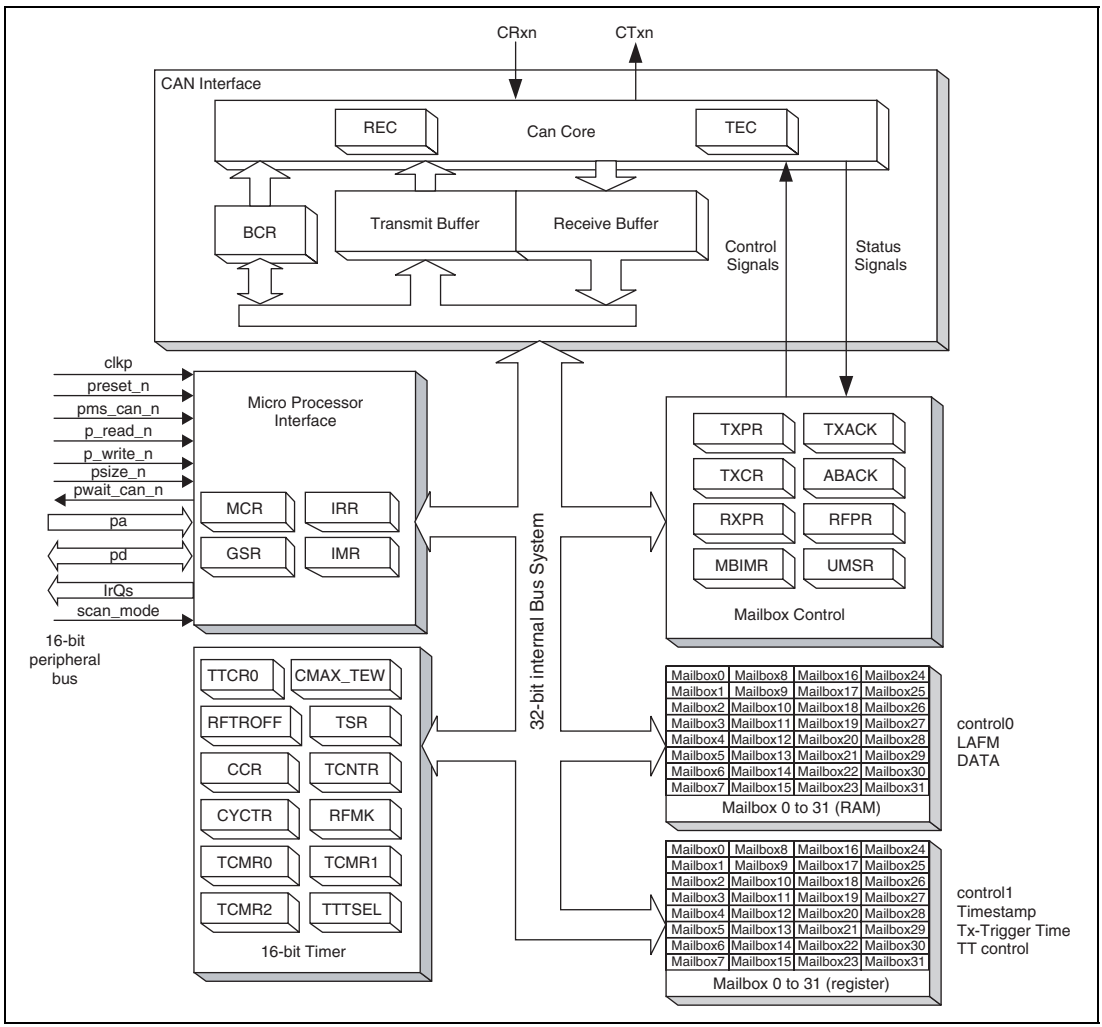


Figure 14.1 RCAN-TL1 architecture

Important: Although core of RCAN-TL1 is designed based on a 32-bit bus system, the whole RCAN-TL1 including MPI for the CPU has 16-bit bus interface to CPU. In that case, LongWord (32-bit) access must be implemented as 2 consecutive word (16-bit) accesses. In this manual, LongWord access means the two consecutive word accesses.

- Micro Processor Interface (MPI)

  The MPI allows communication between the Renesas CPU and the RCAN-TL1's registers/mailboxes to control the memory interface. It also contains the Wakeup Control logic that detects the CAN bus activities and notifies the MPI and the other parts of RCAN-TL1 so that the RCAN-TL1 can automatically exit the Sleep mode.

  It contains registers such as MCR, IRR, GSR and IMR.

- Mailbox

  The Mailboxes consists of RAM configured as message buffers and registers. There are 32 Mailboxes, and each mailbox has the following information.

  <RAM>

  — CAN message control (identifier, rtr, ide,etc)

  — CAN message data (for CAN Data frames)

  — Local Acceptance Filter Mask for reception

  <Registers>

  — CAN message control (dlc)

  — Time Stamp for message reception/transmission

  — 3-bit wide Mailbox Configuration, Disable Automatic Re-Transmission bit, Auto-Transmission for Remote Request bit, New Message Control bit

  — Tx-Trigger Time

- Mailbox Control

  The Mailbox Control handles the following functions.

  — For received messages, compare the IDs and generate appropriate RAM addresses/data to store messages from the CAN Interface into the Mailbox and set/clear appropriate registers accordingly.

  — To transmit event-triggered messages, run the internal arbitration to pick the correct priority message, and load the message from the Mailbox into the Tx-buffer of the CAN Interface and set/clear appropriate registers accordingly. In the case of time-triggered transmission, compare match of Tx-Trigger time invoke loading the messages.

  — Arbitrates Mailbox accesses between the CPU and the Mailbox Control.

  — Contains registers such as TXPR, TXCR, TXACK, ABACK, RXPR, RFPR, UMSR and MBIMR.

- Timer

  The Timer function is the functional entity, which provides RCAN-TL1 with support for transmitting messages at a specific time frame and recording the result.

  The Timer is a 16-bit free running up counter which can be controlled by the CPU. It provides one 16-bit Compare Match Register to compare with Local Time and two 16-bit ones to compare with Cycle Time. The Compare Match Registers can generate interrupt signals and clear the Counter.

  The clock period of this Timer offers a wide selection derived from the system clock or can be programmed to be incremented with one nominal bit timing of CAN Bus.

  Contains registers such as TCNTR, TTCR0, CMAX_TEW, RETROFF, TSR, CCR, CYCTR, RFMK, TCMR0, TCMR1, TCMR2 and TTTSEL.

- CAN Interface

  This block conforms to the requirements for a CAN Bus Data Link Controller which is specified in Ref. [2, 4]. It fulfils all the functions of a standard DLC as specified by the OSI 7 Layer Reference model. This functional entity also provides the registers and the logic which are specific to a given CAN bus, which includes the Receive Error Counter, Transmit Error Counter, the Bit Configuration Registers and various useful Test Modes. This block also contains functional entities to hold the data received and the data to be transmitted for the CAN Data Link Controller.

14.3 Programming Model - Overview

The purpose of this programming interface is to allow convenient, effective access to the CAN bus for efficient message transfer. Please bear in mind that the user manual reports all settings allowed by the RCAN-TL1 IP. Different use of RCAN-TL1 is not allowed.

14.3.1 Memory Map

The diagram of the memory map is shown below.

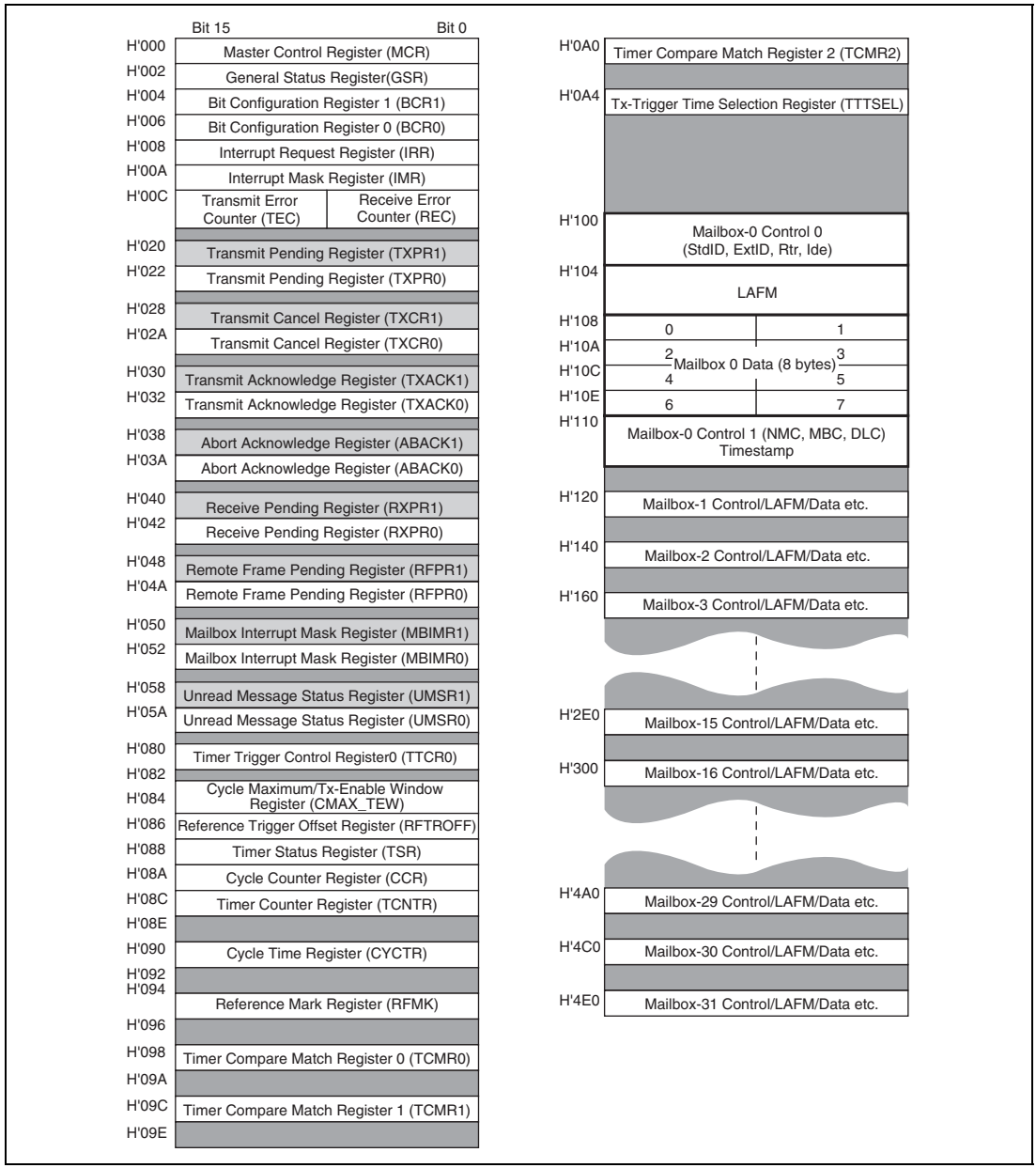


Figure 14.2 RCAN-TL1 Memory Map

The locations not used (between H'000 and H'4F3) are reserved and cannot be accessed.

14.3.2 Mailbox Structure

Mailboxes play a role as message buffers to transmit/receive CAN frames. Each Mailbox is comprised of 3 identical storage fields that are 1): Message Control, 2): Local Acceptance Filter Mask, 3): Message Data. In addition some Mailboxes contain the following extra Fields: 4): Time Stamp, 5): Time Trigger configuration and 6): Time Trigger Control. The following table shows the address map for the control, LAFM, data, timestamp, Transmission Trigger Time and Time Trigger Control addresses for each mailbox.

	Address						
	Control0	LAFM	Data	Control1	Time Stamp	Trigger Time	TT control
Mailbox	4 bytes	4 bytes	8 bytes	2 bytes	2 bytes	2 bytes	2 bytes
0 (Receive Only)	100 – 103	104– 107	108 – 10F	110 – 111	112 – 113	No	No
1	120 – 123	124 – 127	128 – 12F	130 – 131	132 – 133	No	No
2	140 – 143	144 – 147	148 – 14F	150 – 151	152 – 153	No	No
3	160 – 163	164 - 167	168 – 16F	170 – 171	172 – 173	No	No
4	180 – 183	184 – 187	188 – 18F	190 – 191	192 – 193	No	No
5	1A0 – 1A3	1A4 – 1A7	1A8 – 1AF	1B0 – 1B1	1B2 – 1B3	No	No
6	1C0 – 1C3	1C4 – 1C7	1C8 – 1CF	1D0 – 1D1	1D2 – 1D3	No	No
7	1E0 – 1E3	1E4 – 1E7	1E8 – 1EF	1F0 – 1F1	1F2 – 1F3	No	No
8	200 – 203	204 – 207	208 – 20F	210 – 211	212 – 213	No	No
9	220 – 223	224 – 227	228 – 22F	230 – 231	232 – 233	No	No
10	240 – 243	244 – 247	248 – 24F	250 – 251	252 – 253	No	No
11	260 – 263	264 – 267	268 – 26F	270 – 271	272 – 273	No	No
12	280 – 283	284 – 287	288 – 28F	290 – 291	292 – 293	No	No
13	2A0 – 2A3	2A4 – 2A7	2A8 – 2AF	2B0 – 2B1	2B2 – 2B3	No	No
14	2C0 – 2C3	2C4 – 2C7	2C8 – 2CF	2D0 – 2D1	2D2 – 2D3	No	No
15	2E0 – 2E3	2E4 – 2E7	2E8 – 2EF	2F0 – 2F1	2F2 – 2F3	No	No
16	300 – 303	304 – 307	308 – 30F	310 – 311	No	No	No
17	320 – 323	324 – 327	328 – 32F	330 – 331	No	No	No
18	340 – 343	344 – 347	348 – 34F	350 – 351	No	No	No

	Address						
	Control0	LAFM	Data	Control1	Time Stamp	Trigger Time	TT control
Mailbox	4 bytes	4 bytes	8 bytes	2 bytes	2 bytes	2 bytes	2 bytes
19	360 – 363	364 – 367	368 – 36F	370 – 371	No	No	No
20	380 – 383	384 – 387	388 – 38F	390 – 391	No	No	No
21	3A0 – 3A3	3A4 – 3A7	3A8 – 3AF	3B0 – 3B1	No	No	No
22	3C0 – 3C3	3C4 – 3C7	3C8 – 3CF	3D0 – 3D1	No	No	No
23	3E0 – 3E3	3E4 – 3E7	3E8 – 3EF	3F0 – 3F1	No	No	No
24	400 – 403	404 – 407	408 – 40F	410 – 411	No	414 – 415	416 – 417
25	420 – 423	424 – 427	428 – 42F	430 – 431	No	434 – 435	436 – 437
26	440 – 443	444 – 447	448 – 44F	450 – 451	No	454 – 455	456 – 457
27	460 – 463	464 – 467	468 – 46F	470 – 471	No	474 – 475	476 – 477
28	480 – 483	484 – 487	488 – 48F	490 – 491	No	494 – 495	496 – 497
29	4A0 – 4A3	4A4 – 4A7	4A8 – 4AF	4B0 – 4B1	No	4B4 – 4B5	4B6 – 4B7
30	4C0 – 4C3	4C4 – 4C7	4C8 – 4CF	4D0 – 4D1	4D2 – 4D3 (Local Time)	4D4 – 4D5	No
31	4E0 – 4E3	4E4 – 4E7	4E8 – 4EF	4F0 – 4F1	4F2 – 4F3 (Local Time)	No	No

Mailbox-0 is a receive-only box, and all the other Mailboxes can operate as both receive and transmit boxes, dependant upon the MBC (Mailbox Configuration) bits in the Message Control. The following diagram shows the structure of a Mailbox in detail.

Table 14.1 Roles of Mailboxes

	Event Trigger		Time Trigger		Remark	
	Tx	Rx	Tx	Rx	TimeStamp	Tx-Trigger Time
MB31	OK	OK	—	time reference reception	available	—
MB30	OK	OK	time reference transmission in time master mode	reception in time slave mode	available	available
MB29 - 24	OK	OK	OK	OK	—	available
MB23 - 16	OK	OK	— (ET)	OK	—	—
MB15 - 1	OK	OK	— (ET)	OK	available	—
MB0	—	OK	—	OK	available	—

(ET) shows that it works during merged arbitrating window, after completion of time-triggered transmission.

MB0 (reception MB with timestamp)

Byte: 8-bit access, Word: 16-bit access, LW (LongWord) : 32-bit access

Address	Data Bus 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Access Size	Field Name
H'100	IDE	RTR	0	STDID[10:0]											EXTID[17:16]		Word/LW	Control 0
H'102	EXTID[15:0]																Word	
H'104	IDE LAFM	0	0	STDID_LAFM[10:0]											EXTID LAFM[17:16]		Word/LW	LAFM
H'106	EXTID_LAFM[15:0]																Word	
H'108	MSG_DATA_0 (first Rx/Tx Byte)								MSG_DATA_1								Byte/Word/LW	Data
H'10A	MSG_DATA_2								MSG_DATA_3								Byte/Word	
H'10C	MSG_DATA_4								MSG_DATA_5								Byte/Word/LW	
H'10E	MSG_DATA_6								MSG_DATA_7								Byte/Word	
H'110	0	0	NMC	0	0	MBC[2:0]*			0	0	0	0	DLC[3:0]				Byte/Word	Control 1
H'112	TimeStamp[15:0] (CYCTR[15:0] or CCR[5:0]/CYCTR[15:6] at SOF)																Word	TimeStamp

* MBC[1] is fixed to "1"

MB1 to MB15 (MB with timestamp)

Address	Data Bus 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Access Size	Field Name
H'100 + N*32	IDE	RTR	0	STDID[10:0]											EXTID[17:16]		Word/LW	Control 0
H'102 + N*32	EXTID[15:0]																Word	
H'104 + N*32	IDE LAFM	0	0	STDID_LAFM[10:0]											EXTID LAFM[17:16]		Word/LW	LAFM
H'106 + N*32	EXTID_LAFM[15:0]																Word	
H'108 + N*32	MSG_DATA_0 (first Rx/Tx Byte)								MSG_DATA_1								Byte/Word/LW	Data
H'10A + N*32	MSG_DATA_2								MSG_DATA_3								Byte/Word	
H'10C + N*32	MSG_DATA_4								MSG_DATA_5								Byte/Word/LW	
H'10E + N*32	MSG_DATA_6								MSG_DATA_7								Byte/Word	
H'110 + N*32	0	0	NMC	ATX	DART	MBC[2:0]			0	0	0	0	DLC[3:0]				Byte/Word	Control 1
H'112 + N*32	TimeStamp[15:0] (CYCTR[15:0] or CCR[5:0]/CYCTR[15:6] at SOF)																Word	TimeStamp

Figure 14.3 Mailbox-N Structure

MB16 to MB23 (MB without timestamp)

Address	Data Bus																Access Size	Field Name
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
H'100 + N*32	IDE	RTR	0	STDID[10:0]											EXTID[17:16]		Word/LW	Control 0
H'102 + N*32	EXTID[15:0]																Word	
H'104 + N*32	IDE LAFM	0	0	STDID_LAFM[10:0]											EXTID_ LAFM[17:16]		Word/LW	LAFM
H'106 + N*32	EXTID_LAFM[15:0]																Word	
H'108 + N*32	MSG_DATA_0 (first Rx/Tx Byte)								MSG_DATA_1								Byte/Word/LW	Data
H'10A + N*32	MSG_DATA_2								MSG_DATA_3								Byte/Word	
H'10C + N*32	MSG_DATA_4								MSG_DATA_5								Byte/Word/LW	
H'10E + N*32	MSG_DATA_6								MSG_DATA_7								Byte/Word	
H'110 + N*32	0	0	NMC	ATX	DART	MBC[2:0]			0	0	0	0	DLC[3:0]				Byte/Word	Control 1

MB24 to MB29 (Time-Triggered Transmission in Time Trigger mode)

Address	Data Bus																Access Size	Field Name
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
H'100 + N*32	IDE	RTR	0	STDID[10:0]											EXTID[17:16]		Word/LW	Control 0
H'102 + N*32	EXTID[15:0]																Word	
H'104 + N*32	IDE LAFM	0	0	STDID_LAFM[10:0]											EXTID_ LAFM[17:16]		Word/LW	LAFM
H'106 + N*32	EXTID_LAFM[15:0]																Word	
H'108 + N*32	MSG_DATA_0 (first Rx/Tx Byte)								MSG_DATA_1								Byte/Word/LW	Data
H'10A + N*32	MSG_DATA_2								MSG_DATA_3								Byte/Word	
H'10C + N*32	MSG_DATA_4								MSG_DATA_5								Byte/Word/LW	
H'10E + N*32	MSG_DATA_6								MSG_DATA_7								Byte/Word	
H'110 + N*32	0	0	NMC	ATX	DART	MBC[2:0]			0	0	0	0	DLC[3:0]				Byte/Word	Control 1
H'112 + N*32	reserved																-	-
H'114 + N*32	Tx-Triggered Time (TTT)																Word	Trigger Time
H'116 + N*32	TTW[1:0]		offset						0	0	0	0	0	Rep_Factor			Word	TT control

Figure 14.3 Mailbox-N Structure (continued)

MB30 (Time Reference Transmitssion in Time Trigger mode)

Address	Data Bus																Access Size	Field Name
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
H'100 + N*32	IDE	RTR	0	STDID[10:0]											EXTID[17:16]		Word/LW	Control 0
H'102 + N*32	EXTID[15:0]																Word	
H'104 + N*32	IDE LAFM	0	0	STDID_LAFM[10:0]											EXTID LAFM[17:16]		Word/LW	LAFM
H'106 + N*32	EXTID_LAFM[15:0]																Word	
H'108 + N*32	MSG_DATA_0 (first Rx/Tx Byte)								MSG_DATA_1								Byte/Word/LW	Data
H'10A + N*32	MSG_DATA_2								MSG_DATA_3								Byte/Word	
H'10C + N*32	MSG_DATA_4								MSG_DATA_5								Byte/Word/LW	
H'10E + N*32	MSG_DATA_6								MSG_DATA_7								Byte/Word	
H'110 + N*32	0	0	NMC	ATX	DART	MBC[2:0]			0	0	0	0	DLC[3:0]				Byte/Word	Control 1
H'112 + N*32	TimeStamp[15:0] (TCNTR at SOF)																Word	TimeStamp
H'114 + N*32	Tx-Triggered Time (TTT) as Time Reference																Word	Trigger Time

MB31 (Time Reference Reception in Time Trigger mode)

Address	Data Bus																Access Size	Field Name
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
H'100 + N*32	IDE	RTR	0	STDID[10:0]											EXTID[17:16]		Word/LW	Control 0
H'102 + N*32	EXTID[15:0]																Word	
H'104 + N*32	IDE LAFM	0	0	STDID_LAFM[10:0]											EXTID LAFM[17:16]		Word/LW	LAFM
H'106 + N*32	EXTID_LAFM[15:0]																Word	
H'108 + N*32	MSG_DATA_0 (first Rx/Tx Byte)								MSG_DATA_1								Byte/Word/LW	Data
H'10A + N*32	MSG_DATA_2								MSG_DATA_3								Byte/Word	
H'10C + N*32	MSG_DATA_4								MSG_DATA_5								Byte/Word/LW	
H'10E + N*32	MSG_DATA_6								MSG_DATA_7								Byte/Word	
H'110 + N*32	0	0	NMC	ATX	DART	MBC[2:0]			0	0	0	0	DLC[3:0]				Byte/Word	Control 1
H'112 + N*32	TimeStamp[15:0] (TCNTR at SOF)																Word	TimeStamp

Figure 14.3 Mailbox-N Structure (continued)

Notes: 1. All bits shadowed in grey are reserved and must be written LOW. The value returned by a read may not always be '0' and should not be relied upon.

2. ATX and DART are not supported by Mailbox-0, and the MBC setting of Mailbox-0 is limited.

3. Since the initial value of ID Reorder (bit MCR15) is 1, the order of STDID, RTR, IDE and EXTID of both message control and LAFM is initially different from that of HCAN2.

(1) Message Control Field

STDID[10:0]: These bits set the identifier (standard identifier) of data frames and remote frames.

EXTID[17:0]: These bits set the identifier (extended identifier) of data frames and remote frames.

RTR (Remote Transmission Request bit): Used to distinguish between data frames and remote frames. This bit is overwritten by received CAN Frames depending on Data Frames or Remote Frames.

Important: Please note that, when ATX bit is set with the setting MBC = 001(bin), the RTR bit will never be set. When a Remote Frame is received, the CPU can be notified by the corresponding RFPR set or IRR[2] (Remote Frame Receive Interrupt), however, as RCAN-TL1 needs to transmit the current message as a Data Frame, the RTR bit remains unchanged.

Important: In order to support automatic answer to remote frame when MBC = 001 (bin) is used and ATX = 1 the RTR flag must be programmed to zero to allow data frame to be transmitted.

Note: when a Mailbox is configured to send a remote frame request the DLC used for transmission is the one stored into the Mailbox.

RTR	Description
0	Data frame
1	Remote frame

IDE (Identifier Extension bit): Used to distinguish between the standard format and extended format of CAN data frames and remote frames.

IDE	Description
0	Standard format
1	Extended format

- Mailbox-0

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	NMC	0	0	MBC[2:0]			0	0	0	0	DLC[3:0]			
Initial value:	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
R/W:	R	R	R/W	R	R	R/W	R	R/W	R	R	R	R	R/W	R/W	R/W	R/W

Note: MBC[1] of MB0 is always "1".

- Mailbox-31 to 1

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	NMC	ATX	DART	MBC[2:0]			0	0	0	0	DLC[3:0]			
Initial value:	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
R/W:	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R/W	R/W

NMC (New Message Control): When this bit is set to ‘0’, the Mailbox of which the RXPR or RFPR bit is already set does not store the new message but maintains the old one and sets the UMSR correspondent bit. When this bit is set to ‘1’, the Mailbox of which the RXPR or RFPR bit is already set overwrites with the new message and sets the UMSR correspondent bit.

Important: Please note that if a remote frame is overwritten with a data frame or vice versa could be that both RXPR and RFPR flags (together with UMSR) are set for the same Mailbox. In this case the RTR bit within the Mailbox Control Field should be relied upon.

Important: Please note that when the Time Triggered mode is used NMC needs to be set to ‘1’ for Mailbox 31 to allow synchronization with all incoming reference messages even when RXPR[31] is not cleared.

NMC	Description
0	Overrun mode (Initial value)
1	Overwrite mode

ATX (Automatic Transmission of Data Frame): When this bit is set to '1' and a Remote Frame is received into the Mailbox DLC is stored. Then, a Data Frame is transmitted from the same Mailbox using the current contents of the message data and updated DLC by setting the corresponding TXPR automatically. The scheduling of transmission is still governed by ID priority or Mailbox priority as configured with the Message Transmission Priority control bit (MCR.2). In order to use this function, MBC[2:0] needs to be programmed to be '001' (Bin). When a transmission is performed by this function, the DLC (Data Length Code) to be used is the one that has been received. Application needs to guarantee that the DLC of the remote frame correspond to the DLC of the data frame requested.

Important: When ATX is used and MBC = 001 (Bin) the filter for the IDE bit cannot be used since ID of remote frame has to be exactly the same as that of data frame as the reply message.

Important: Please note that, when this function is used, the RTR bit will never be set despite receiving a Remote Frame. When a Remote Frame is received, the CPU will be notified by the corresponding RFPR set, however, as RCAN-TL1 needs to transmit the current message as a Data Frame, the RTR bit remains unchanged.

Important: Please note that in case of overrun condition (UMSR) flag set when the Mailbox has its NMC = 0), the message received is discarded. In case a remote frame is causing overrun into a Mailbox configured with ATX = 1, a request to automatically transmit the corresponding message may be accepted.

ATX	Description
0	Automatic Transmission of Data Frame disabled (Initial value)
1	Automatic Transmission of Data Frame enabled

DART (Disable Automatic Re-Transmission): When this bit is set, it disables the automatic re-transmission of a message in the event of an error on the CAN bus or an arbitration lost on the CAN bus. In effect, when this function is used, the corresponding TXCR bit is automatically set at the start of transmission. When this bit is set to '0', RCAN-TL1 tries to transmit the message as many times as required until it is successfully transmitted or it is cancelled by the TXCR.

DART	Description
0	Re-transmission enabled (Initial value)
1	Re-Transmission disabled

MBC[2:0] (Mailbox Configuration): These bits configure the nature of each Mailbox as follows. When MBC = 111 (Bin), the Mailbox is inactive, i.e., it does not receive or transmit a message regardless of TXPR or other settings. The MBC = ’110’, ‘101’ and ‘100’ settings are prohibited. When the MBC is set to any other value, the LAFM field becomes available. Please don't set TXPR when MBC is set as reception as there is no hardware protection, and TXPR will remain set. MBC[1] of Mailbox-0 is fixed to "1" by hardware. This is to ensure that MB0 cannot be configured to transmit Messages.

MBC[2]	MBC[1]	MBC[0]	Data Frame Transmit	Remote Frame Transmit	Data Frame Receive	Remote Frame Receive	Remarks
0	0	0	Yes	Yes	No	No	• Not allowed for Mailbox-0 • Time-Triggered transmission can be used
0	0	1	Yes	Yes	No	Yes	• Can be used with ATX* • Not allowed for Mailbox-0 • LAFM can be used
0	1	0	No	No	Yes	Yes	• Allowed for Mailbox-0 • LAFM can be used
0	1	1	No	No	Yes	No	• Allowed for Mailbox-0 • LAFM can be used
1	0	0	Setting prohibited				
1	0	1	Setting prohibited				
1	1	0	Setting prohibited				
1	1	1	Mailbox inactive (Initial value)				

Notes: * In order to support automatic retransmission, RTR shall be "0" when MBC = 001(bin) and ATX = 1.
When ATX = 1 is used the filter for IDE must not be used.

DLC[3:0] (Data Length Code): These bits encode the number of data bytes from 0,1, 2, … 8 that will be transmitted in a data frame. Please note that when a remote frame request is transmitted the DLC value to be used must be the same as the DLC of the data frame that is requested.

DLC[3]	DLC[2]	DLC[1]	DLC[0]	Description
0	0	0	0	Data Length = 0 bytes (Initial value)
0	0	0	1	Data Length = 1 byte
0	0	1	0	Data Length = 2 bytes
0	0	1	1	Data Length = 3 bytes
0	1	0	0	Data Length = 4 bytes
0	1	0	1	Data Length = 5 bytes
0	1	1	0	Data Length = 6 bytes
0	1	1	1	Data Length = 7 bytes
1	x	x	x	Data Length = 8 bytes

(2) Local Acceptance Filter Mask (LAFM)

This area is used as Local Acceptance Filter Mask (LAFM) for receive boxes**.**

LAFM: When MBC is set to 001, 010, 011(Bin), this field is used as LAFM Field. It allows a Mailbox to accept more than one identifier. The LAFM is comprised of two 16-bit read/write areas as follows.

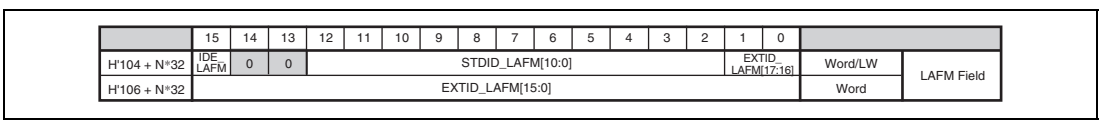


Figure 14.4 Acceptance filter

If a bit is set in the LAFM, then the corresponding bit of a received CAN identifier is ignored when the RCAN-TL1 searches a Mailbox with the matching CAN identifier. If the bit is cleared, then the corresponding bit of a received CAN identifier must match to the STDID/IDE/EXTID set in the mailbox to be stored. The structure of the LAFM is same as the message control in a Mailbox. If this function is not required, it must be filled with ‘0’.

Important: RCAN-TL1 starts to find a matching identifier from Mailbox-31 down to Mailbox-0. As soon as RCAN-TL1 finds one matching, it stops the search. The message will be stored or not depending on the NMC and RXPR/RFPR flags. This means that, even using LAFM, a received message can only be stored into 1 Mailbox.

Important: When a message is received and a matching Mailbox is found, the whole message is stored into the Mailbox. This means that, if the LAFM is used, the STDID, RTR, IDE and EXTID may differ to the ones originally set as they are updated with the STDID, RTR, IDE and EXTID of the received message.

STD_LAFM[10:0] — Filter mask bits for the CAN base identifier [10:0] bits.

STD_LAFM[10:0]	Description
0	Corresponding STD_ID bit is cared
1	Corresponding STD_ID bit is "don't cared"

EXT_LAFM[17:0] — Filter mask bits for the CAN Extended identifier [17:0] bits.

EXT_LAFM[17:0]	Description
0	Corresponding EXT_ID bit is cared
1	Corresponding EXT_ID bit is "don't cared"

IDE_LAFM — Filter mask bit for the CAN IDE bit.

IDE_LAFM	Description
0	Corresponding IDE bit is cared
1	Corresponding IDE bit is "don't cared"

(3) Message Data Fields

Storage for the CAN message data that is transmitted or received. MSG_DATA[0] corresponds to the first data byte that is transmitted or received. The bit order on the CAN bus is bit 7 through to bit 0.

When CMAX!= 3'b111/MBC[30] = 3'b000 and TXPR[30] is set, Mailbox-30 is configured as transmission of time reference. Its DLC must be greater than 0 and its RTR must be zero (as specified for TTCAN Level 1) so that the Cycle_count (CCR register) is embedded in the first byte of the data field instead of MSG_DATA_0[5:0] when this Mailbox starts transmission. This function shall be used when RCAN-TL1 is enabled to work in TTCAN mode to perform a Potential Time Master role to send the Time reference message. MSG_DATA_0[7:6] is still transmitted as stored in the Mailbox. User can set MSG_DATA_0[7] when a Next_is_Gap needs to be transmitted.

Please note that the CCR value is only embedded on the frame transmitted but not stored back into Mailbox 30.

When CMAX!= 3'b111, MBC[31] = 3'b011 and TXPR[31] is cleared, Mailbox-31 is configured as reception of time reference. When a valid reference message is received (DLC > 0) RCAN-TL1 performs internal synchronisation (modifying its RFMK and basic cycle CCR).

MB30 - 31

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
H'108 + N*32	Next_is_Gap/Cycle_Counter (first Rx/Tx Byte)								MSG_DATA_1								Byte/Word/LW	Data
H'10A + N*32	MSG_DATA_2								MSG_DATA_3								Byte/Word	
H'10C + N*32	MSG_DATA_4								MSG_DATA_5								Byte/Word/LW	
H'10E + N*32	MSG_DATA_6								MSG_DATA_7								Byte/Word	

Figure 14.5 Message Data Field

(4) Timestamp

Storage for the Timestamp recorded on messages for transmit/receive. The Timestamp will be a useful function to monitor if messages are received/transmitted within expected schedule.

- Timestamp

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TS15	TS14	TS13	TS12	TS11	TS10	TS9	TS8	TS7	TS6	TS5	TS4	TS3	TS2	TS1	TS0
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Message Receive: For received messages of Mailbox-15 to 0, Timestamp always captures the CYCTR (Cycle Time Register) value or Cycle_Counter CCR[5:0] + CYCTR[15:6] value, depending on the programmed value in the bit 14 of TTCR0 (Timer Trigger Control Register 0) at SOF.

For messages received into Mailboxes 30 and 31, Timestamp captures the TCNTR (Timer Counter Register) value at SOF.

Message Transmit: For transmitted messages of Mailbox-15 to 1, Timestamp always captures the CYCTR (Cycle Time Register) value or Cycle_Counter CCR[5:0] + CYCTR[15:6] value, depending on the programmed value in the bit 14 of TTCR0 (Timer Trigger Control Register 0), at SOF.

For messages transmitted from Mailboxes30 and 31, Timestamp captures the TCNTR (Timer Counter Register) value at SOF.

Important: Please note that the TimeStamp is stored in a temporary register. Only after a successful transmission or reception the value is then copied into the related Mailbox field. The TimeStamp may also be updated if the CPU clears RXPR[N]/RFPR[N] at the same time that UMSR[N] is set in overrun, however it can be read properly before clearing RXPR[N]/RFPR[N].

(5) Tx-Trigger Time (TTT) and Time Trigger control

For Mailbox-29 to 24, when MBC is set to 000 (Bin) in time trigger mode (CMAX!= 3'b111), Tx-Trigger Time works as Time_Mark to determine the boundary between time windows. The TTT and TT control are comprised of two 16-bit read/write areas as follows. Mailbox-30 doesn't have TT control and works as Time_Ref.

Mailbox 30 to 24 can be used for reception if not used for transmission in TT mode. However they cannot join the event trigger transmission queue when the TT mode is used.

- Tx-Trigger Time

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TTT15	TTT14	TTT13	TTT12	TTT11	TTT10	TTT9	TTT8	TTT7	TTT6	TTT5	TTT4	TTT3	TTT2	TTT1	TTT0
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- Time Trigger control

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TTW[1:0]		Offset[5:0]						0	0	0	0	0	rep_factor[2:0]		
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R/W	R/W	R/W

The following figure shows the differences between all Mailboxes supporting Time Triggered mode.

MB29 to 24

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
H'114 + N*32	Tx-Trigger Time (Cycle Time)																Word	Trigger Time
H'116 + N*32	TTW[1:0]		Offset[5:0]						0	0	0	0	0	rep_factor[2:0]			Word	TT control

MB30

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
H'114 + N*32	Tx-Trigger Time (Cycle Time)																Word	Trigger Time

Figure 14.6 Tx-Trigger control field

- **TTW[1:0] (Time Trigger Window):** These bits show the attribute of time windows. Please note that once a merged arbitrating window is opened by TTW = 2'b10, the window must be closed by TTW = 2'b11. Several messages with TTW = 2'b10 may be used within the start and the end of a merged arbitrating window.

TTW[1]	TTW[0]	Description
0	0	Exclusive window (initial value)
0	1	Arbitrating window
1	0	Start of merged arbitrating window
1	1	End of merged arbitrating window

The first 16-bit area specifies the time that triggers the transmission of the message in cycle time. The second 16-bit area specifies the basic cycle in the system matrix where the transmission must start (Offset) and the frequency for periodic transmission. When the internal TTT register matches to the CYCTR value, and the internal Offset matches to CCR value transmission is attempted from the corresponding Mailbox. In order to enable this function, the CMAX (Cycle Maximum Register) must be set to a value different from 3'b111, the Timer (TCNTR) must be running (TTCR0 bit15 = 1), the corresponding MBC must be set to 3'b000 and the corresponding TXPR bit must be set. Once TXPR is set by S/W, RCAN-TL1 does not clear the corresponding TXPR bit (among Mailbox-30 to 24) to carry on performing the periodic transmission. In order to stop the periodic transmission, TXPR must be cleared by TXCR. Please note that in this case it is possible that both TXACK and ABACK are set for the same Mailbox if TXACK is not cleared right after completion of transmission. Please refer to Figure 14.7.

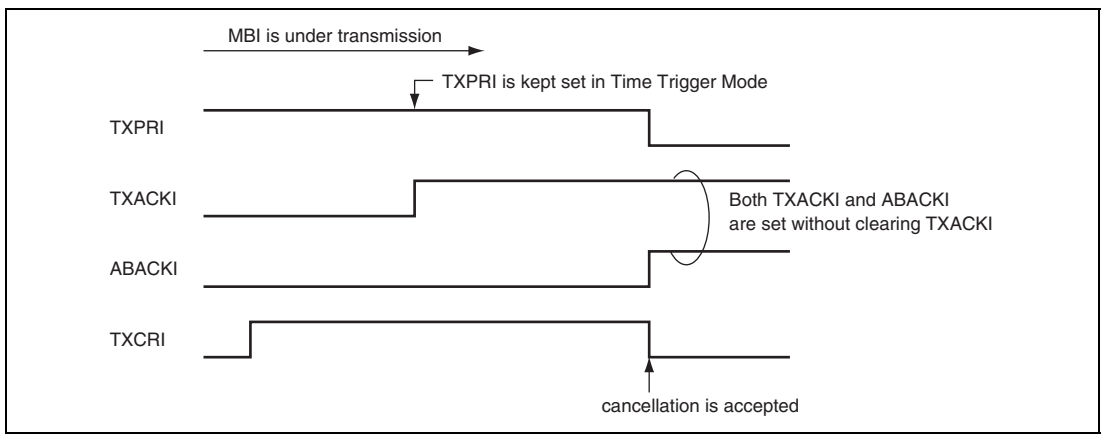


Figure 14.7 TXACK and ABACK in Time Trigger Transmission

Please note that for Mailbox 30 TTW is fixed to ‘01’, Offset to ‘00’ and rep_factor to ‘0’.The following tables report the combinations for the rep_factor and the offset.

Rep_factor	Description
3'b000	Every basic cycle (initial value)
3'b001	Every two basic cycle
3'b010	Every four basic cycle
3'b011	Every eight basic cycle
3'b100	Every sixteen basic cycle
3'b101	Every thirty two basic cycle
3'b110	Every sixty four basic cycle (once in system matrix)
3'b111	Reserved

The Offset Field determines the first cycle in which a Time Triggered Mailbox may start transmitting its Message.

Offset	Description
6'b000000	Initial Offset = 1st Basic Cycle (initial value)
6'b000001	Initial Offset = 2nd Basic Cycles
6'b000010	Initial Offset = 3rd Basic Cycles
6'b000011	Initial Offset = 4th Basic Cycles
6'b000100	Initial Offset = 5th Basic Cycles
…	
…	
6'b111110	Initial Offset = 63rd Basic Cycles
6'b111111	Initial Offset = 64th Basic Cycles

The following relation must be maintained:

Cycle_Count_Maximum + 1 >= Repeat_Factor > Offset

Cycle_Count_Maximum $= 2^{CMAX} - 1$

Repeat_Factor = 2^{rep_factor}

CMAX, Repeat_Factor, and Offset are register values

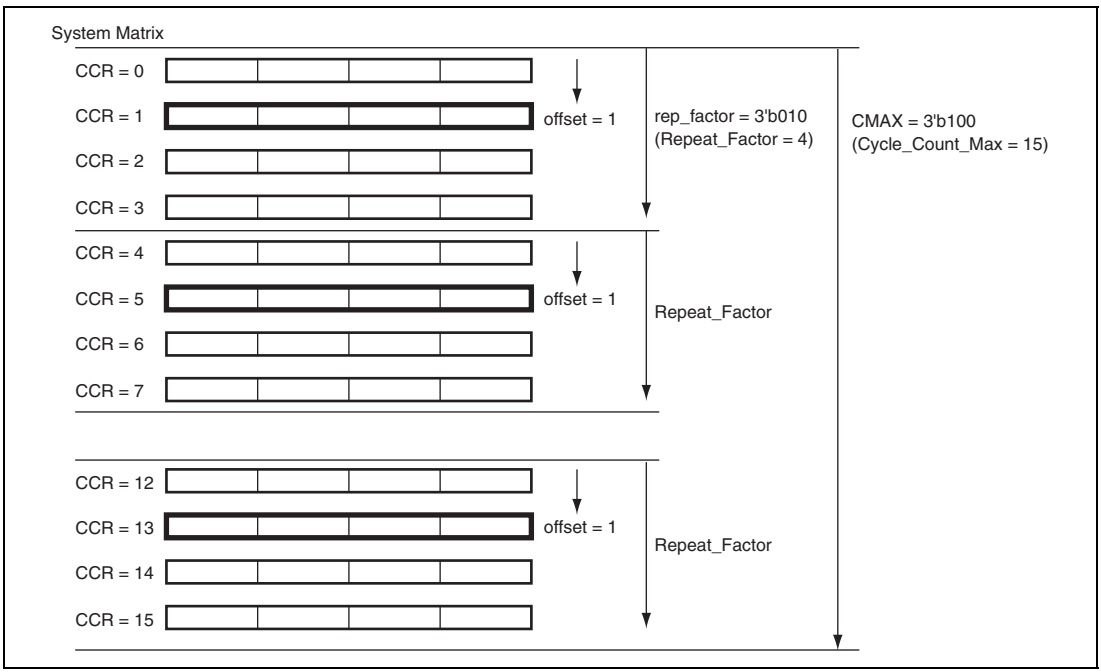


Figure 14.8 System Matrix

The Tx-Trigger Time must be set in ascending order. Please bear in mind that a minimum difference of TEW's width between Tx-Trigger Times is allowed.

14.3.3 RCAN-TL1 Control Registers

The following sections describe RCAN-TL1 control registers. The address is mapped as follow.

Important: These registers can only be accessed in Word size (16-bit).

Description	Address	Name	Access Size (bits)
Master Control Register	000	MCR	Word
General Status Register	002	GSR	Word
Bit Configuration Register 1	004	BCR1	Word
Bit Configuration Register 0	006	BCR0	Word
Interrupt Register	008	IRR	Word
Interrupt Mask Register	00A	IMR	Word
Error Counter Register	00C	TEC/REC	Word

Figure 14.9 RCAN-TL1 control registers

(1) Master Control Register (MCR)

The Master Control Register (MCR) is a 16-bit read/write register that controls RCAN-TL1.

- MCR (Address = H'000)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MCR15	MCR14	-	-	-	TST[2:0]			MCR7	MCR6	MCR5	-	-	MCR2	MCR1	MCR0
Initial value:	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R/W:	R/W	R/W	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W

Bit 15 — ID Reorder (MCR15): This bit changes the order of STDID, RTR, IDE and EXTID of both message control and LAFM.

Bit15: MCR15	Description
0	RCAN-TL1 is the same as HCAN2
1	RCAN-TL1 is not the same as HCAN2 (Initial value)

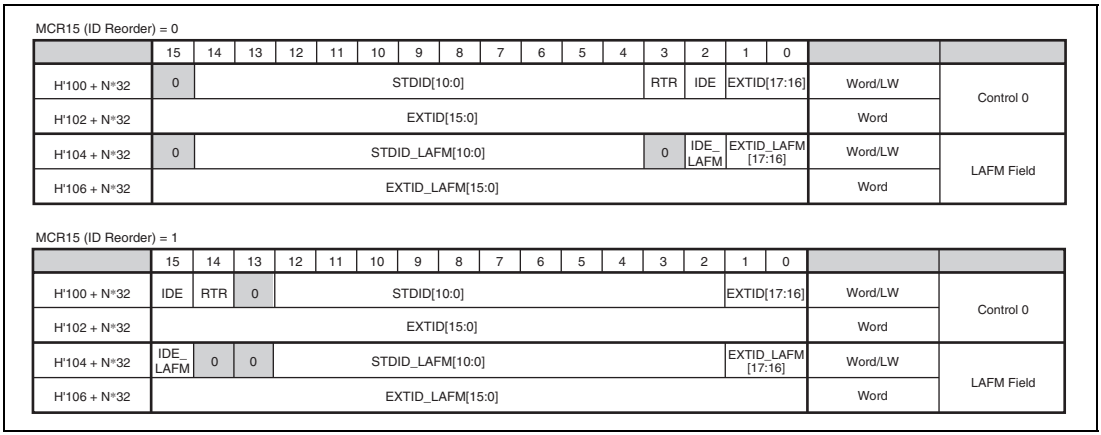


Figure 14.10 ID Reorder

This bit can be modified only in reset mode.

Bit 14 — Auto Halt Bus Off (MCR14): If both this bit and MCR6 are set, MCR1 is automatically set as soon as RCAN-TL1 enters BusOff.

Bit14: MCR14	Description
0	RCAN-TL1 remains in BusOff for normal recovery sequence (128 x 11 Recessive Bits) (Initial value)
1	RCAN-TL1 moves directly into Halt Mode after it enters BusOff if MCR6 is set.

This bit can be modified only in reset mode.

Bit 13 — Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bit 12 — Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bit 11 — Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bit 10 - 8 — Test Mode (TST[2:0]): This bit enables/disables the test modes. Please note that before activating the Test Mode it is requested to move RCAN-TL1 into Halt mode or Reset mode. This is to avoid that the transition to Test Mode could affect a transmission/reception in progress. For details, please refer to section 14.4.1, Test Mode Settings.

Please note that the test modes are allowed only for diagnosis and tests and not when RCAN-TL1 is used in normal operation.

Bit10: TST2	Bit9: TST1	Bit8: TST0	Description
0	0	0	Normal Mode (initial value)
0	0	1	Listen-Only Mode (Receive-Only Mode)
0	1	0	Self Test Mode 1 (External)
0	1	1	Self Test Mode 2 (Internal)
1	0	0	Write Error Counter
1	0	1	Error Passive Mode
1	1	0	Setting prohibited
1	1	1	Setting prohibited

Bit 7 — Auto-wake Mode (MCR7): MCR7 enables or disables the Auto-wake mode. If this bit is set, the RCAN-TL1 automatically cancels the sleep mode (MCR5) by detecting CAN bus activity (dominant bit). If MCR7 is cleared the RCAN-TL1 does not automatically cancel the sleep mode.

RCAN-TL1 cannot store the message that wakes it up.

Note: This bit can be modified only Reset or Halt mode.

Bit7: MCR7	Description
0	Auto-wake by CAN bus activity disabled (Initial value)
1	Auto-wake by CAN bus activity enabled

Bit 6 — Halt during Bus Off (MCR6): MCR6 enables or disables entering Halt mode immediately when MCR1 is set during Bus Off. This bit can be modified only in Reset or Halt mode. Please note that when Halt is entered in Bus Off the CAN engine is also recovering immediately to Error Active mode.

Bit6: MCR6	Description
0	If MCR[1] is set, RCAN-TL1 will not enter Halt mode during Bus Off but wait up to end of recovery sequence (Initial value)
1	Enter Halt mode immediately during Bus Off if MCR[1] or MCR[14] are asserted.

Bit 5 — Sleep Mode (MCR5): Enables or disables Sleep mode transition. If this bit is set, while RCAN-TL1 is in halt mode, the transition to sleep mode is enabled. Setting MCR5 is allowed after entering Halt mode. The two Error Counters (REC, TEC) will remain the same during Sleep mode. This mode will be exited in two ways:

1. by writing a '0' to this bit position,
2. or, if MCR[7] is enabled, after detecting a dominant bit on the CAN bus.

If Auto wake up mode is disabled, RCAN-TL1 will ignore all CAN bus activities until the sleep mode is terminated. When leaving this mode the RCAN-TL1 will synchronise to the CAN bus (by checking for 11 recessive bits) before joining CAN Bus activity. This means that, when the No.2 method is used, RCAN-TL1 will miss the first message to receive. CAN transceivers stand-by mode will also be unable to cope with the first message when exiting stand by mode, and the S/W needs to be designed in this manner.

In sleep mode only the following registers can be accessed: MCR, GSR, IRR and IMR.

Important: RCAN-TL1 is required to be in Halt mode before requesting to enter in Sleep mode. That allows the CPU to clear all pending interrupts before entering sleep mode. Once all interrupts are cleared RCAN-TL1 must leave the Halt mode and enter Sleep mode simultaneously (by writing MCR[5] = 1 and MCR[1] = 0 at the same time).

Bit 5: MCR5	Description
0	RCAN-TL1 sleep mode released (Initial value)
1	Transition to RCAN-TL1 sleep mode enabled

Bit 4 — Reserved. The written value should always be '0' and the returned value is '0'.

Bit 3 — Reserved. The written value should always be '0' and the returned value is '0'.

Bit 2 — Message Transmission Priority (MCR2): MCR2 selects the order of transmission for pending transmit data. If this bit is set, pending transmit data are sent in order of the bit position in the Transmission Pending Register (TXPR). The order of transmission starts from Mailbox-31 as the highest priority, and then down to Mailbox-1 (if those mailboxes are configured for transmission). Please note that this feature cannot be used for time trigger transmission of the Mailboxes 24 to 30.

If MCR2 is cleared, all messages for transmission are queued with respect to their priority (by running internal arbitration). The highest priority message has the Arbitration Field (STDID + IDE bit + EXTID (if IDE = 1) + RTR bit) with the lowest digital value and is transmitted first. The internal arbitration includes the RTR bit and the IDE bit (internal arbitration works in the same

way as the arbitration on the CAN Bus between two CAN nodes starting transmission at the same time).

This bit can be modified only in Reset or Halt mode.

Bit 2: MCR2	Description
0	Transmission order determined by message identifier priority (Initial value)
1	Transmission order determined by mailbox number priority (Mailbox-31 → Mailbox-1)

Bit 1—Halt Request (MCR1): Setting the MCR1 bit causes the CAN controller to complete its current operation and then enter Halt mode (where it is cut off from the CAN bus). The RCAN-TL1 remains in Halt Mode until the MCR1 is cleared. During the Halt mode, the CAN Interface does not join the CAN bus activity and does not store messages or transmit messages. All the user registers (including Mailbox contents and TEC/REC) remain unchanged with the exception of IRR0 and GSR4 which are used to notify the halt status itself. If the CAN bus is in idle or intermission state regardless of MCR6, RCAN-TL1 will enter Halt Mode within one Bit Time. If MCR6 is set, a halt request during Bus Off will be also processed within one Bit Time. Otherwise the full Bus Off recovery sequence will be performed beforehand. Entering the Halt Mode can be notified by IRR0 and GSR4.

If both MCR14 and MCR6 are set, MCR1 is automatically set as soon as RCAN-TL1 enters BusOff.

In the Halt mode, the RCAN-TL1 configuration can be modified with the exception of the Bit Timing setting, as it does not join the bus activity. MCR[1] has to be cleared by writing a ‘0’ in order to re-join the CAN bus. After this bit has been cleared, RCAN-TL1 waits until it detects 11 recessive bits, and then joins the CAN bus.

Notes: 1. After issuing a Halt request the CPU is not allowed to set TXPR or TXCR or clear MCR1 until the transition to Halt mode is completed (notified by IRR0 and GSR4). After MCR1 is set this can be cleared only after entering Halt mode or through a reset operation (SW or HW).
2. Transition into or recovery from HALT mode, is only possible if the BCR1 and BCR0 registers are configured to a proper Baud Rate.

Bit 1: MCR1	Description
0	Clear Halt request (Initial value)
1	Halt mode transition request

Bit 0 — Reset Request (MCR0): Controls resetting of the RCAN-TL1 module. When this bit is changed from ‘0’ to ‘1’ the RCAN-TL1 controller enters its reset routine, re-initialising the internal logic, which then sets GSR3 and IRR0 to notify the reset mode. During a re-initialisation, all user registers are initialised.

RCAN-TL1 can be re-configured while this bit is set. This bit has to be cleared by writing a ‘0’ to join the CAN bus. After this bit is cleared, the RCAN-TL1 module waits until it detects 11 recessive bits, and then joins the CAN bus. The Baud Rate needs to be set up to a proper value in order to sample the value on the CAN Bus.

After Power On Reset, this bit and GSR3 are always set. This means that a reset request has been made and RCAN-TL1 needs to be configured.

The Reset Request is equivalent to a Power On Reset but controlled by Software.

Bit 0: MCR0	Description
0	Clear Reset Request
1	CAN Interface reset mode transition request (Initial value)

(2) General Status Register (GSR)

The General Status Register (GSR) is a 16-bit read-only register that indicates the status of RCAN-TL1.

- GSR (Address = H'002)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	-	-	-	GSR5	GSR4	GSR3	GSR2	GSR1	GSR0
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
R/W:	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bits 15 to 6: Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bit 5 — Error Passive Status Bit (GSR5): Indicates whether the CAN Interface is in Error Passive or not. This bit will be set high as soon as the RCAN-TL1 enters the Error Passive state and is cleared when the module enters again the Error Active state (this means the GSR5 will stay high during Error Passive and during Bus Off). Consequently to find out the correct state both GSR5 and GSR0 must be considered.

Bit 5: GSR5	Description
0	RCAN-TL1 is not in Error Passive or in Bus Off status (Initial value) [Reset condition] RCAN-TL1 is in Error Active state
1	RCAN-TL1 is in Error Passive (if GSR0 = 0) or Bus Off (if GSR0 = 1) [Setting condition] When TEC ≥ 128 or REC ≥ 128 or if Error Passive Test Mode is selected

Bit 4 — Halt/Sleep Status Bit (GSR4): Indicates whether the CAN engine is in the halt/sleep state or not. Please note that the clearing time of this flag is not the same as the setting time of IRR12.

Please note that this flag reflects the status of the CAN engine and not of the full RCAN-TL1 IP. RCAN-TL1 exits sleep mode and can be accessed once MCR5 is cleared. The CAN engine exits sleep mode only after two additional transmission clocks on the CAN Bus.

Bit 4: GSR4	Description
0	RCAN-TL1 is not in the Halt state or Sleep state (Initial value)
1	Halt mode (if MCR1 = 1) or Sleep mode (if MCR5 = 1) [Setting condition] If MCR1 is set and the CAN bus is either in intermission or idle or MCR5 is set and RCAN-TL1 is in the halt mode or RCAN-TL1 is moving to Bus Off when MCR14 and MCR6 are both set

Bit 3 — Reset Status Bit (GSR3): Indicates whether the RCAN-TL1 is in the reset state or not.

Bit 3: GSR3	Description
0	RCAN-TL1 is not in the reset state
1	Reset state (Initial value) [Setting condition] After an RCAN-TL1 internal reset (due to SW or HW reset)

Bit 2 — Message Transmission in progress Flag (GSR2): Flag that indicates to the CPU if the RCAN-TL1 is in Bus Off or transmitting a message or an error/overload flag due to error detected during transmission. The timing to set TXACK is different from the time to clear GSR2. TXACK is set at the 7th bit of End Of Frame. GSR2 is set at the 3rd bit of intermission if there are no more messages ready to be transmitted. It is also set by arbitration lost, bus idle, reception, reset or halt transition.

Bit 2: GSR2	Description
0	RCAN-TL1 is in Bus Off or a transmission is in progress
1	[Setting condition] Not in Bus Off and no transmission in progress (Initial value)

Bit 1—Transmit/Receive Warning Flag (GSR1): Flag that indicates an error warning.

Bit 1: GSR1	Description
0	[Reset condition] When (TEC < 96 and REC < 96) or Bus Off (Initial value)
1	[Setting condition] When 96 ≤ TEC < 256 or 96 ≤ REC < 256

Note: REC is incremented during Bus Off to count the recurrences of 11 recessive bits as requested by the Bus Off recovery sequence. However the flag GSR1 is not set in Bus Off.

Bit 0—Bus Off Flag (GSR0): Flag that indicates that RCAN-TL1 is in the bus off state.

Bit 0: GSR0	Description
0	[Reset condition] Recovery from bus off state or after a HW or SW reset (Initial value)
1	[Setting condition] When TEC ≥ 256 (bus off state)

Note: Only the lower 8 bits of TEC are accessible from the user interface. The 9th bit is equivalent to GSR0.

(3) Bit Configuration Register (BCR0, BCR1)

The bit configuration registers (BCR0 and BCR1) are 2 X 16-bit read/write register that are used to set CAN bit timing parameters and the baud rate pre-scaler for the CAN Interface.

The Time quanta is defined as:

$$Timequanta = \frac{2 * BRP}{f_{clk}}$$

Where: BRP (Baud Rate Pre-scaler) is the value stored in BCR0 incremented by 1 and fclk is the used peripheral bus frequency.

- BCR1 (Address = H'004)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TSG1[3:0]				-	TSG2[2:0]			-	-	SJW[1:0]		-	-	-	BSP
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	R	R	R/W	R/W	R	R	R	R/W

Bits 15 to 12 — Time Segment 1 (TSG1[3:0] = BCR1[15:12]): These bits are used to set the segment TSEG1 (= PRSEG + PHSEG1) to compensate for edges on the CAN Bus with a positive phase error. A value from 4 to 16 time quanta can be set.

Bit 15: TSG1[3]	Bit 14: TSG1[2]	Bit 13: TSG1[1]	Bit 12: TSG1[0]	Description
0	0	0	0	Setting prohibited (Initial value)
0	0	0	1	Setting prohibited
0	0	1	0	Setting prohibited
0	0	1	1	PRSEG + PHSEG1 = 4 time quanta
0	1	0	0	PRSEG + PHSEG1 = 5 time quanta
:	:	:	:	:
:	:	:	:	:
1	1	1	1	PRSEG + PHSEG1 = 16 time quanta

Bit 11: Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bits 10 to 8 — Time Segment 2 (TSG2[2:0] = BCR1[10:8]): These bits are used to set the segment TSEG2 (= PHSEG2) to compensate for edges on the CAN Bus with a negative phase error. A value from 2 to 8 time quanta can be set as shown below.

Bit 10: TSG2[2]	Bit 9: TSG2[1]	Bit 8: TSG2[0]	Description
0	0	0	Setting prohibited (Initial value)
0	0	1	PHSEG2 = 2 time quanta (conditionally prohibited)
0	1	0	PHSEG2 = 3 time quanta
0	1	1	PHSEG2 = 4 time quanta
1	0	0	PHSEG2 = 5 time quanta
1	0	1	PHSEG2 = 6 time quanta
1	1	0	PHSEG2 = 7 time quanta
1	1	1	PHSEG2 = 8 time quanta

Bits 7 and 6: Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bits 5 and 4 - ReSynchronisation Jump Width (SJW[1:0] = BCR0[5:4]): These bits set the synchronisation jump width.

Bit 5: SJW[1]	Bit 4: SJW[0]	Description
0	0	Synchronisation Jump width = 1 time quantum (Initial value)
0	1	Synchronisation Jump width = 2 time quanta
1	0	Synchronisation Jump width = 3 time quanta
1	1	Synchronisation Jump width = 4 time quanta

Bits 3 to 1: Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bit 0 — Bit Sample Point (BSP = BCR1[0]): Sets the point at which data is sampled.

Bit 0 : BSP	Description
0	Bit sampling at one point (end of time segment 1) (Initial value)
1	Bit sampling at three points (rising edge of the last three clock cycles of PHSEG1)

- BCR0 (Address = H'006)

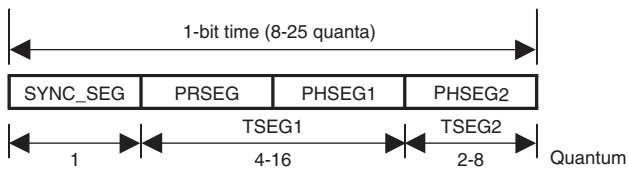
Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	-	BRP[7:0]							
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bits 8 to 15: Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bits 7 to 0—Baud Rate Pre-scale (BRP[7:0] = BCR0 [7:0]): These bits are used to define the peripheral bus clock periods contained in a Time Quantum.

Bit 7: BRP[7]	Bit 6: BRP[6]	Bit 5: BRP[5]	Bit 4: BRP[4]	Bit 3: BRP[3]	Bit 2: BRP[2]	Bit 1: BRP[1]	Bit 0: BRP[0]	Description
0	0	0	0	0	0	0	0	2 X peripheral bus clock (Initial value)
0	0	0	0	0	0	0	1	4 X peripheral bus clock
0	0	0	0	0	0	1	0	6 X peripheral bus clock
: :	: :	: :	: :	: :	: :	: :	: :	2*(register value + 1) X peripheral bus clock
1	1	1	1	1	1	1	1	512 X peripheral bus clock

- Requirements of Bit Configuration Register



SYNC_SEG: Segment for establishing synchronisation of nodes on the CAN bus. (Normal bit edge transitions occur in this segment.)

PRSEG: Segment for compensating for physical delay between networks.

PHSEG1: Buffer segment for correcting phase drift (positive). (This segment is extended when synchronisation (resynchronisation) is established.)

PHSEG2: Buffer segment for correcting phase drift (negative). (This segment is shortened when synchronisation (resynchronisation) is established)

TSEG1: TSG1 + 1

TSEG2: TSG2 + 1

The RCAN-TL1 Bit Rate Calculation is:

$$\text{Bit Rate} = \frac{f_{clk}}{2 \times (BRP + 1) \times (TSEG1 + TSEG2 + 1)}$$

Where BRP is given by the register value and TSEG1 and TSEG2 are derived values from TSG1 and TSG2 register values. The '+1' in the above formula is for the Sync-Seg which duration is 1 time quanta.

f_{CLK} = Peripheral Clock

BCR Setting Constraints

TSEG1min > TSEG2 ≥ SJWmax (SJW = 1 to 4)

8 ≤ TSEG1 + TSEG2 + 1 ≤ 25 time quanta (TSEG1 + TSEG2 + 1 = 7 is not allowed)

TSEG2 ≥ 2

These constraints allow the setting range shown in the table below for TSEG1 and TSEG2 in the Bit Configuration Register. The number in the table shows possible setting of SJW. "No" shows that there is no allowed combination of TSEG1 and TSEG2.

		001	010	011	100	101	110	111	TSG2
		2	3	4	5	6	7	8	TSEG2
TSG1	TSEG1								
0011	4	No	1-3	No	No	No	No	No	
0100	5	1-2	1-3	1-4	No	No	No	No	
0101	6	1-2	1-3	1-4	1-4	No	No	No	
0110	7	1-2	1-3	1-4	1-4	1-4	No	No	
0111	8	1-2	1-3	1-4	1-4	1-4	1-4	No	
1000	9	1-2	1-3	1-4	1-4	1-4	1-4	1-4	
1001	10	1-2	1-3	1-4	1-4	1-4	1-4	1-4	
1010	11	1-2	1-3	1-4	1-4	1-4	1-4	1-4	
1011	12	1-2	1-3	1-4	1-4	1-4	1-4	1-4	
1100	13	1-2	1-3	1-4	1-4	1-4	1-4	1-4	
1101	14	1-2	1-3	1-4	1-4	1-4	1-4	1-4	
1110	15	1-2	1-3	1-4	1-4	1-4	1-4	1-4	
1111	16	1-2	1-3	1-4	1-4	1-4	1-4	1-4	

Example 1: To have a Bit rate of 500Kbps with a frequency of fclk = 40MHz it is possible to set: BRP = 3, TSEG1 = 6, TSEG2 = 3.

Then the configuration to write is BCR1 = 5200 and BCR0 = 0003.

Example 2: To have a Bit rate of 250Kps with a frequency of 35MHz it is possible to set: BPR = 4, TSEG1 = 8, TSEG2 = 5.

Then the configuration to write is BCR1 = 7400 and BCR0 = 0004.

Example 3: To have a Bit rate of 500Kbps with a frequency of fclk = 32MHz it is possible to set: BRP = 1, TSEG1 = 11, TSEG2 = 4.

Then the configuration to write is BCR1 = A300 and BCR0 = 0001.

Example 4: To have a Bit rate of 500Kbps with a frequency of fclk = 20MHz it is possible to set: BRP = 1, TSEG1 = 6, TSEG2 = 3.

Then the configuration to write is BCR1 = 5200 and BCR0 = 0001.

(4) Interrupt Request Register (IRR)

The interrupt register (IRR) is a 16-bit read/write-clearable register containing status flags for the various interrupt sources.

- IRR (Address = H'008)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IRR15	IRR14	IRR13	IRR12	IRR11	IRR10	IRR9	IRR8	IRR7	IRR6	IRR5	IRR4	IRR3	IRR2	IRR1	IRR0
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W	R/W	R	R	R/W

Bit 15 — Timer Compare Match Interrupt 1 (IRR15): Indicates that a Compare-Match condition occurred to the Timer Compare Match Register 1 (TCMR1). When the value set in the TCMR1 matches to Cycle Time (TCMR1 = CYCTR), this bit is set.

Bit 15: IRR15	Description
0	Timer Compare Match has not occurred to the TCMR1 (Initial value) [Clearing condition] Writing 1
1	Timer Compare Match has occurred to the TCMR1 [Setting condition] TCMR1 matches to Cycle Time (TCMR1 = CYCTR)

Bit 14 — Timer Compare Match Interrupt 0 (IRR14): Indicates that a Compare-Match condition occurred to the Timer Compare Match Register 0 (TCMR0). When the value set in the TCMR0 matches to Local Time (TCMR0 = TCNTR), this bit is set.

Bit 14: IRR14	Description
0	Timer Compare Match has not occurred to the TCMR0 (Initial value) [Clearing condition] Writing 1
1	Timer Compare Match has occurred to the TCMR0 [Setting condition] TCMR0 matches to the Timer value (TCMR0 = TCNTR)

Bit 13 - Timer Overrun Interrupt/Next_is_Gap Reception Interrupt/Message Error Interrupt (IRR13): This interrupt assumes a different meaning depending on the RCAN-TL1 mode. It indicates that:

— The Timer (TCNTR) has overrun when RCAN-TL1 is working in event-trigger mode (including test modes)

— Time reference message with Next_is_Gap set has been received when working in time-trigger mode. Please note that when a Next_is_Gap is received the application is responsible to stop all transmission at the end of the current basic cycle (including test modes)

— Message error has occurred when in test mode. Note: If a Message Overload condition occurs when in Test Mode, then this bit will not be set.

Bit 13: IRR13	Description
0	Timer (TCNTR) has not overrun in event-trigger mode (including test modes) (Initial value)
	Time reference message with Next_is_Gap has not been received in time-trigger mode (including test modes)
	Message error has not occurred in test mode
	[Clearing condition] Writing 1
1	[Setting condition]
	Timer (TCNTR) has overrun and changed from H'FFFF to H'0000 in event-trigger mode (including test modes)
	Time reference message with Next_is_Gap has been received in time-trigger mode (including test modes)
	Message error has occurred in test mode

Bit 12 – Bus activity while in sleep mode (IRR12): IRR12 indicates that a CAN bus activity is present. While the RCAN-TL1 is in sleep mode and a dominant bit is detected on the CAN bus, this bit is set. This interrupt is cleared by writing a '1' to this bit position. Writing a '0' has no effect. If auto wakeup is not used and this interrupt is not requested it needs to be disabled by the related interrupt mask register. If auto wake up is not used and this interrupt is requested it should be cleared only after recovering from sleep mode. This is to avoid that a new falling edge of the reception line causes the interrupt to get set again.

Please note that the setting time of this interrupt is different from the clearing time of GSR4.

Bit 12: IRR12	Description
0	Bus idle state (Initial value)
	[Clearing condition] Writing 1
1	CAN bus activity detected in RCAN-TL1 sleep mode
	[Setting condition]
	Dominant bit level detection on the Rx line while in sleep mode

Bit 11 — Timer Compare Match Interrupt 2 (IRR11): Indicates that a Compare-Match condition occurred to the Timer Compare Match Register 2 (TCMR2). When the value set in the TCMR2 matches to Cycle Time (TCMR2 = CYCTR), this bit is set.

Bit 11: IRR11	Description
0	Timer Compare Match has not occurred to the TCMR2 (initial value) [Clearing condition] Writing 1
1	Timer Compare Match has occurred to the TCMR2 [Setting condition] TCMR2 matches to Cycle Time (TCMR2 = CYCTR)

Bit 10 — Start of new system matrix Interrupt (IRR10): Indicates that a new system matrix is starting.

When CCR = 0, this bit is set at the successful completion of reception/transmission of time reference message. Please note that when CMAX = 0 this interrupt is set at every basic cycle.

Bit 10: IRR10	Description
0	A new system matrix is not starting (initial value) [Clearing condition] Writing 1
1	Cycle counter reached zero. [Setting condition] Reception/transmission of time reference message is successfully completed when CMAX!= 3'b111 and CCR = 0

Bit 9 – Message Overrun/Overwrite Interrupt Flag (IRR9): Flag indicating that a message has been received but the existing message in the matching Mailbox has not been read as the corresponding RXPR or RFPR is already set to ‘1’ and not yet cleared by the CPU. The received message is either abandoned (overrun) or overwritten dependant upon the NMC (New Message Control) bit. This bit is cleared when all bit in UMSR (Unread Message Status Register) are cleared (by writing ‘1’) or by setting MBIMR (MailBox interrupt Mast Register) for all UMSR flag set. It is also cleared by writing a ‘1’ to all the correspondent bit position in MBIMR. Writing to this bit position has no effect.

Bit 9: IRR9	Description
0	No pending notification of message overrun/overwrite [Clearing condition] Clearing of all bit in UMSR/setting MBIMR for all UMSR set (initial value)
1	A receive message has been discarded due to overrun condition or a message has been overwritten [Setting condition] Message is received while the corresponding RXPR and/or RFPR = 1 and MBIMR = 0

Bit 8 - Mailbox Empty Interrupt Flag (IRR8): This bit is set when one of the messages set for transmission has been successfully sent (corresponding TXACK flag is set) or has been successfully aborted (corresponding ABACK flag is set). In Event Triggered mode the related TXPR is also cleared and this mailbox is now ready to accept a new message data for the next transmission. In Time Trigger mode TXPR for the Mailboxes from 30 to 24 is not cleared after a successful transmission in order to keep transmitting at each programmed basic cycle. In effect, this bit is set by an OR'ed signal of the TXACK and ABACK bits not masked by the corresponding MBIMR flag. Therefore, this bit is automatically cleared when all the TXACK and ABACK bits are cleared. It is also cleared by writing a '1' to all the correspondent bit position in MBIMR. Writing to this bit position has no effect.

Bit 8: IRR8	Description
0	Messages set for transmission or transmission cancellation request NOT progressed. (Initial value) [Clearing Condition] All the TXACK and ABACK bits are cleared/setting MBIMR for all TXACK and ABACK set
1	Message has been transmitted or aborted, and new message can be stored (in TT mode Mailbox 24 to 30 can be programmed with a new message only in case of abortion) [Setting condition] When a TXACK or ABACK bit is set (if related MBIMR = 0).

Bit 7 - Overload Frame (IRR7): Flag indicating that the RCAN-TL1 has detected a condition that should initiate the transmission of an overload frame. Note that in the condition of transmission being prevented, such as listen only mode, an Overload Frame will NOT be transmitted, but IRR7 will still be set. IRR7 remains asserted until reset by writing a '1' to this bit position - writing a '0' has no effect.

Bit 7: IRR7	Description
0	[Clearing condition] Writing 1 (Initial value)
1	[Setting conditions] Overload condition detected

Bit 6 - Bus Off Interrupt Flag (IRR6): This bit is set when RCAN-TL1 enters the Bus-off state or when RCAN-TL1 leaves Bus-off and returns to Error-Active. The cause therefore is the existing condition TEC ≥ 256 at the node or the end of the Bus-off recovery sequence (128X11 consecutive recessive bits) or the transition from Bus Off to Halt (automatic or manual). This bit remains set even if the RCAN-TL1 node leaves the bus-off condition, and needs to be explicitly cleared by S/W. The S/W is expected to read the GSR0 to judge whether RCAN-TL1 is in the bus-off or error active status. It is cleared by writing a '1' to this bit position even if the node is still bus-off. Writing a '0' has no effect.

Bit 6: IRR6	Description
0	[Clearing condition] Writing 1 (Initial value)
1	Enter Bus off state caused by transmit error or Error Active state returning from Bus-off [Setting condition] When TEC becomes ≥ 256 or End of Bus-off after 128X11 consecutive recessive bits or transition from Bus Off to Halt

Bit 5 - Error Passive Interrupt Flag (IRR5): Interrupt flag indicating the error passive state caused by the transmit or receive error counter or by Error Passive forced by test mode. This bit is reset by writing a '1' to this bit position, writing a '0' has no effect. If this bit is cleared the node may still be error passive. Please note that the SW needs to check GSR0 and GSR5 to judge whether RCAN-TL1 is in Error Passive or Bus Off status.

Bit 5: IRR5	Description
0	[Clearing condition] Writing 1 (Initial value)
1	Error passive state caused by transmit/receive error [Setting condition] When TEC ≥ 128 or REC ≥ 128 or Error Passive test mode is used

Bit 4 - Receive Error Counter Warning Interrupt Flag (IRR4): This bit becomes set if the receive error counter (REC) reaches a value greater than 95 when RCAN-TL1 is not in the Bus Off status. The interrupt is reset by writing a '1' to this bit position, writing '0' has no effect.

Bit 4: IRR4	Description
0	[Clearing condition] Writing 1 (Initial value)
1	Error warning state caused by receive error [Setting condition] When REC ≥ 96 and RCAN-TL1 is not in Bus Off

Bit 3 - Transmit Error Counter Warning Interrupt Flag (IRR3): This bit becomes set if the transmit error counter (TEC) reaches a value greater than 95. The interrupt is reset by writing a '1' to this bit position, writing '0' has no effect.

Bit 3: IRR3	Description
0	[Clearing condition] Writing 1 (Initial value)
1	Error warning state caused by transmit error [Setting condition] When TEC ≥ 96

Bit 2 - Remote Frame Receive Interrupt Flag (IRR2): Flag indicating that a remote frame has been received in a mailbox. This bit is set if at least one receive mailbox, with related MBIMR not set, contains a remote frame transmission request. This bit is automatically cleared when all bits in the Remote Frame Receive Pending Register (RFPR), are cleared. It is also cleared by writing a '1' to all the correspondent bit position in MBIMR. Writing to this bit has no effect.

Bit 2: IRR2	Description
0	[Clearing condition] Clearing of all bits in RFPR (Initial value)
1	At least one remote request is pending [Setting condition] When remote frame is received and the corresponding MBIMR = 0

Bit 1 – Data Frame Received Interrupt Flag (IRR1): IRR1 indicates that there are pending Data Frames received. If this bit is set at least one receive mailbox contains a pending message. This bit is cleared when all bits in the Data Frame Receive Pending Register (RXPR) are cleared, i.e. there is no pending message in any receiving mailbox. It is in effect a logical OR of the RXPR flags from each configured receive mailbox with related MBIMR not set. It is also cleared by writing a '1' to all the correspondent bit position in MBIMR. Writing to this bit has no effect.

Bit 1: IRR1	Description
0	[Clearing condition] Clearing of all bits in RXPR (Initial value)
1	Data frame received and stored in Mailbox [Setting condition] When data is received and the corresponding MBIMR = 0

Bit 0 – Reset/Halt/Sleep Interrupt Flag (IRR0): This flag can get set for three different reasons. It can indicate that:

1. Reset mode has been entered after a SW (MCR0) or HW reset
2. Halt mode has been entered after a Halt request (MCR1)
3. Sleep mode has been entered after a sleep request (MCR5) has been made while in Halt mode.

The GSR may be read after this bit is set to determine which state RCAN-TL1 is in.

Important: When a Sleep mode request needs to be made, the Halt mode must be used beforehand. Please refer to the MCR5 description and Figure 14.15.

IRR0 is set by the transition from "0" to "1" of GSR3 or GSR4 or by transition from Halt mode to Sleep mode. So, IRR0 is not set if RCAN-TL1 enters Halt mode again right after exiting from Halt mode, without GSR4 being cleared. Similarly, IRR0 is not set by direct transition from Sleep mode to Halt Request. At the transition from Halt/Sleep mode to Transition/Reception, clearing GSR4 needs (one-bit time - TSEG2) to (one-bit time * 2 - TSEG2).

In the case of Reset mode, IRR0 is set, however, the interrupt to the CPU is not asserted since IMR0 is automatically set by initialisation.

Bit 0: IRR0	Description
0	[Clearing condition] Writing 1
1	Transition to S/W reset mode or transition to halt mode or transition to sleep mode (Initial value) [Setting condition] When reset/halt/sleep transition is completed after a reset (MCR0 or HW) or Halt mode (MCR1) or Sleep mode (MCR5) is requested

(5) Interrupt Mask Register (IMR)

The interrupt mask register is a 16 bit register that protects all corresponding interrupts in the Interrupt Request Register (IRR) from generating an output signal on the IRQ. An interrupt request is masked if the corresponding bit position is set to '1'. This register can be read or written at any time. The IMR directly controls the generation of IRQ, but does not prevent the setting of the corresponding bit in the IRR.

- IMR (Address = H'00A)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IMR15	IMR14	IMR13	IMR12	IMR11	IMR10	IMR9	IMR8	IMR7	IMR6	IMR5	IMR4	IMR3	IMR2	IMR1	IMR0
Initial value:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 15 to 0: Maskable interrupt sources corresponding to IRR[15:0] respectively. When a bit is set, the interrupt signal is not generated, although setting the corresponding IRR bit is still performed.

Bit[15:0]: IMRn	Description
0	Corresponding IRR is not masked (IRQ is generated for interrupt conditions)
1	Corresponding interrupt of IRR is masked (Initial value)

(6) Transmit Error Counter (TEC) and Receive Error Counter (REC)

The Transmit Error Counter (TEC) and Receive Error Counter (REC) is a 16-bit read/(write) register that functions as a counter indicating the number of transmit/receive message errors on the CAN Interface. The count value is stipulated in the CAN protocol specification Refs. [1], [2], [3] and [4]. When not in (Write Error Counter) test mode this register is read only, and can only be modified by the CAN Interface. This register can be cleared by a Reset request (MCR0) or entering to bus off.

In Write Error Counter test mode (i.e. TST[2:0] = 3'b100), it is possible to write to this register. The same value can only be written to TEC/REC, and the value written into TEC is set to TEC and REC. When writing to this register, RCAN-TL1 needs to be put into Halt Mode. This feature is only intended for test purposes.

- TEC/REC (Address = H'00C)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*

Note: * It is only possible to write the value in test mode when TST[2:0] in MCR is 3'b100.
REC is incremented during Bus Off to count the recurrences of 11 recessive bits as requested by the Bus Off recovery sequence.

14.3.4 RCAN-TL1 Mailbox Registers

The following sections describe RCAN-TL1 Mailbox registers that control/flag individual Mailboxes. The address is mapped as follows.

Important: LongWord access is carried out as two consecutive Word accesses.

32-Mailboxes version

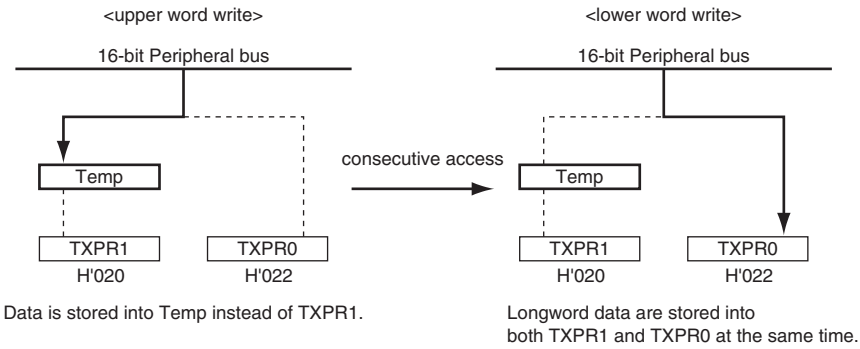
Description	Address	Name	Access Size (bits)
Transmit Pending 1	020	TXPR1	LW
Transmit Pending 0	022	TXPR0	—
	024		
	026		
Transmit Cancel 1	028	TXCR1	Word/LW
Transmit Cancel 0	02A	TXCR0	Word
	02C		
	02E		
Transmit Acknowledge 1	030	TXACK1	Word/LW
Transmit Acknowledge 0	032	TXACK0	Word
	034		
	036		
Abort Acknowledge 1	038	ABACK1	Word/LW
Abort Acknowledge 0	03A	ABACK0	Word
	03C		
	03E		
Data Frame Receive Pending 1	040	RXPR1	Word/LW
Data Frame Receive Pending 0	042	RXPR0	Word
	044		
	046		
Remote Frame Receive Pending 1	048	RFPR1	Word/LW
Remote Frame Receive Pending 0	04A	RFPR0	Word
	04C		
	04E		
Mailbox Interrupt Mask Register 1	050	MBIMR1	Word/LW
Mailbox Interrupt Mask Register 0	052	MBIMR0	Word
	054		
	056		
Unread message Status Register 1	058	UMSR1	Word/LW
Unread message Status Register 0	05A	UMSR0	Word
	05C		
	05E		

Figure 14.11 RCAN-TL1 Mailbox registers

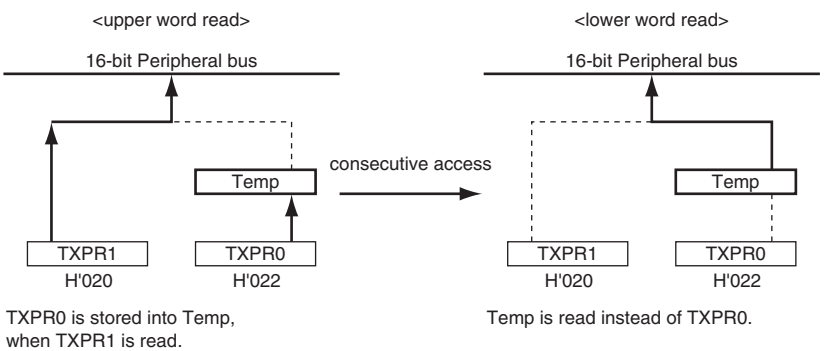
(1) Transmit Pending Register (TXPR1, TXPR0)

The concatenation of TXPR1 and TXPR0 is a 32-bit register that contains any transmit pending flags for the CAN module. In the case of 16-bit bus interface, Long Word access is carried out as two consecutive word accesses.

<Longword Write Operation>



<Longword Read Operation>



The TXPR1 controls Mailbox-31 to Mailbox-16, and the TXPR0 controls Mailbox-15 to Mailbox-1. The CPU may set the TXPR bits to affect any message being considered for transmission by writing a ‘1’ to the corresponding bit location. Writing a ‘0’ has no effect, and TXPR cannot be cleared by writing a ‘0’ and must be cleared by setting the corresponding TXCR bits. TXPR may be read by the CPU to determine which, if any, transmissions are pending or in progress. In effect there is a transmit pending bit for all Mailboxes except for the Mailbox-0. Writing a ‘1’ to a bit location when the mailbox is not configured to transmit is not allowed.

In Event Triggered Mode RCAN-TL1 will clear a transmit pending flag after successful transmission of its corresponding message or when a transmission abort is requested successfully from the TXCR. In Time Trigger Mode, TXPR for the Mailboxes from 30 to 24 is NOT cleared after a successful transmission, in order to keep transmitting at each programmed basic cycle. The TXPR flag is not cleared if the message is not transmitted due to the CAN node losing the arbitration process or due to errors on the CAN bus, and RCAN-TL1 automatically tries to transmit it again unless its DART bit (Disable Automatic Re-Transmission) is set in the Message-Control of the corresponding Mailbox. In such case (DART set), the transmission is cleared and notified through Mailbox Empty Interrupt Flag (IRR8) and the correspondent bit within the Abort Acknowledgement Register (ABACK).

If the status of the TXPR changes, the RCAN-TL1 shall ensure that in the identifier priority scheme (MCR2 = 0), the highest priority message is always presented for transmission in an intelligent way even under circumstances such as bus arbitration losses or errors on the CAN bus. Please refer to the Application Note for details.

When the RCAN-TL1 changes the state of any TXPR bit position to a '0', an empty slot interrupt (IRR8) may be generated. This indicates that either a successful or an aborted mailbox transmission has just been made. If a message transmission is successful it is signalled in the TXACK register, and if a message transmission abortion is successful it is signalled in the ABACK register. By checking these registers, the contents of the Message of the corresponding Mailbox may be modified to prepare for the next transmission.

- TXPR1

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TXPR1[15:0]															
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*

Note: * It is possible only to write a ‘1’ for a Mailbox configured as transmitter.

Bit 15 to 0 — Requests the corresponding Mailbox to transmit a CAN Frame. The bit 15 to 0 corresponds to Mailbox-31 to 16 respectively. When multiple bits are set, the order of the transmissions is governed by the MCR2 – CAN-ID or Mailbox number.

Bit[15:0]: TXPR1	Description
0	Transmit message idle state in corresponding mailbox (Initial value) [Clearing Condition] Completion of message transmission (for Event Triggered Messages) or message transmission abortion (automatically cleared)
1	Transmission request made for corresponding mailbox

- TXPR0

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TXPR0[15:1]															-
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R

Note: * It is possible only to write a ‘1’ for a Mailbox configured as transmitter.

Bit 15 to 1 — Indicates that the corresponding Mailbox is requested to transmit a CAN Frame. The bit 15 to 1 corresponds to Mailbox-15 to 1 respectively. When multiple bits are set, the order of the transmissions is governed by the MCR2 – CAN-ID or Mailbox number.

Bit[15:1]: TXPR0	Description
0	Transmit message idle state in corresponding mailbox (Initial value) [Clearing Condition] Completion of message transmission (for Event Triggered Messages) or message transmission abortion (automatically cleared)
1	Transmission request made for corresponding mailbox

Bit 0— Reserved: This bit is always ‘0’ as this is a receive-only Mailbox. Writing a ‘1’ to this bit position has no effect. The returned value is ‘0’.

(2) Transmit Cancel Register (TXCR1, TXCR0)

The TXCR1 and TXCR0 are 16-bit read/conditionally-write registers. The TXCR1 controls Mailbox-31 to Mailbox-16, and the TXCR0 controls Mailbox-15 to Mailbox-1.This register is used by the CPU to request the pending transmission requests in the TXPR to be cancelled. To clear the corresponding bit in the TXPR the CPU must write a '1' to the bit position in the TXCR. Writing a '0' has no effect.

When an abort has succeeded the CAN controller clears the corresponding TXPR + TXCR bits, and sets the corresponding ABACK bit. However, once a Mailbox has started a transmission, it cannot be cancelled by this bit. In such a case, if the transmission finishes in success, the CAN controller clears the corresponding TXPR + TXCR bit, and sets the corresponding TXACK bit, however, if the transmission fails due to a bus arbitration loss or an error on the bus, the CAN controller clears the corresponding TXPR + TXCR bit, and sets the corresponding ABACK bit. If an attempt is made by the CPU to clear a mailbox transmission that is not transmit-pending it has no effect. In this case the CPU will be not able at all to set the TXCR flag.

- TXCR1

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TXCR1[15:0]															
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*

Note: * Only writing a '1' to a Mailbox that is requested for transmission and is configured as transmit.

Bit 15 to 0 — Requests the corresponding Mailbox, that is in the queue for transmission, to cancel its transmission. The bit 15 to 0 corresponds to Mailbox-31 to 16 (and TXPR1[15:0]) respectively.

Bit[15:0]:TXCR1	Description
0	Transmit message cancellation idle state in corresponding mailbox (Initial value) [Clearing Condition] Completion of transmit message cancellation (automatically cleared)
1	Transmission cancellation request made for corresponding mailbox

- TXCR0

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TXCR0[15:1]															-
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R

Note: * Only writing a ‘1’ to a Mailbox that is requested for transmission and is configured as transmit.

Bit 15 to 1 — Requests the corresponding Mailbox, that is in the queue for transmission, to cancel its transmission. The bit 15 to 1 corresponds to Mailbox-15 to 1 (and TXPR0[15:1]) respectively.

Bit[15:1]: TXCR0	Description
0	Transmit message cancellation idle state in corresponding mailbox (Initial value)
	[Clearing Condition]
	Completion of transmit message cancellation (automatically cleared)
1	Transmission cancellation request made for corresponding mailbox

Bit 0 — This bit is always ‘0’ as this is a receive-only mailbox. Writing a ‘1’ to this bit position has no effect and always read back as a ‘0’.

(3) Transmit Acknowledge Register (TXACK1, TXACK0)

The TXACK1 and TXACK0 are 16-bit read/conditionally-write registers. These registers are used to signal to the CPU that a mailbox transmission has been successfully made. When a transmission has succeeded the RCAN-TL1 sets the corresponding bit in the TXACK register. The CPU may clear a TXACK bit by writing a ‘1’ to the corresponding bit location. Writing a ‘0’ has no effect.

- TXACK1

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TXACK1[15:0]															
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*

Note: * Only when writing a ‘1’ to clear.

Bit 15 to 0 — Notifies that the requested transmission of the corresponding Mailbox has been finished successfully. The bit 15 to 0 corresponds to Mailbox-31 to 16 respectively.

Bit[15:0]:TXACK1	Description
0	[Clearing Condition] Writing ‘1’ (Initial value)
1	Corresponding Mailbox has successfully transmitted message (Data or Remote Frame) [Setting Condition] Completion of message transmission for corresponding mailbox

- TXACK0

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TXACK0[15:1]															-
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	-

Note: * Only when writing a ‘1’ to clear.

Bit 15 to 1 — Notifies that the requested transmission of the corresponding Mailbox has been finished successfully. The bit 15 to 1 corresponds to Mailbox-15 to 1 respectively.

Bit[15:1]:TXACK0	Description
0	[Clearing Condition] Writing ‘1’ (Initial value)
1	Corresponding Mailbox has successfully transmitted message (Data or Remote Frame) [Setting Condition] Completion of message transmission for corresponding mailbox

Bit 0 — This bit is always ‘0’ as this is a receive-only mailbox. Writing a ‘1’ to this bit position has no effect and always read back as a ‘0’.

(4) Abort Acknowledge Register (ABACK1, ABACK0)

The ABACK1 and ABACK0 are 16-bit read/conditionally-write registers. These registers are used to signal to the CPU that a mailbox transmission has been aborted as per its request. When an abort has succeeded the RCAN-TL1 sets the corresponding bit in the ABACK register. The CPU may clear the Abort Acknowledge bit by writing a '1' to the corresponding bit location. Writing a '0' has no effect. An ABACK bit position is set by the RCAN-TL1 to acknowledge that a TXPR bit has been cleared by the corresponding TXCR bit.

- ABACK1

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ABACK1[15:0]															
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*

Note: * Only when writing a '1' to clear.

Bit 15 to 0 — Notifies that the requested transmission cancellation of the corresponding Mailbox has been performed successfully. The bit 15 to 0 corresponds to Mailbox-31 to 16 respectively.

Bit[15:0]:ABACK1	Description
0	[Clearing Condition] Writing '1' (Initial value)
1	Corresponding Mailbox has cancelled transmission of message (Data or Remote Frame) [Setting Condition] Completion of transmission cancellation for corresponding mailbox

- ABACK0

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ABACK0[15:1]															-
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R

Note: * Only when writing a '1' to clear.

Bit 15 to 1 — Notifies that the requested transmission cancellation of the corresponding Mailbox has been performed successfully. The bit 15 to 1 corresponds to Mailbox-15 to 1 respectively.

Bit[15:1]:ABACK0	Description
0	[Clearing Condition] Writing ‘1’ (Initial value)
1	Corresponding Mailbox has cancelled transmission of message (Data or Remote Frame) [Setting Condition] Completion of transmission cancellation for corresponding mailbox

Bit 0 — This bit is always ‘0’ as this is a receive-only mailbox. Writing a ‘1’ to this bit position has no effect and always read back as a ‘0’.

(5) Data Frame Receive Pending Register (RXPR1, RXPR0)

The RXPR1 and RXPR0 are 16-bit read/conditionally-write registers. The RXPR is a register that contains the received Data Frames pending flags associated with the configured Receive Mailboxes. When a CAN Data Frame is successfully stored in a receive mailbox the corresponding bit is set in the RXPR. The bit may be cleared by writing a ‘1’ to the corresponding bit position. Writing a ‘0’ has no effect. However, the bit may only be set if the mailbox is configured by its MBC (Mailbox Configuration) to receive Data Frames. When a RXPR bit is set, it also sets IRR1 (Data Frame Received Interrupt Flag) if its MBIMR (Mailbox Interrupt Mask Register) is not set, and the interrupt signal is generated if IMR1 is not set. Please note that these bits are only set by receiving Data Frames and not by receiving Remote frames.

- RXPR1

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RXPR1[15:0]															
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*

Note : * Only when writing a ‘1’ to clear.

Bit 15 to 0 — Configurable receive mailbox locations corresponding to each mailbox position from 31 to 16 respectively.

Bit[15:0]: RXPR1	Description
0	[Clearing Condition] Writing ‘1’ (Initial value)
1	Corresponding Mailbox received a CAN Data Frame [Setting Condition] Completion of Data Frame receive on corresponding mailbox

- RXPR0

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RXPR0[15:0]															
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*

Note: * Only when writing a ‘1’ to clear.

Bit 15 to 0 — Configurable receive mailbox locations corresponding to each mailbox position from 15 to 0 respectively.

Bit[15:0]: RXPR0	Description
0	[Clearing Condition] Writing ‘1’ (Initial value)
1	Corresponding Mailbox received a CAN Data Frame [Setting Condition] Completion of Data Frame receive on corresponding mailbox

(6) Remote Frame Receive Pending Register (RFPR1, RFPR0)

The RFPR1 and RFPR0 are 16-bit read/conditionally-write registers. The RFPR is a register that contains the received Remote Frame pending flags associated with the configured Receive Mailboxes. When a CAN Remote Frame is successfully stored in a receive mailbox the corresponding bit is set in the RFPR. The bit may be cleared by writing a ‘1’ to the corresponding bit position. Writing a ‘0’ has no effect. In effect there is a bit position for all mailboxes. However, the bit may only be set if the mailbox is configured by its MBC (Mailbox Configuration) to receive Remote Frames. When a RFPR bit is set, it also sets IRR2 (Remote Frame Receive Interrupt Flag) if its MBIMR (Mailbox Interrupt Mask Register) is not set, and the interrupt signal is generated if IMR2 is not set. Please note that these bits are only set by receiving Remote Frames and not by receiving Data frames.

- RFPR1

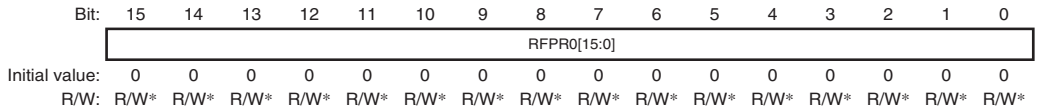
Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RFPR1[15:0]															
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*

Note: * Only when writing a ‘1’ to clear.

Bit 15 to 0 — Remote Request pending flags for mailboxes 31 to 16 respectively.

Bit[15:0]: RFPR1	Description
0	[Clearing Condition] Writing ‘1’ (Initial value)
1	Corresponding Mailbox received Remote Frame [Setting Condition] Completion of remote frame receive in corresponding mailbox

- RFPR0



Note: * Only when writing a ‘1’ to clear.

Bit 15 to 0 — Remote Request pending flags for mailboxes 15 to 0 respectively.

Bit[15:0]: RFPR0	Description
0	[Clearing Condition] Writing ‘1’ (Initial value)
1	Corresponding Mailbox received Remote Frame [Setting Condition] Completion of remote frame receive in corresponding mailbox

(7) Mailbox Interrupt Mask Register (MBIMR)

The MBIMR1 and MBIMR0 are 16-bit read/write registers. The MBIMR only prevents the setting of IRR related to the Mailbox activities, that are IRR[1] – Data Frame Received Interrupt, IRR[2] – Remote Frame Receive Interrupt, IRR[8] – Mailbox Empty Interrupt, and IRR[9] – Message OverRun/OverWrite Interrupt. If a mailbox is configured as receive, a mask at the corresponding bit position prevents the generation of a receive interrupt (IRR[1] and IRR[2] and IRR[9]) but does not prevent the setting of the corresponding bit in the RXPR or RFPR or UMSR. Similarly when a mailbox has been configured for transmission, a mask prevents the generation of an Interrupt signal and setting of an Mailbox Empty Interrupt due to successful transmission or abortion of transmission (IRR[8]), however, it does not prevent the RCAN-TL1 from clearing the corresponding TXPR/TXCR bit + setting the TXACK bit for successful transmission, and it does not prevent the RCAN-TL1 from clearing the corresponding TXPR/TXCR bit + setting the ABACK bit for abortion of the transmission.

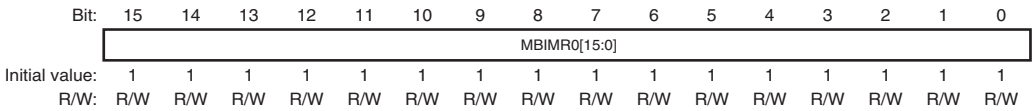
A mask is set by writing a ‘1’ to the corresponding bit position for the mailbox activity to be masked. At reset all mailbox interrupts are masked.

- MBIMR1

Bit 15 to 0 — Enable or disable interrupt requests from individual Mailbox-31 to Mailbox-16 respectively.

Bit[15:0]: MBIMR1	Description
0	Interrupt Request from IRR1/IRR2/IRR8/IRR9 enabled
1	Interrupt Request from IRR1/IRR2/IRR8/IRR9 disabled (initial value)

- MBIMR0



Bit 15 to 0 — Enable or disable interrupt requests from individual Mailbox-15 to Mailbox-0 respectively.

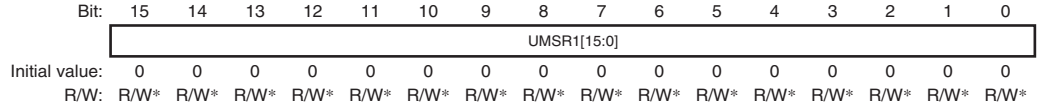
Bit[15:0]: MBIMR0	Description
0	Interrupt Request from IRR1/IRR2/IRR8/IRR9 enabled
1	Interrupt Request from IRR1/IRR2/IRR8/IRR9 disabled (initial value)

(8) Unread Message Status Register (UMSR)

This register is a 32-bit read/conditionally write register and it records the mailboxes whose contents have not been accessed by the CPU prior to a new message being received. If the CPU has not cleared the corresponding bit in the RXPR or RFPR when a new message for that mailbox is received, the corresponding UMSR bit is set to ‘1’. This bit may be cleared by writing a ‘1’ to the corresponding bit location in the UMSR. Writing a ‘0’ has no effect.

If a mailbox is configured as transmit box, the corresponding UMSR will not be set.

- UMSR1

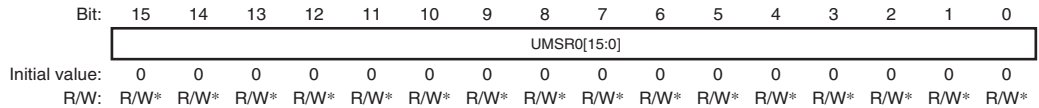


Note: * Only when writing a ‘1’ to clear.

Bit 15 to 0 — Indicate that an unread received message has been overwritten or overrun condition has occurred for Mailboxes 31 to 16.

Bit[15:0]: UMSR1	Description
0	[Clearing Condition] Writing ‘1’ (initial value)
1	Unread received message is overwritten by a new message or overrun condition [Setting Condition] When a new message is received before RXPR or RFPR is cleared

- UMSR0



Note: * Only when writing a ‘1’ to clear.

Bit 15 to 0 — Indicate that an unread received message has been overwritten or overrun condition has occurred for Mailboxes 15 to 0.

Bit[15:0]: UMSR0	Description
0	[Clearing Condition] Writing ‘1’ (initial value)
1	Unread received message is overwritten by a new message or overrun condition [Setting Condition] When a new message is received before RXPR or RFPR is cleared

14.3.5 Timer Registers

The Timer is 16 bits and supports several source clocks. A pre-scale counter can be used to reduce the speed of the clock. It also supports three Compare Match Registers (TCMR2, TCMR1, TCMR0). The address map is as follows.

Important: These registers can only be accessed in Word size (16-bit).

Description	Address	Name	Access Size (bits)
Timer Trigger Control Register 0	080	TTCR0	Word (16)
Cycle Maximum/Tx-Enable Window Register	084	CMAX_TEW	Word (16)
Reference Trigger Offset Register	086	RFTROFF	Word (16)
Timer Status Register	088	TSR	Word (16)
Cycle Counter Register	08A	CCR	Word (16)
Timer Counter Register	08C	TCNTR	Word (16)
Cycle Time Register	090	CYCTR	Word (16)
Reference Mark Register	094	RFMK	Word (16)
Timer Compare Match Register 0	098	TCMR0	Word (16)
Timer Compare Match Register 1	09C	TCMR1	Word (16)
Timer Compare Match Register 2	0A0	TCMR2	Word (16)
Tx-Trigger Time Selection Register	0A4	TTTSEL	Word (16)

Figure 14.12 RCAN-TL1 Timer registers

(1) Time Trigger Control Register0 (TTCR0)

The Time Trigger Control Register0 is a 16-bit read/write register and provides functions to control the operation of the Timer. When operating in Time Trigger Mode, please refer to section 14.4.3 (1), Time Triggered Transmission.

- TTCR0 (Address = H'080)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TCR15	TCR14	TCR13	TCR12	TCR11	TCR10	-	-	-	TCR6	TPSC5	TPSC4	TPSC3	TPSC2	TPSC1	TPSC0
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 15 — Enable Timer: When this bit is set, the timer TCNTR is running. When this bit is cleared, TCNTR and CCR are cleared.

Bit15: TTCR0 15	Description
0	Timer and CCR are cleared and disabled (initial value)
1	Timer is running

Bit 14 — TimeStamp value: Specifies if the Timestamp for transmission and reception in Mailboxes 15 to 1 must contain the Cycle Time (CYCTR) or the concatenation of CCR[5:0] + CYCTR[15:6]. This feature is very useful for time triggered transmission to monitor Rx_Trigger.

This register does not affect the TimeStamp for Mailboxes 30 and 31.

Bit14: TTCR0 14	Description
0	CYCTR[15:0] is used for the TimeStamp in Mailboxes 15 to 1 (initial value)
1	CCR[5:0] + CYCTR[15:6] is used for the TimeStamp in Mailboxes 15 to 1

Bit 13 — Cancellation by TCMR2: The messages in the transmission queue are cancelled by setting TXCR, when both this bit and bit12 are set and compare match occurs when RCAN-TL1 is not in the Halt status, causing the setting of all TXCR bits with the corresponding TXPR bits set.

Bit13: TTCR0 13	Description
0	Cancellation by TCMR2 compare match is disabled (initial value)
1	Cancellation by TCMR2 compare match is enabled

Bit 12 — TCMR2 compare match enable: When this bit is set, IRR11 is set by TCMR2 compare match.

Bit12 TTCR0 12	Description
0	IRR11 isn't set by TCMR2 compare match (initial value)
1	IRR11 is set by TCMR2 compare match

Bit 11 — TCMR1 compare match enable: When this bit is set, IRR15 is set by TCMR1 compare match.

Bit11 TTCR0 11	Description
0	IRR15 isn't set by TCMR1 compare match (initial value)
1	IRR15 is set by TCMR1 compare match

Bit 10 — TCMR0 compare match enable: When this bit is set, IRR14 is set by TCMR0 compare match.

Bit10 TTCR0 10	Description
0	IRR14 isn't set by TCMR0 compare match (initial value)
1	IRR14 is set by TCMR0 compare match

Bits 9 to 7: Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bit 6 — Timer Clear-Set Control by TCMR0: Specifies if the Timer is to be cleared and set to H'0000 when the TCMR0 matches to the TCNTR. Please note that the TCMR0 is also capable to generate an interrupt signal to the CPU via IRR14.

Note: If RCAN-TL1 is working in TTCAN mode (CMAX isn't 3'b111), TTCR0 bit6 has to be ‘0’ to avoid clearing Local Time.

Bit6: TTCR0 6	Description
0	Timer is not cleared by the TCMR0 (initial value)
1	Timer is cleared by the TCMR0

Bit5 to 0 — RCAN-TL1 Timer Prescaler (TPSC[5:0]): This control field allows the timer source clock (4*[RCAN-TL1 system clock]) to be divided before it is used for the timer. This function is available only in event-trigger mode. In time trigger mode (CMAX is not 3'b111), one nominal Bit Timing (= one bit length of CAN bus) is automatically chosen as source clock of TCNTR.

The following relationship exists between source clock period and the timer period.

Bit[5:0]: TPSC[5:0]	Description
0 0 0 0 0 0	1 X Source Clock (initial value)
0 0 0 0 0 1	2 X Source Clock
0 0 0 0 1 0	3 X Source Clock
0 0 0 0 1 1	4 X Source Clock
0 0 0 1 0 0	5 X Source Clock
. . . . .	. . . . .
. . . . .	. . . . .
1 1 1 1 1 1	64 X Source Clock

(2) Cycle Maximum/Tx-Enable Window Register (CMAX_TEW)

This register is a 16-bit read/write register. CMAX specifies the maximum value for the cycle counter (CCR) for TT Transmissions to set the number of basic cycles in the matrix system. When the Cycle Counter reaches the maximum value (CCR = CMAX), after a full basic cycle, it is cleared to zero and an interrupt is generated on IRR.10.

TEW specifies the width of Tx-Enable window.

- CMAX_TEW (Address = H'084)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-	-	-	-	-	CMAX[2:0]			-	-	-	-	TEW[3:0]			
Initial value:	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R/W	R/W	R/W	R	R	R	R	R/W	R/W	R/W	R/W

Bits 15 to 11: Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bit 10 to 8 — Cycle Count Maximum (CMAX): Indicates the maximum number of CCR. The number of basic cycles available in the matrix cycle for Timer Triggered transmission is (Cycle Count Maximum + 1).

Unless CMAX = 3'b111, RCAN-TL1 is in time-trigger mode and time trigger function is available. If CMAX = 3'b111, RCAN-TL1 is in event-trigger mode.

Bit[10:8]: CMAX[2:0]	Description
0 0 0	Cycle Count Maximum = 0
0 0 1	Cycle Count Maximum = 1
0 1 0	Cycle Count Maximum = 3
0 1 1	Cycle Count Maximum = 7
1 0 0	Cycle Count Maximum = 15
1 0 1	Cycle Count Maximum = 31
1 1 0	Cycle Count Maximum = 63
1 1 1	CCR is cleared and RCAN-TL1 is in event-trigger mode. (initial value)

Important: Please set CMAX = 3'b111 when event-trigger mode is used.

Bits 7 to 4: Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bit 3 to 0 — Tx-Enable Window (TEW): Indicates the width of Tx-Enable Window. TEW = H'00 shows the width is one nominal Bit Timing. All values from 0 to 15 are allowed to be set.

Bit[3:0]: TEW[3:0]	Description
0 0 0 0	The width of Tx-Enable Window = 1 (initial value)
0 0 0 1	The width of Tx-Enable Window = 2
0 0 1 0	The width of Tx-Enable Window = 3
0 0 1 1	The width of Tx-Enable Window = 4
. . . .	. . . . .
. . . .	. . . . .
1 1 1 1	The width of Tx-Enable Window = 16

Note: The CAN core always needs a time between 1 to 2 bit timing to initiate transmission. The above values are not considering this accuracy.

(3) Reference Trigger Offset Register (RFTROFF)

This is a 8-bit read/write register that affects Tx-Trigger Time (TTT) of Mailbox-30. The TTT of Mailbox-30 is compared with CYCTR after RFTROFF extended with sign is added to the TTT. However, the value of TTT is not modified. The offset value doesn't affect others except Mailbox-30.

- RFTROFF (Address = H'086)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RFTROFF[7:0]								-	-	-	-	-	-	-	-
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R

Bit 15 to 8 — Indicate the value of Reference Trigger Offset.

Bits 7 to 0: Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Description
0	0	0	0	0	0	0	0	Ref_trigger_offset = +0 (initial value)
0	0	0	0	0	0	0	1	Ref_trigger_offset = +1
0	0	0	0	0	0	1	0	Ref_trigger_offset = +2
.	.	.	.	.	.	.	.	
0	1	1	1	1	1	1	1	Ref_trigger_offset = +127
.	.	.	.	.	.	.	.	
1	1	1	1	1	1	1	1	Ref_trigger_offset = –1
1	1	1	1	1	1	1	0	Ref_trigger_offset = –2
.	.	.	.	.	.	.	.	
1	0	0	0	0	0	0	1	Ref_trigger_offset = –127
1	0	0	0	0	0	0	0	Prohibited

(4) Timer Status Register (TSR)

This register is a 16-bit read-only register, and allows the CPU to monitor the Timer Compare Match status and the Timer Overrun Status.

- TSR (Address = H'088)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	-	-	-	-	TSR4	TSR3	TSR2	TSR1	TSR0
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bits 15 to 5: Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bit 4 to 0 — RCAN-TL1 Timer Status (TSR[4:0]): This read-only field allows the CPU to monitor the status of the Cycle Counter, the Timer and the Compare Match registers. Writing to this field has no effect.

Bit 4 — Start of New System Matrix (TSR4): Indicates that a new system matrix is starting. When CCR = 0, this bit is set at the successful completion of reception/transmission of time reference message.

Bit4: TSR4	Description
0	A new system matrix is not starting (initial value) [Clearing condition] Writing ‘1’ to IRR10 (Cycle Counter Overflow Interrupt)
1	Cycle counter reached zero [Setting condition] When the Cycle Counter value changes from the maximum value (CMAX) to H'0. Reception/transmission of time reference message is successfully completed when CMAX!= 3'b111 and CCR = 0

Bit 3 — Timer Compare Match Flag 2 (TSR3): Indicates that a Compare-Match condition occurred to the Timer Compare Match Register 2 (TCMR2). When the value set in the TCMR2 matches to Cycle Time Register (TCMR2 = CYCTR), this bit is set if TTCR0 bit12 = 1. Please note that this bit is read-only and is cleared when IRR11 (Timer Compare Match Interrupt 2) is cleared.

Bit3: TSR3	Description
0	Timer Compare Match has not occurred to the TCMR2 (Initial value) [Clearing condition] Writing ‘1’ to IRR11 (Timer Compare Match Interrupt 1)
1	Timer Compare Match has occurred to the TCMR2 [Setting condition] TCMR2 matches to Cycle Time (TCMR2 = CYCTR), if TTCR0 bit12 = 1.

Bit 2 — Timer Compare Match Flag 1 (TSR2): Indicates that a Compare-Match condition occurred to the Timer Compare Match Register 1 (TCMR1). When the value set in the TCMR1 matches to Cycle Time Register (TCMR1 = CYCTR), this bit is set if TTCR0 bit11 = 1. Please note that this bit is read-only and is cleared when IRR15 (Timer Compare Match Interrupt 1) is cleared.

Bit2: TSR2	Description
0	Timer Compare Match has not occurred to the TCMR1 (Initial value) [Clearing condition] Writing ‘1’ to IRR15 (Timer Compare Match Interrupt 1)
1	Timer Compare Match has occurred to the TCMR1 [Setting condition] TCMR1 matches to Cycle Time (TCMR1 = CYCTR), if TTCR0 bit11 = 1.

Bit 1 — Timer Compare Match Flag 0 (TSR1): Indicates that a Compare-Match condition occurred to the Compare Match Register 0 (TCMR0). When the value set in the TCMR0 matches to the Timer value (TCMR0 = TCNTR), this bit is set if TTCR0 bit10 = 1. Please note that this bit is read-only and is cleared when IRR14 (Timer Compare Match Interrupt 0) is cleared.

Bit1: TSR1	Description
0	Compare Match has not occurred to the TCMR0 (Initial value) [Clearing condition] Writing ‘1’ to IRR14 (Timer Compare Match Interrupt 0)
1	Compare Match has occurred to the TCMR0 [Setting condition] TCMR0 matches to the Timer value (TCMR0 = TCNTR)

Bit 0 — Timer Overrun/Next_is_Gap Reception/Message Error (TSR0): This flag is assigned to three different functions. It indicates that the Timer has overrun when working in event-trigger mode, time reference message with Next_is_Gap set has been received in time-trigger mode, and error detected on the CAN bus has occurred in test mode, respectively. Test mode has higher priority with respect to the other settings.

Bit0: TSR0	Description
0	Timer (TCNTR) has not overrun in event-trigger mode (Initial value) Time reference message with Next_is_Gap has not been received in time-trigger mode message error has not occurred in test mode. [Clearing condition] Writing ‘1’ to IRR13
1	[Setting condition] Timer (TCNTR) has overrun and changed from H'FFFF to H'0000 in event-trigger mode.time reference message with Next_is_Gap has been received in time-trigger mode message error has occurred in test mode

(5) Cycle Counter Register (CCR)

This register is a 6-bit read/write register. Its purpose is to store the number of the basic cycle for Time -Triggered Transmissions. Its value is updated in different fashions depending if RCAN-TL1 is programmed to work as a potential time master or as a time slave. If RCAN-TL1 is working as (potential) time master, CCR is:

— Incremented by one every time the cycle time (CYCTR) matches to Tx-Trigger Time of Mailbox-30 or
— Overwritten with the value contained in MSG_DATA_0[5:0] of Mailbox 31 when a valid reference message is received.

If RCAN-TL1 is working as a time slave, CCR is only overwritten with the value of MSG_DATA_0[5:0] of Mailbox 31 when a valid reference message is received.

If CMAX = 3'111, CCR is always H'0000.

- CCR (Address = H'08A)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	-	-	-	CCR[5:0]					
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W

Bits 15 to 6: Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bit 5 to 0 — Cycle Counter Register (CCR): Indicates the number of the current Base Cycle of the matrix cycle for Timer Triggered transmission.

(6) Timer Counter Register (TCNTR)

This is a 16-bit read/write register that allows the CPU to monitor and modify the value of the Free Running Timer Counter. When the Timer meets TCMR0 (Timer Compare Match Register 0) + TTCR0 [6] is set to '1', the TCNTR is cleared to H'0000 and starts running again. In Time-Trigger mode, this timer can be used as Local Time and TTCR0[6] has to be cleared to work as a free running timer.

Notes: 1. It is possible to write into this register only when it is enabled by the bit 15 in TTCR0. If TTCR0 bit15 = 0, TCNTR is always H'0000.
2. There could be a delay of a few clock cycles between the enabling of the timer and the moment where TCNTR starts incrementing. This is caused by the internal logic used for the pre-scaler.

- TCNTR (Address = H'08C)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TCNTR[15:0]															
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*

Note: * The register can be written only when enabled in TTCR0[15]. Write operation is not allowed in Time Trigger mode (i.e. CMAX is not 3'b111).

Bit 15 to 0 — Indicate the value of the Free Running Timer.

(7) Cycle Time register (CYCTR)

This register is a 16-bit read-only register. This register shows Cycle Time = Local Time (TCNTR) - Reference_Mark (RFMK). In ET mode this register is the exact copy of TCNTR as RFMK is always fixed to zero.

- CYCTR (Address = H'090)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CYCTR[15:0]															
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

(8) Reference Mark Register (RFMK)

This register is a 16-bit read-only register. The purpose of this register is to capture Local Time (TCNTR) at SOF of the reference message when the message is received or transmitted successfully. In ET mode this register is not used and it is always cleared to zero.

- RFMK (Address = H'094)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RFMK[15:0]															
Initial value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit 15 to 0 — Reference Mark Register (RFMK): Indicates the value of TCNTR at SOF of time reference message.

(9) Timer Compare Match Registers (TCMR0, TCMR1, TCMR2)

These three registers are 16-bit read/write registers and are capable of generating interrupt signals, clearing-setting the Timer value (only supported by TCMR0) or clear the transmission messages in the queue (only supported by TCMR2). TCMR0 is compared with TCNTR, however, TCMR1 and TCMR2 are compared with CYCTR.

The value used for the compare can be configured independently for each register. In order to set flags, TTCR0 bit 12-10 needs to be set.

In Time-Trigger mode, TTCR0 bit6 has to be cleared by software to prevent TCNTR from being cleared.

TMCR0 is for Init_Watch_Trigger, and TCMR2 is for Watch_Trigger.

Interrupt:

The interrupts are flagged by the Bit11, Bit15 and 14 in the IRR accordingly when a Compare Match occurs, and setting these bits can be enabled by Bit12, Bit11, Bit10 in TTCR0. The generation of interrupt signals itself can be prevented by the Bit11, Bit15 and Bit14 in the IMR. When a Compare Match occurs and the IRR11 (or IRR15 or IRR14) is set, the Bit3 or Bit2 or Bit1 in the TSR (Timer Status Register) is also set. Clearing the IRR bit also clears the corresponding bit of TSR.

Timer Clear-Set:

The Timer value can only be cleared when a Compare Match occurs if it is enabled by the Bit6 in the TTCR0. TCMR1 and TCMR2 do not have this function.

Cancellation of the messages in the transmission queue:

The messages in the transmission queue can only be cleared by the TCMR2 through setting TXCR when a Compare Match occurs while RCAN-TL1 is not in the halt status. TCMR1 and TCMR0 do not have this function.

- TCMR0 (Address = H'098)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TCMR0[15:0]															
Initial value:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 15 to 0 — Timer Compare Match Register (TCMR0): Indicates the value of TCNTR when compare match occurs.

- TCMR1 (Address = H'09C)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TCMR1[15:0]															
Initial value:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 15 to 0 — Timer Compare Match Register (TCMR1): Indicates the value of CYCTR when compare match occurs.

- TCMR2 (Address = H'0A0)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TCMR2[15:0]															
Initial value:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 15 to 0 — Timer Compare Match Register (TCMR2): Indicates the value of CYCTR when compare match occurs.

(10) Tx-Trigger Time Selection Register (TTTSEL)

This register is a 16-bit read/write register and specifies the Tx-Trigger Time waiting for compare match with Cycle Time. Only one bit is allowed to be set. Please don't set more bits than one, or clear all bits.

This register may only be modified during configuration mode. The modification algorithm is shown in Figure 14.13.

Please note that this register is only indented for test and diagnosis. When not in test mode, this register must not be written to and the returned value is not guaranteed.

- TTTSEL (Address = H'0A4)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-	TTTSEL[14:8]							-	-	-	-	-	-	-	-
Initial value:	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R	R	R	R	R	R	R

Note: Only one bit is allowed to be set.

Bit 15: Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

Bit 14 to 8 — Specifies the Tx-Trigger Time waiting for compare match with CYCTR The bit 14 to 8 corresponds to Mailbox-30 to 24, respectively.

Bits 7 to 0: Reserved. The written value should always be ‘0’ and the returned value is ‘0’.

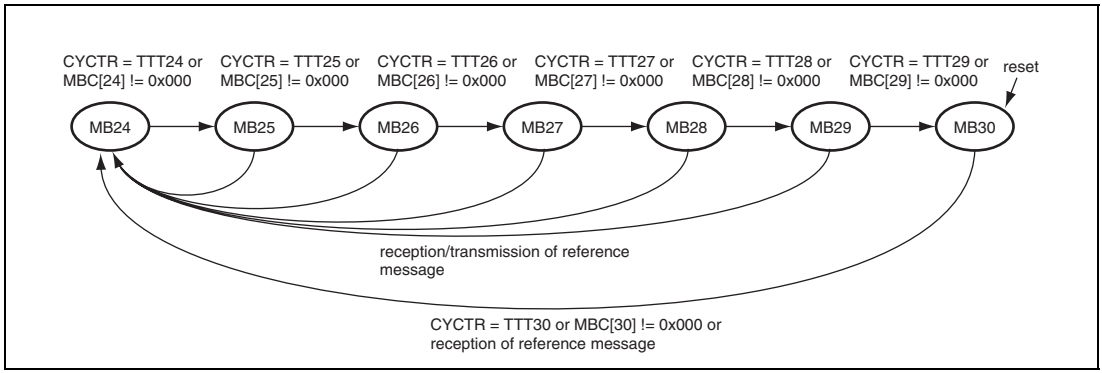


Figure 14.13 TTTSEL modification algorithm

14.4 Application Note

14.4.1 Test Mode Settings

The RCAN-TL1 has various test modes. The register TST[2:0] (MCR[10:8]) is used to select the RCAN-TL1 test mode. The default (initialised) settings allow RCAN-TL1 to operate in Normal mode. The following table is examples for test modes.

Test Mode can be selected only while in configuration mode. The user must then exit the configuration mode (ensuring BCR0/BCR1 is set) in order to run the selected test mode.

Bit10: TST2	Bit9: TST1	Bit8: TST0	Description
0	0	0	Normal Mode (initial value)
0	0	1	Listen-Only Mode (Receive-Only Mode)
0	1	0	Self Test Mode 1 (External)
0	1	1	Self Test Mode 2 (Internal)
1	0	0	Write Error Counter
1	0	1	Error Passive Mode
1	1	0	Setting prohibited
1	1	1	Setting prohibited

Normal Mode: RCAN-TL1 operates in the normal mode.

Listen-Only Mode: ISO-11898 requires this mode for baud rate detection. The Error Counters are cleared and disabled so that the TEC/REC does not increase the values, and the CTxn (n = A, B, C) Output is disabled so that RCAN-TL1 does not generate error frames or acknowledgment bits. IRR13 is set when a message error occurs.

Self Test Mode 1: RCAN-TL1 generates its own Acknowledge bit, and can store its own messages into a reception mailbox (if required). The CRxn/CTxn (n = A, B, C) pins must be connected to the CAN bus.

Self Test Mode 2: RCAN-TL1 generates its own Acknowledge bit, and can store its own messages into a reception mailbox (if required). The CRxn/CTxn (n = A, B, C) pins do not need to be connected to the CAN bus or any external devices, as the internal CTxn (n = A, B, C) is looped back to the internal CRxn (n = A, B, C). CTxn (n = A, B, C) pin outputs only recessive bits and CRxn (n = A, B, C) pin is disabled.

Write Error Counter: TEC/REC can be written in this mode. RCAN-TL1 can be forced to become an Error Passive mode by writing a value greater than 127 into the Error Counters. The value written into TEC is used to write into REC, so only the same value can be set to these registers. Similarly, RCAN-TL1 can be forced to become an Error Warning by writing a value greater than 95 into them.

RCAN-TL1 needs to be in Halt Mode when writing into TEC/REC (MCR1 must be "1" when writing to the Error Counter). Furthermore this test mode needs to be exited prior to leaving Halt mode.

Error Passive Mode: RCAN-TL1 can be forced to enter Error Passive mode.
Note: The REC will not be modified by implementing this Mode. However, once running in Error Passive Mode, the REC will increase normally should errors be received. In this Mode, RCAN-TL1 will enter BusOff if TEC reaches 256 (Dec). However when this mode is used RCAN-TL1 will not be able to become Error Active. Consequently, at the end of the Bus Off recovery sequence, RCAN-TL1 will move to Error Passive and not to Error Active.

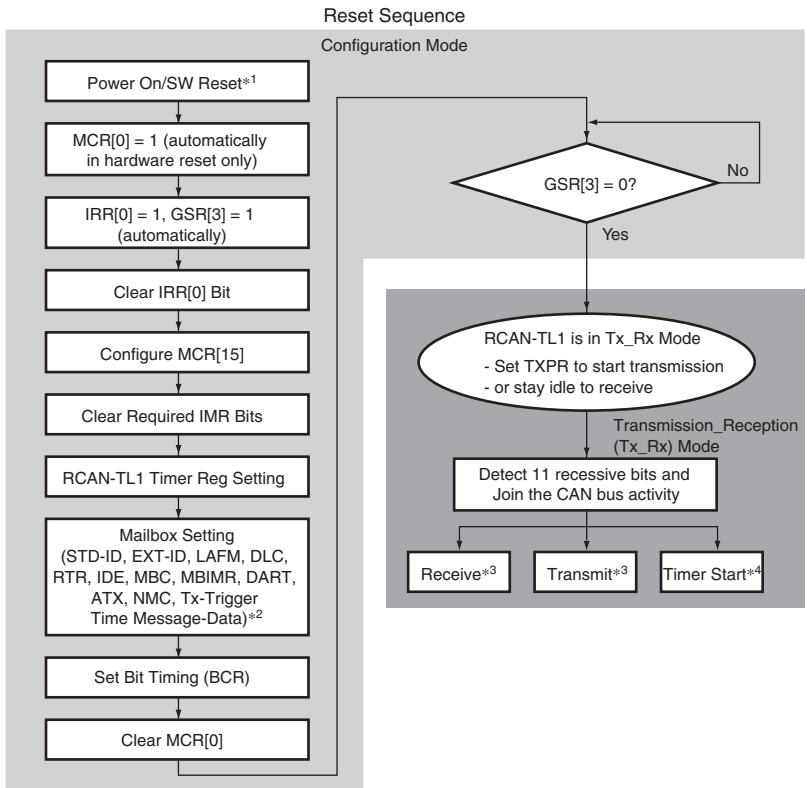
When message error occurs, IRR13 is set in all test modes.

14.4.2 Configuration of RCAN-TL1

RCAN-TL1 is considered in configuration mode or after a H/W (Power On Reset)/S/W (MCR[0]) reset or when in Halt mode. In both conditions RCAN-TL1 cannot join the CAN Bus activity and configuration changes have no impact on the traffic on the CAN Bus.

- After a Reset request

The following sequence must be implemented to configure the RCAN-TL1 after (S/W or H/W) reset. After reset, all the registers are initialised, therefore, RCAN-TL1 needs to be configured before joining the CAN bus activity. Please read the notes carefully.



Notes: 1. SW reset could be performed at any time by setting MCR[0] = 1.
2. Mailboxes are comprised of RAMs, therefore, please initialise all the mailboxes enabled by MBC.
3. If there is no TXPR set, RCAN-TL1 will receive the next incoming message. If there is a TXPR(s) set, RCAN-TL1 will start transmission of the message and will be arbitrated by the CAN bus. If it loses the arbitration, it will become a receiver.
4. Timer can be started at any time after the Timer Control regs and Tx-Trigger Time are set.

Figure 14.14 Reset Sequence

- Halt mode

When RCAN-TL1 is in Halt mode, it cannot take part to the CAN bus activity. Consequently the user can modify all the requested registers without influencing existing traffic on the CAN Bus. It is important for this that the user waits for the RCAN-TL1 to be in halt mode before to modify the requested registers - note that the transition to Halt Mode is not always immediate (transition will occurs when the CAN Bus is idle or in intermission). After RCAN-TL1 transit to Halt Mode, GSR4 is set.

Once the configuration is completed the Halt request needs to be released. RCAN-TL1 will join CAN Bus activity after the detection of 11 recessive bits on the CAN Bus.

- Sleep mode

When RCAN-TL1 is in sleep mode the clock for the main blocks of the IP is stopped in order to reduce power consumption. Only the following user registers are clocked and can be accessed: MCR, GSR, IRR and IMR. Interrupt related to transmission (TXACK and ABACK) and reception (RXPR and RFPR) cannot be cleared when in sleep mode (as TXACK, ABACK, RXPR and RFPR are not accessible) and must to be cleared beforehand.

The following diagram shows the flow to follow to move RCAN-TL1 into sleep mode.

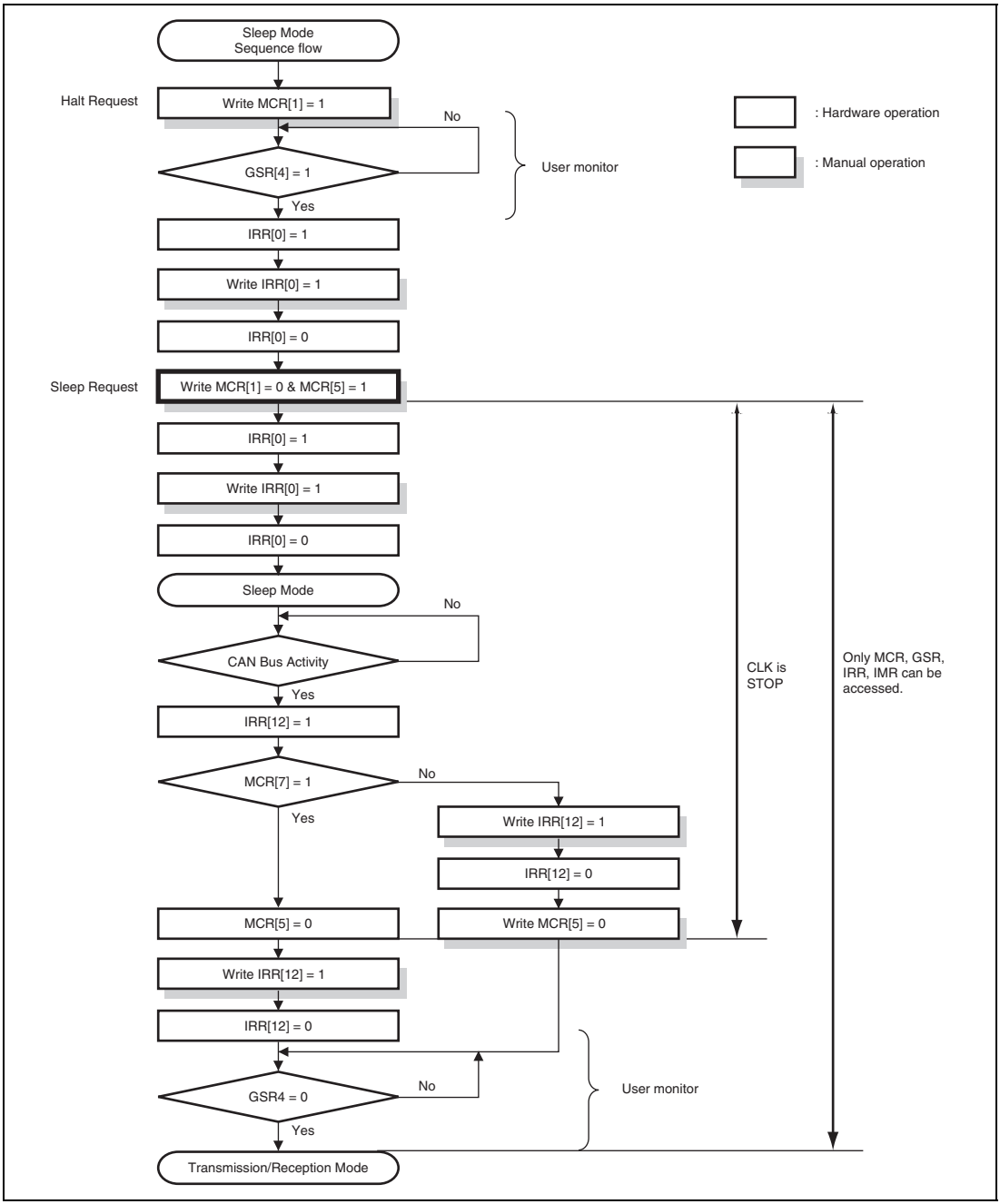


Figure 14.15 shows allowed state transitions.

— Please don't set MCR5 (Sleep Mode) without entering Halt Mode.

— After MCR1 is set, please don't clear it before GSR4 is set and RCAN-TL1 enters Halt Mode.

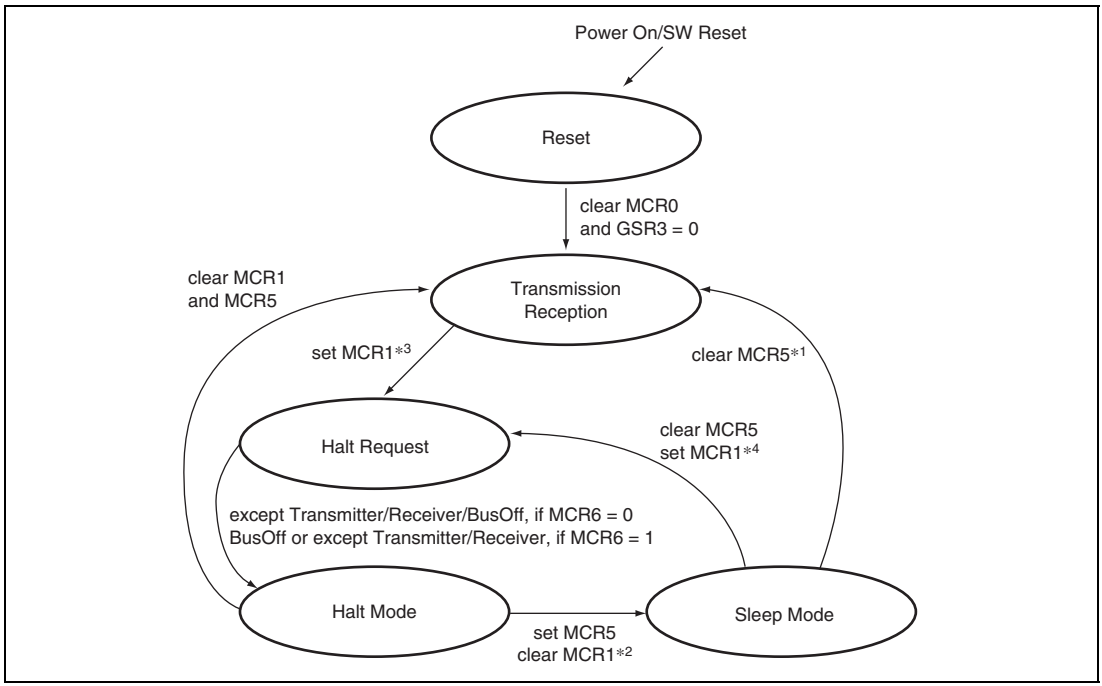


Figure 14.15 Halt Mode/Sleep Mode

Notes: 1. MCR5 can be cleared by automatically by detecting a dominant bit on the CAN Bus if MCR7 is set or by writing '0'.

2. MCR1 is cleared in SW. Clearing MCR1 and setting MCR5 have to be carried out by the same instruction.

3. MCR1 must not be cleared in SW, before GSR4 is set. MCR1 can be set automatically in HW when RCAN-TL1 moves to Bus Off and MCR14 and MCR6 are both set.

4. When MCR5 is cleared and MCR1 is set at the same time, RCAN-TL1 moves to Halt Request. Right after that, it moves to Halt Mode with no reception/transmission.

The following table shows conditions to access registers.

	RCAN-TL1 Registers									
Status Mode	**MCR GSR**	**IRR IMR**	**BCR**	**MBIMR timer TT_register**	**Flag_ register**	**Mailbox (ctrl0, LAFM)**		**Mailbox (data)**	**Mailbox (ctrl1)**		**Mailbox Trigger Time TT control**
Reset	yes	yes	yes	yes	yes	yes		yes	yes		yes
Transmission Reception Halt Request	yes	yes	no*[1]	yes	yes	no*[1]	yes*[2]	yes*[2]	no*[1]	yes*[2]	yes*[2]
Halt	yes	yes	no*[1]	yes	yes	yes		yes	yes		yes
Sleep	yes	yes	no	no	no	no		no	no		no

Notes: 1. No hardware protection.
2. When TXPR is not set.

14.4.3 Message Transmission Sequence

- Message Transmission Request

The following sequence is an example to transmit a CAN frame onto the bus. As described in the previous register section, please note that IRR8 is set when one of the TXACK or ABACK bits is set, meaning one of the Mailboxes has completed its transmission or transmission abortion and is now ready to be updated for the next transmission, whereas, the GSR2 means that there is currently no transmission request made (No TXPR flags set).

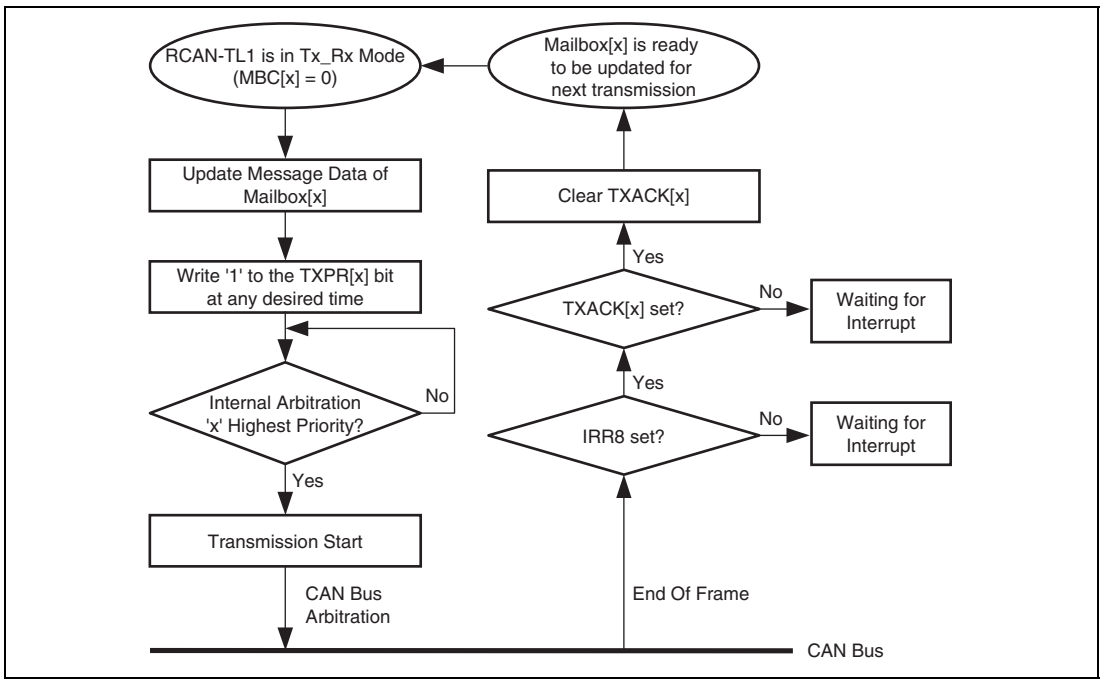


Figure 14.16 Transmission request

- Internal Arbitration for transmission

The following diagram explains how RCAN-TL1 manages to schedule transmission-requested messages in the correct order based on the CAN identifier. ‘Internal arbitration’ picks up the highest priority message amongst transmit-requested messages.

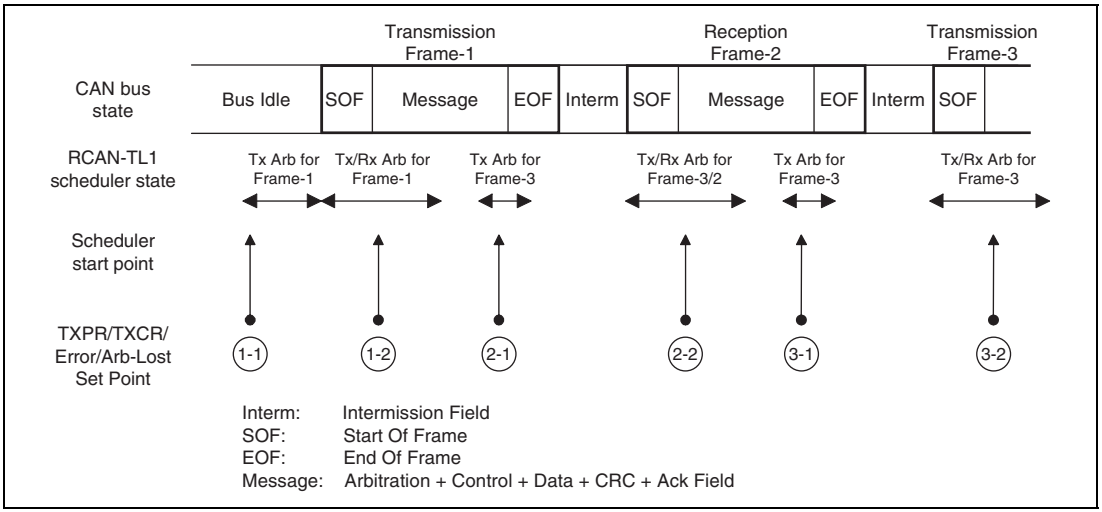


Figure 14.17 Internal Arbitration for transmission

The RCAN-TL1 has two state machines. One is for transmission, and the other is for reception.

1-1: When a TXPR bit(s) is set while the CAN bus is idle, the internal arbitration starts running immediately and the transmission is started.

1-2: Operations for both transmission and reception starts at SOF. Since there is no reception frame, RCAN-TL1 becomes transmitter.

2-1: At crc delimiter, internal arbitration to search next message transmitted starts.

2-2: Operations for both transmission and reception starts at SOF. Because of a reception frame with higher priority, RCAN-TL1 becomes receiver. Therefore, Reception is carried out instead of transmitting Frame-3.

3-1: At crc delimiter, internal arbitration to search next message transmitted starts.

3-2: Operations for both transmission and reception starts at SOF. Since a transmission frame has higher priority than reception one, RCAN-TL1 becomes transmitter.

Internal arbitration for the next transmission is also performed at the beginning of each error delimiter in case of an error is detected on the CAN Bus. It is also performed at the beginning of error delimiters following overload frame.

As the arbitration for transmission is performed at CRC delimiter, in case a remote frame request is received into a Mailbox with ATX = 1 the answer can join the arbitration for transmission only at the following Bus Idle, CRC delimiter or Error Delimiter.

Depending on the status of the CAN bus, following the assertion of the TXCR, the corresponding Message abortion can be handled with a delay of maximum 1 CAN Frame.

(1) Time Triggered Transmission

RCAN-TL1 offers a H/W support to perform communication in Time Trigger mode in line with the emerging ISO-11898-4 Level 1 Specification.

This section reports the basic procedures to use this mode.

- Setting Time Trigger Mode

In order to set up the time trigger mode the following settings need to be used.

— CMAX in CMAX_TEW must be programmed to a value different from 3'b111.
— Bit 15 in TTCR0 has to be set, to start TCNTR.
— Bit 6 in TTCR0 has to be cleared to prevent TCNTR from being cleared after a match.
— DART in Mailboxes used for time-triggered transmission cannot be used, since for Time Triggered Mailboxes, TXPR is not cleared to support periodic transmission.

- Roles of Registers

The user registers of RCAN-TL1 can be used to handle the main functions requested by the TTCAN standard.

TCNTR	Local Time
RFMK	Ref_Mark
CYCTR	Cycle Time = TCNTR - RFMK
RFTROFF	Ref_Trigger_Offset for Mailbox-30
Mailbox-31	Mailbox dedicated to the reception of time reference message
Mailbox-30	Mailbox dedicated to the transmission of time reference message when working as a potential time master
Mailbox-29 to 24	Mailboxes supporting time-triggered transmission
Mailbox-23 to 16	Mailboxes supporting reception without timestamp (may also be implemented as Mailboxes supporting Event Triggered transmission)
Mailbox-15 to 0	Mailboxes supporting reception with timestamp timestamp (may also be implemented as Mailboxes supporting Event Triggered transmission)
Tx-Trigger Time	Time_Mark to specify when a message should be transmitted

CMAX	Specifies the maximum number of basic cycles when working as potential time master
TEW	Specify the width of Tx_Enable
TCMR0	Init_Watch_Trigger (compare match with Local Time)
TCMR1	Compare match with Cycle Time to monitor users-specified events
TCMR2	Watch_Trigger (compare match with Cycle Time). This can be programmed to abort all pending transmissions
TTW	Specifies the attribute of a time window used for transmission
TTTSEL	Specifies the next Mailbox waiting for transmission

- Time Master/Time Slave

RCAN-TL1 can be programmed to work as a potential time master of the network or as a time slave. The following table shows the settings and the operation automatically performed by RCAN-TL1 in each mode.

mode	requested setting	function
Time Slave	TXPR[30] = 0 & MBC[30]!= 3'b000 & CMAX!= 3'b111 & MBC[31] = 3'b011	TCNTR is sampled at each SOF detected on the CAN Bus and stored into an internal register. When a valid Time Reference Message is received into Mailbox-31 the value of TCNTR (stored at the SOF) is copied into Ref_Mark. CCR embedded in the received Reference Message is copied to CCR. If Next_is_Gap = 1, IRR13 is set.
(Potential) Time Master	TXPR[30] = 1 & MBC[30] = 3'b000 & DLC[30] > 0 & CMAX!= 3'b111 & MBC[31] = 3'b011	Two cases are covered: (1) When a valid Time Reference message is received into Mailbox-31 the value of TCNTR stored into an internal register at the SOF is copied into Ref_Mark. CCR embedded in the received Reference Message is copied to CCR. If Next_is_Gap = 1, IRR13 is set. (2) When a Time Reference message is transmitted from Mailbox-30 the value of TCNTR stored into an internal register at the SOF is copied into Ref_Mark. CCR is incremented when TTT of Mailbox-30 matches with CYCTR . CCR is embedded into the first data byte of the time reference message { Data0[7:6], CCR[5:0] } .

- Setting Tx-Trigger Time

The Tx-Trigger Time(TTT) must be set in ascending order shown below, and the difference between them has to satisfy the following expressions. TEW in the following expressions is the register value.

TTT (Mailbox-24) < TTT (Mailbox-25) < TTT (Mailbox-26) < TTT (Mailbox-27) < TTT (Mailbox-28) < TTT (Mailbox-29) < TTT (Mailbox-30)

and

TTT (Mailbox-i) – TTT (Mailbox- i-1) > TEW + the maximum frame length + 9

TTT (Mailbox-24) to TTT (Mailbox-29) correspond to Time_Marks, and TTT (Mailbox-30) corresponds to Time_Ref showing the length of a basic cycle, respectively when working as potential time master.

The above limitation is not applied to mailboxes which are not set as time-triggered transmission.

Important: Because of limitation on setting Tx-Trigger Time, only one Mailbox can be assigned to one time window.

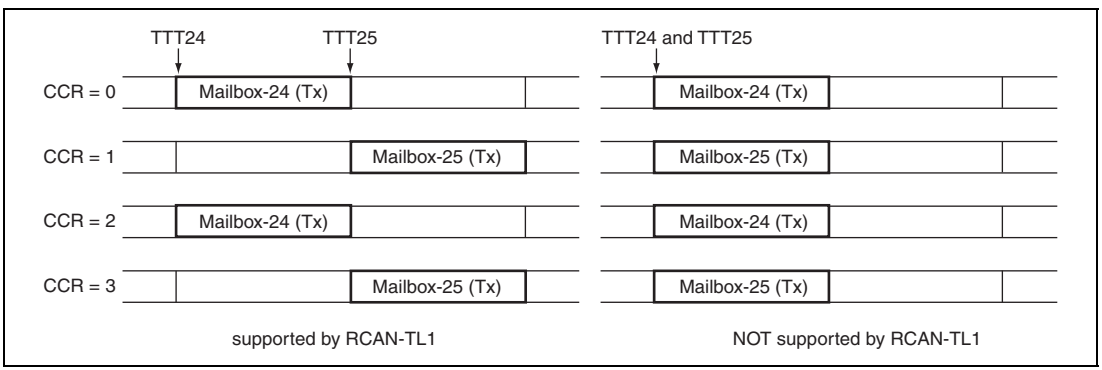


Figure 14.18 Limitation on Tx-Trigger Time

The value of TCMR2 as Watch_Trigger has to be larger than TTT(Mailbox-30), which shows the length of a basic cycle.

Figure 14.19 and Figure 14.20 show examples of configurations for (Potential) Time Master and Time Slave. “L” in diagrams shows the length in time of the time reference messages.

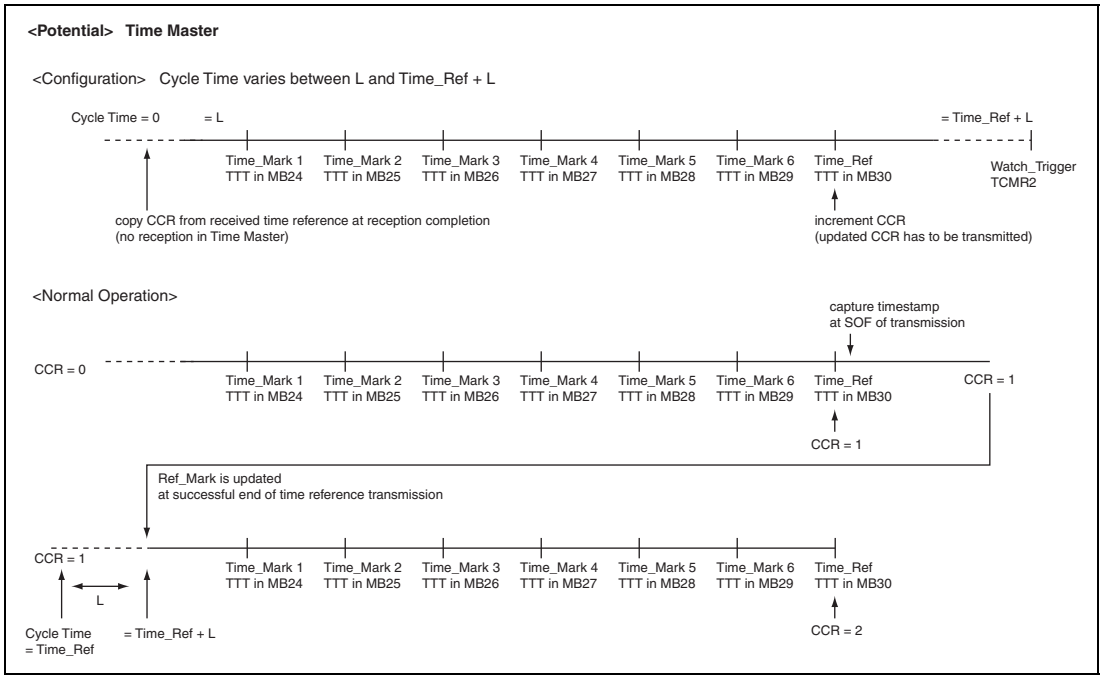


Figure 14.19 (Potential) Time Master

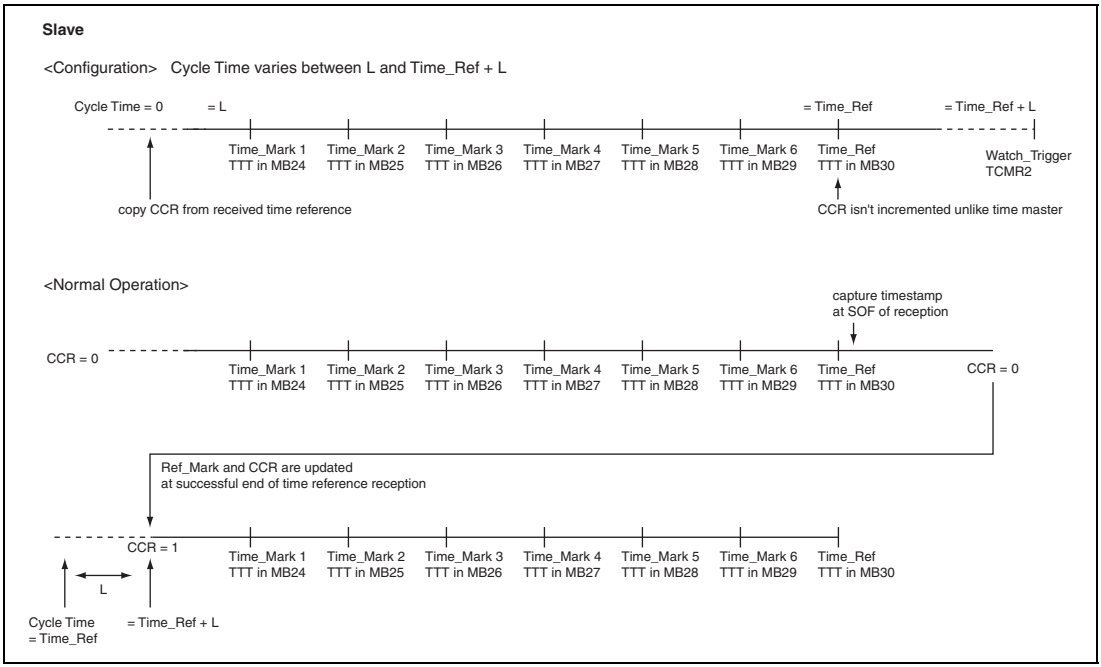


Figure 14.20 Time Slave

- Function to be implemented by software

Some of the TTCAN functions need to be implemented in software. The main details are reported hereafter. Please refer to ISO-11898-4 for more details.

— Change from Init_Watch_Trigger to Watch_Trigger

RCAN-TL1 offers the two registers TCMR0 and TCMR2 as H/W support for Init_Watch_Trigger and Watch_Trigger respectively. The SW is requested to enable TCMR0 and disable TCMR2 up to the first reference message is detected on the CAN Bus and then disable TCMR0 and enable TCMR2.- Schedule Synchronization state machine.

Only reception of Next_is_Gap interrupt is supported. The application needs to take care of stopping all transmission at the end of the current basic cycle by setting the related TXCR flags.Master-Slave Mode control.

Only automatic cycle time synchronization and CCR increment is supported.

— Message status count

Software has to count scheduling errors for periodic messages in exclusive windows.

- Message Transmission Request for Time Triggered communication

When the Time Triggered mode is used communications must fulfils the ISO11898-4 requirements.

The following procedure should be used.

— Send RCAN-TL1 to reset or halt mode
— Set TCMR0 to the Init_Watch_Trigger (0xFFFF)
— Enable TCMR0 compare match setting bit 10 of TTCR0
— Set TCMR2 to the specified Watch_Trigger value
— Keep TCMR2 compare match disabled by keeping cleared the bit 12 of TTCR0
— Set CMAX to the requested value (different from 111 bin)
— Set TEW to the requested value
— Configure the necessary Mailboxes for Time Trigger transmission and reception
— Set LAFM for the 3 LSBs of Mailbox 31
— Configure MCR, BCR1 and BCR0 to the requested values
— If working as a potential time master:
  • Set RFTROFF to the requested Init_Ref_Offset value
  • Set TXPR for Mailbox 30
  • Write H'4000 into TTTSEL
— Enable the TCNTR timer through the bit 15 of TTCR0
— Move to Transmission_Reception mode
— Wait for the reception or transmission of a valid reference message or for TCMR0 match
— If the local time reaches the value of TCMR0 the Init_Watch_Trigger is reached and the application needs to set TXCR for Mailbox 30 and start again
— If the reference message is transmitted (TXACK[30] is set) set RFTROFF to zero
— If a valid reference message is received (RXPR[31] is set) then:
  • If 3 LSBs of ID of Mailbox 31 have high priority than the 3 LSBs of Mailbox 30 (if working as potential time master) keep RFTROFF to Init_Ref_Offset
  • If 3 LSBs of ID of Mailbox 31 have lower priority than the 3 LSBs of Mailbox 30 (if working as potential time master) decrement by 1 the value in RFTROFF
— Disable TCMR0 compare match by clearing bit 10 of TTCR0
— Enable TCMR2 compare match by setting bit 12 of TTCR0
— Only after two reference messages have been detected on the CAN Bus (transmitted or received) can the application set TXPR for the other Time Triggered Mailboxes.

If, at any time, a reference message cannot be detected on the CAN Bus, and the cycle time CYCTR reaches TCMR2, RCAN-TL1 automatically aborts all pending transmissions (including the Reference Message).

The following is the sequence to request further transmission in Time Triggered mode.

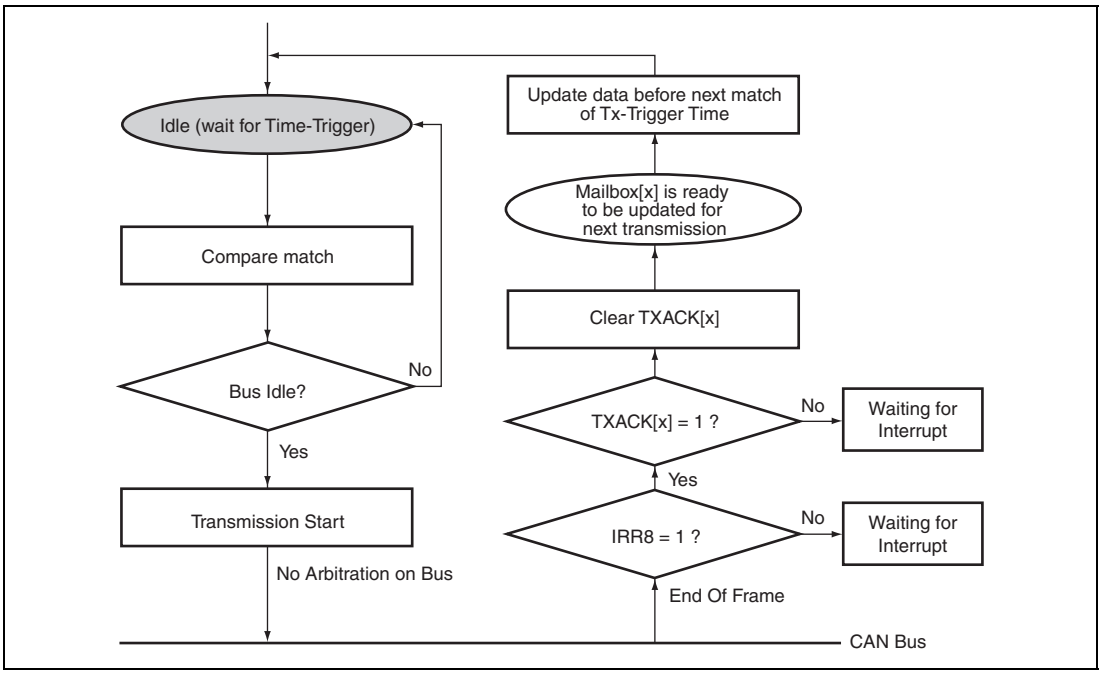


Figure 14.21 Message transmission request

S/W has to ensure that a message is updated before a Tx trigger for transmission occurs.

When the CYCTR reaches to TTT (Tx-Trigger Time) of a Mailbox and CCR matches with the programmed cycle for transmission, RCAN-TL1 immediately transfers the message into the Tx buffer. At this point, RCAN-TL1 will attempt a transmission within the specified Time Enable Window. If RCAN-TL1 misses this time slot, it will suspend the transmission request up to the next Tx Trigger, keeping the corresponding TXPR bit set to '1' if the transmission is periodic (Mailbox-24 to 30). There are three factors that may cause RCAN-TL1 to miss the time slot –

1. The CAN bus currently used
2. An error on the CAN bus during the time triggered message transmission
3. Arbitration loss during the time triggered message transmission

In case of Merged Arbitrating Window the slot for transmission goes from the Tx_Trig of the Mailbox opening the Window (TTW = 10 bin) to the end to the TEW of the Mailbox closing the Window (TTW = 11 bin).The TXPR can be modified at any time. RCAN-TL1 ensures the transmission of Time Triggered messages is always scheduled correctly. However, in order to guarantee the correct schedule, there are some important rules that are :

— TTT (Tx Trigger Time) can be modified during configuration mode.
— TTT cannot be set outside the range of Time_Ref, which specifies the length of basic cycle. This could cause a scheduling problem.
— TXPR is not automatically cleared for periodic transmission. If a periodic transmission needs to be cancelled, the corresponding TXCR bit needs to be set by the application.

- Example of Time Triggered System

The following diagram shows a simple example of how time trigger system works using RCAN-TL1 in time slave mode.

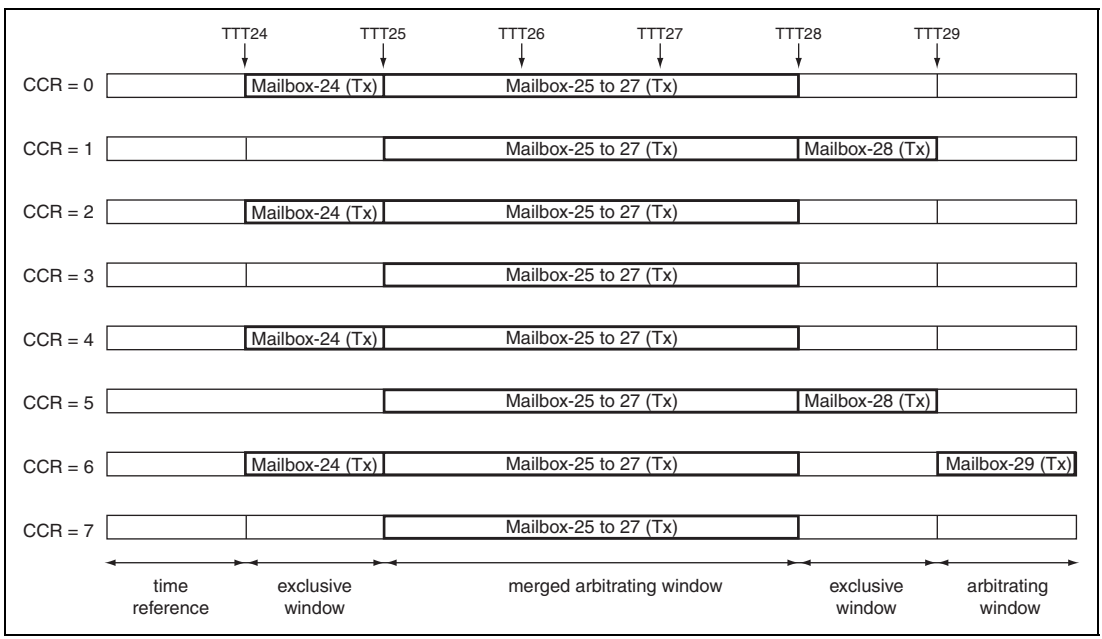


Figure 14.22 Example of Time trigger system as Time Slave

The following settings were used in the above example:

	rep_factor (register)	Offset	TTW[1:0]	MBC[2:0]
Mailbox-24	3'b001	6'b000000	2'b00	3'b000
Mailbox-25	3'b000	6'b000000	2'b10	3'b000
Mailbox-26	3'b000	6'b000000	2'b10	3'b000
Mailbox-27	3'b000	6'b000000	2'b11	3'b000
Mailbox-28	3'b010	6'b000001	2'b00	3'b000
Mailbox-29	3'b011	6'b000110	2'b01	3'b000
Mailbox-30	—	—	—	3'b111
Mailbox-31	—	—	—	3'b011

CMAX = 3'b011, TXPR[30] = 0

During merged arbitrating window, request by time-triggered transmission is served in the way of FCFS (First Come First Served). For example, if Mailbox-25 cannot be transmitted between Tx-Trigger Time 25 (TTT25) and TTT26, Mailbox-25 has higher priority than Mailbox-26 between TTT26 and 28.

MBC needs to be set into 3'b111, in order to disable time-triggered transmission. If RCAN-TL1 is Time Master, MBC[30] has to be 3'b000 and time reference window is automatically recognized as arbitrating window.

- Timer Operation

Figure 14.23 shows the timing diagram of the timer. By setting Tx-Trigger Time = n, time trigger transmission starts between CYCTR = n + 2 and
CYCTR = n + 3.

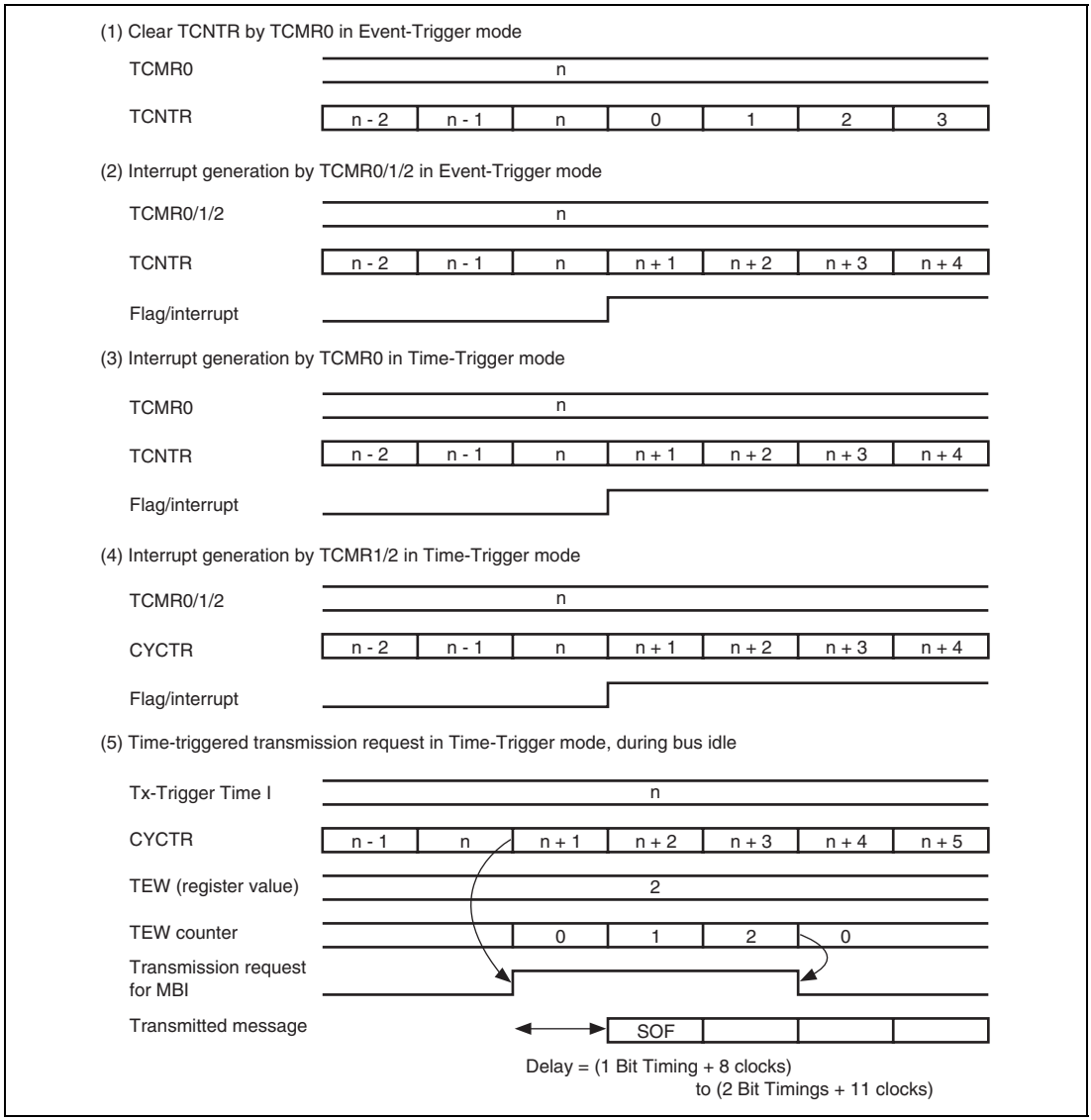


Figure 14.23 Timing Diagram of Timer

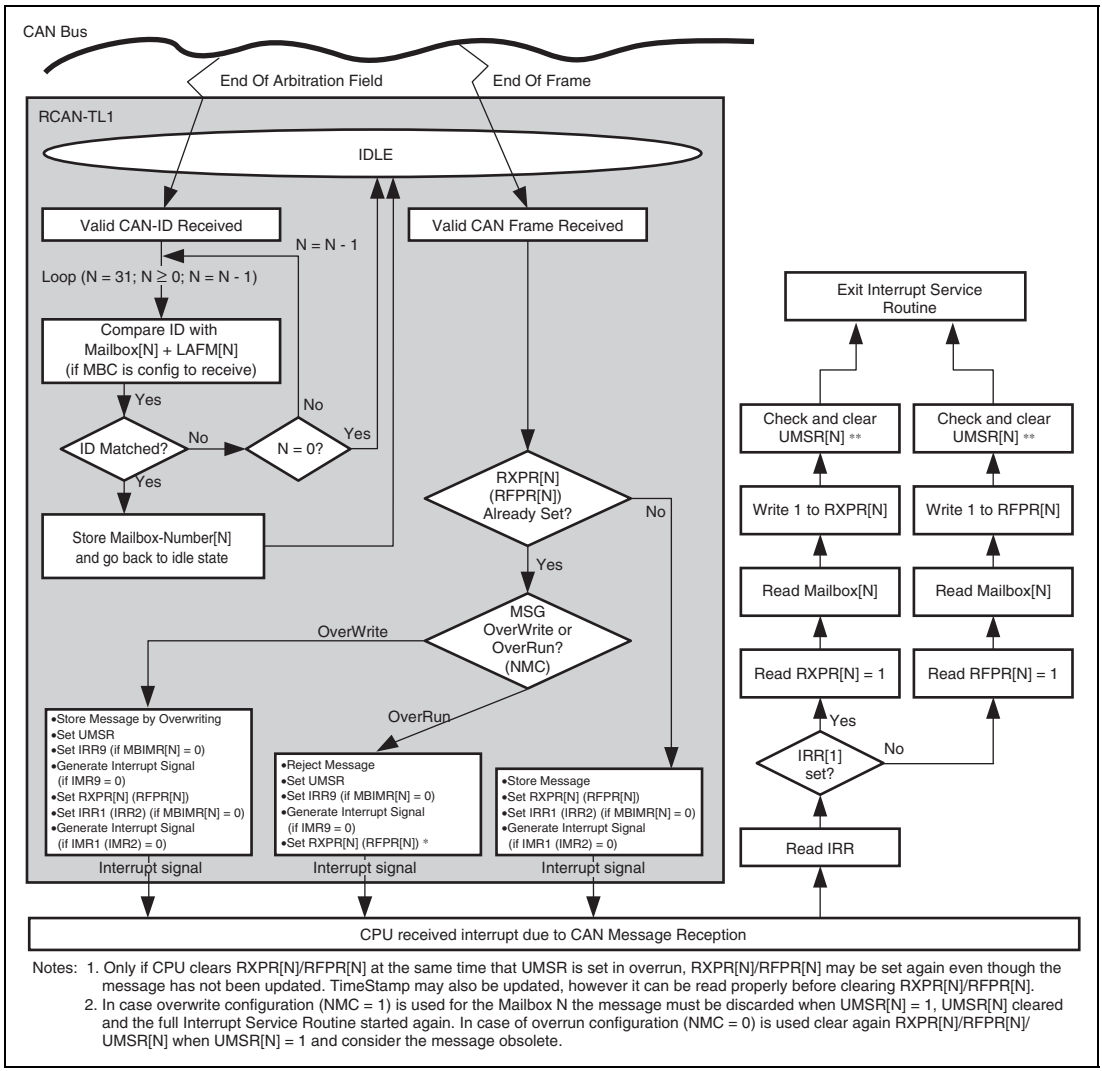
During merged arbitrating window, event-trigger transmission is served after completion of time-triggered transmission. For example, If transmission of Mailbox-25 is completed and CYCTR doesn't reach TTT26, event-trigger transmission starts based on message transmission priority specified by MCR2. TXPR of time-triggered transmission is not cleared after transmission completion, however, that of event-triggered transmission is cleared.

Note: that in the case that the TXPR is not set for the Mailbox which is assigned to close the Merged Arbitrating Window (MAW), then the MAW will still be closed (at the end of the TEW following the TTT of the assigned Mailbox.

Please refer to Table, Roles of Mailboxes, in section 14.3.2, Mailbox Structure.

14.4.4 Message Receive Sequence

The diagram below shows the message receive sequence.



Notes: 1. Only if CPU clears RXPR[N]/RFPR[N] at the same time that UMSR is set in overrun, RXPR[N]/RFPR[N] may be set again even though the message has not been updated. TimeStamp may also be updated, however it can be read properly before clearing RXPR[N]/RFPR[N].
2. In case overwrite configuration (NMC = 1) is used for the Mailbox N the message must be discarded when UMSR[N] = 1, UMSR[N] cleared and the full Interrupt Service Routine started again. In case of overrun configuration (NMC = 0) is used clear again RXPR[N]/RFPR[N]/ UMSR[N] when UMSR[N] = 1 and consider the message obsolete.

Figure 14.24 Message receive sequence

When RCAN-TL1 recognises the end of the Arbitration field while receiving a message, it starts comparing the received identifier to the identifiers set in the Mailboxes, starting from Mailbox-31 down to Mailbox-0. It first checks the MBC if it is configured as a receive box, and reads LAFM, and reads the CAN-ID of Mailbox-31 (if configured as receive) to finally compare them to the received ID. If it does not match, the same check takes place at Mailbox-30 (if configured as receive). Once RCAN-TL1 finds a matching identifier, it stores the number of Mailbox-[N] into an internal buffer, stops the search, and goes back to idle state, waiting for the EndOfFrame (EOF) to come. When the 6th bit of EOF is notified by the CAN Interface logic, the received message is written or abandoned, depending on the NMC bit. No modification of configuration during communication is allowed. Entering Halt Mode is one of ways to modify configuration. If it is written into the corresponding Mailbox, including the CAN-ID, i.e., there is a possibility that the CAN-ID is overwritten by a different CAN-ID of the received message due to the LAFM used. This also implies that, if the identifier of a received message matches to ID + LAFM of 2 or more Mailboxes, the higher numbered Mailbox will always store the relevant messages and the lower numbered Mailbox will never receive messages. Therefore, the settings of the identifiers and LAFMs need to be carefully selected.

With regards to the reception of data and remote frames described in the above flow diagram the clearing of the UMSR flag after the reading of IRR is to detect situations where a message is overwritten by a new incoming message stored in the same mailbox (if its NMC = 1) while the interrupt service routine is running. If during the final check of UMSR a overwrite condition is detected the message needs to be discarded and read again.

In case UMSR is set and the Mailbox is configured for overrun (NMC = 0) the message is still valid, however it is obsolete as it is not reflecting the latest message monitored on the CAN Bus.

Please access the full Mailbox content before clearing the related RXPR/RFPR flag.

Please note that in the case a received remote frame is overwritten by a data frame, both the remote frame receive interrupt (IRR2) and data frame received interrupt (IRR1) and also the Receive Flags (RXPR and RFPR) are set. In an analogous way, the overwriting of a data frame by a remote frame, leads to setting both IRR2 and IRR1.

When a message is received and stored into a Mailbox all the fields of the data not received are stored as zero. The same applies when a standard frame is received. The extended identifier part (EXTID[17:0]) is written as zero.

14.4.5 Reconfiguration of Mailbox

When re-configuration of Mailboxes is required, the following procedures should be taken.

- Change configuration of transmit box

  Two cases are possible.

  — Change of ID, RTR, IDE, LAFM, Data, DLC, NMC, ATX, DART

  This change is possible only when MBC = 3'b000. Confirm that the corresponding TXPR is not set. The configuration (except MBC bit) can be changed at any time.

  — Change from transmit to receive configuration (MBC)

  Confirm that the corresponding TXPR is not set. The configuration can be changed only in Halt or reset state. Please note that it might take longer for RCAN-TL1 to transit to halt state if it is receiving or transmitting a message (as the transition to the halt state is delayed until the end of the reception/transmission), and also RCAN-TL1 will not be able to receive/transmit messages during the Halt state.

  In case RCAN-TL1 is in the Bus Off state the transition to halt state depends on the configuration of the bit 6 of MCR and also bit and 14 of MCR.

- Change configuration (ID, RTR, IDE, LAFM, Data, DLC, NMC, ATX, DART, MBC) of receiver box or Change receiver box to transmitter box

  The configuration can be changed only in Halt Mode.

  RCAN-TL1 will not lose a message if the message is currently on the CAN bus and RCAN-TL1 is a receiver. RCAN-TL1 will be moving into Halt Mode after completing the current reception. Please note that it might take longer if RCAN-TL1 is receiving or transmitting a message (as the transition to the halt state is delayed until the end of the reception/transmission), and also RCAN-TL1 will not be able to receive/transmit messages during the Halt Mode.

  In case RCAN-TL1 is in the Bus Off state the transition to halt mode depends on the configuration of the bit 6 and 14 of MCR.

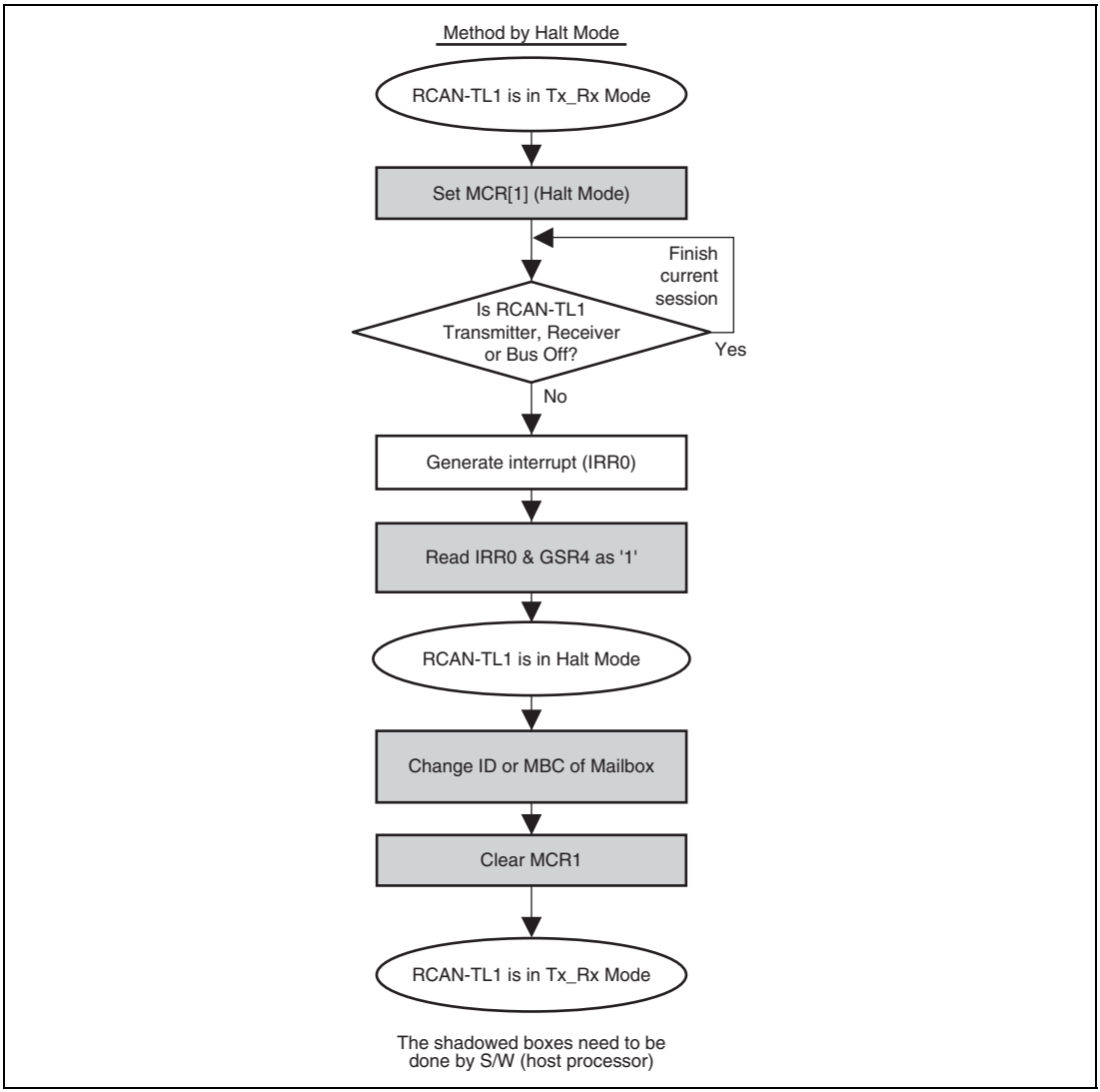


Figure 14.25 Change ID of receive box or Change receive box to transmit box

14.5 Interrupt Sources

Table 14.2 lists the RCAN-TL1 interrupt sources. These sources can be masked. Masking is implemented using the mailbox interrupt mask registers (MBIMR) and interrupt mask register (IMR). For details on the interrupt vector of each interrupt source, see section 5, Interrupt Controller.

Table 14.2 RCAN-TL1 Interrupt Sources

Interrupt	Description	Interrupt Flag	DMAC Activation
ERSn*[1]	Error Passive Mode (TEC ≥ 128 or REC ≥ 128)	IRR5	Not possible
	Bus Off (TEC ≥ 256)/Bus Off recovery	IRR6	
	Error warning (TEC ≥ 96)	IRR3	
	Error warning (REC ≥ 96)	IRR4	
OVRn*[1]	Reset/halt/CAN sleep transition	IRR0	
	Overload frame transmission	IRR7	
	Unread message overwrite (overrun)	IRR9	
	Start of new system matrix	IRR10	
	TCMR2 compare match	IRR11	
	Bus activity while in sleep mode	IRR12	
	Timer overrun/Next_is_Gap reception/message error	IRR13	
	TCMR0 compare match	IRR14	
	TCMR1 compare match	IRR15	
RMn0*[1]*[2], RMn1*[1]*[2]	Data frame reception	IRR1*[3]	Possible*[4]
	Remote frame reception	IRR2*[3]	
SLEn*[1]	Message transmission/transmission disabled (slot empty)	IRR8	Not possible

Notes: 1. n = A, B, C
2. RM0 is an interrupt generated by the remote request pending flag for mailbox 0 (RFPR0[0]) or the data frame receive flag for mailbox 0 (RXPR0[0]). RM1 is an interrupt generated by the remote request pending flag for mailbox n (RFPR0[n]) or the data frame receive flag for mailbox n (RXPR0[n]) (n = 1 to 31).
3. IRR1 is a data frame received interrupt flag for mailboxes 0 to 31, and IRR2 is a remote frame request interrupt flag for mailboxes 0 to 31.
4. The DMAC is activated only by an RMn0 interrupt.

14.6 DMAC Interface

The DMAC can be activated by the reception of a message in RCAN-TL1 mailbox 0. When DMAC transfer ends after DMAC activation has been set, flags of RXPR0 and RFPR0 are cleared automatically. An interrupt request due to a receive interrupt from the RCAN-TL1 cannot be sent to the CPU in this case. Figure 14.26 shows a DMAC transfer flowchart.

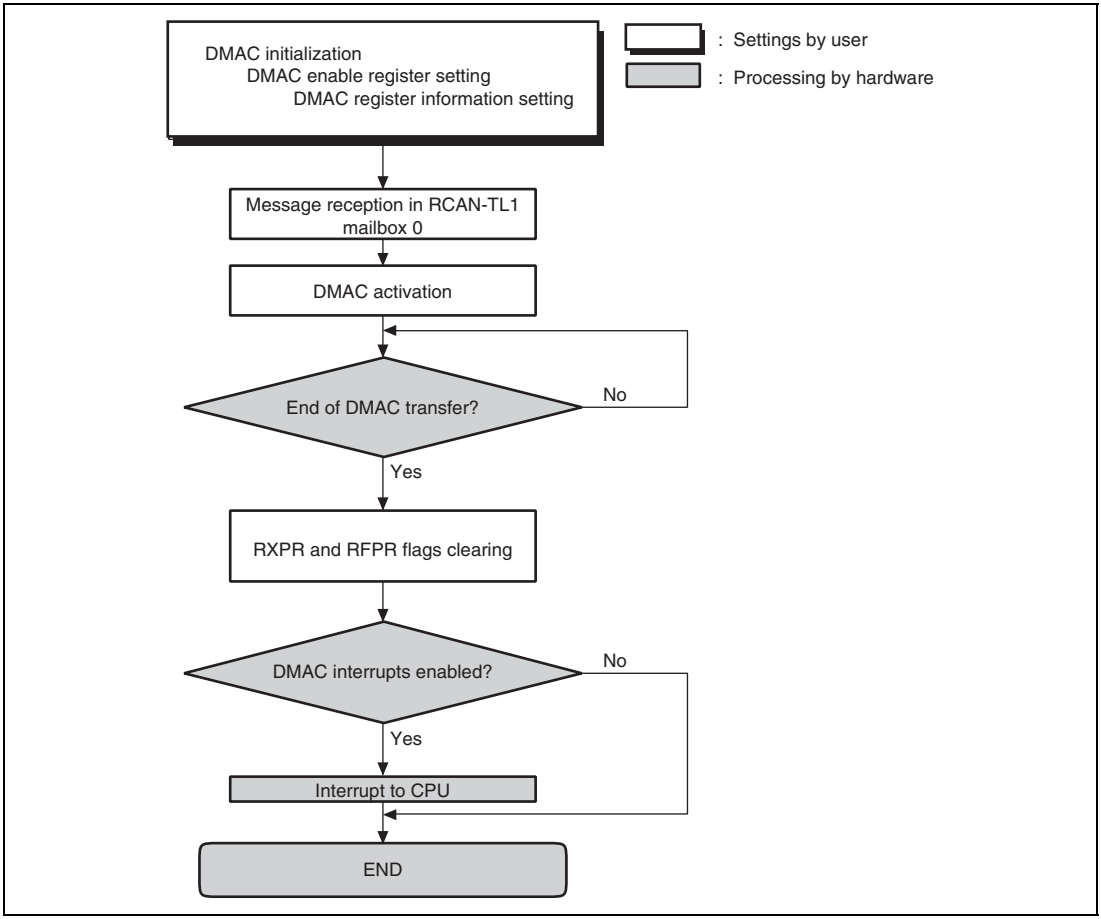


Figure 14.26 DMAC Transfer Flowchart

14.7 PORT Interface

The RCAN monitor register (RCANMON) controls RCAN-TL1 pins.

14.7.1 RCAN Monitor Register (RCANMON)

RCANMON controls the transmit stop condition of transmit pins, enables/disables RCAN-TL1 transmit/receive pins, and monitors the status of RCAN-TL1 pins. This register is not affected by the module stop or CAN sleep mode, nor initialized by the software reset (MCR0).

Bit	7	6	5	4	3	2	1	0
Bit Name	—	CTxSTP	RCANE	—	—	—	CTxD	CRxD
Initial Value	0	0	0	0	0	0	Undefined	Undefined
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R

Bit	Bit Name	Initial Value	R/W	Description
7	—	0	R/W	Reserved This bit is always read as 0. The write value should always be 0.
6	CTxSTP	0	R/W	RCAN Transmit Stop When this bit is set to 1, CTx pins are set to 1 regardless of the RCAN-TL1 transmit data.
5	RCANE	0	R/W	RCAN-TL1 Transmit/Receive Pin Enable When this bit is set to 1, CTx and CRx pins of RCAN-TL1 are enabled.
4 to 2	—	All 0	R/W	Reserved These bits are always read as 0. The write value should always be 0.
1	CTxD	Undefined	R	RCAN Transmit Data Monitor The condition of CTx pins is acquired when this bit is read. Writing to this bit is invalid.
0	CRxD	Undefined	R	RCAN Receive Data Monitor The condition of CRx pins is acquired when this bit is read. Writing to this bit is invalid.

The overview of RCANMON bits in the PORT interface is shown in figure 14.27.

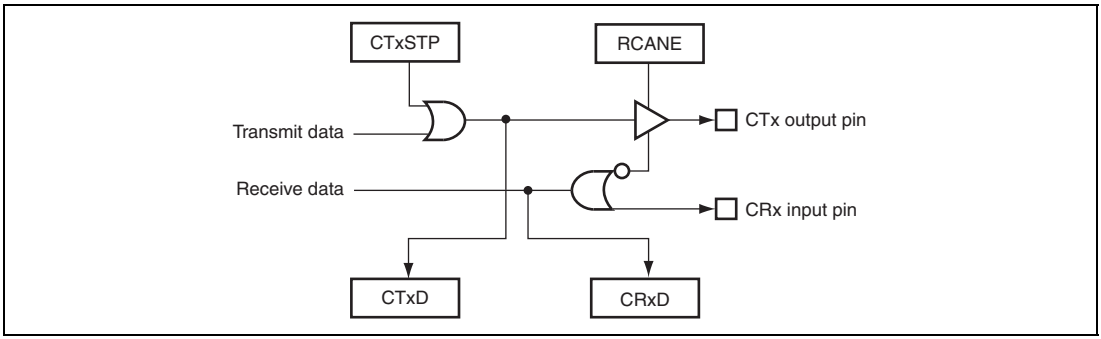


Figure 14.27 Overview of PORT Interface

14.8 CAN Bus Interface

A bus transceiver IC is necessary to connect this LSI to a CAN bus. A Renesas HA13721 transceiver IC and its compatible products are recommended. Figure 14.28 shows a sample connection diagram.

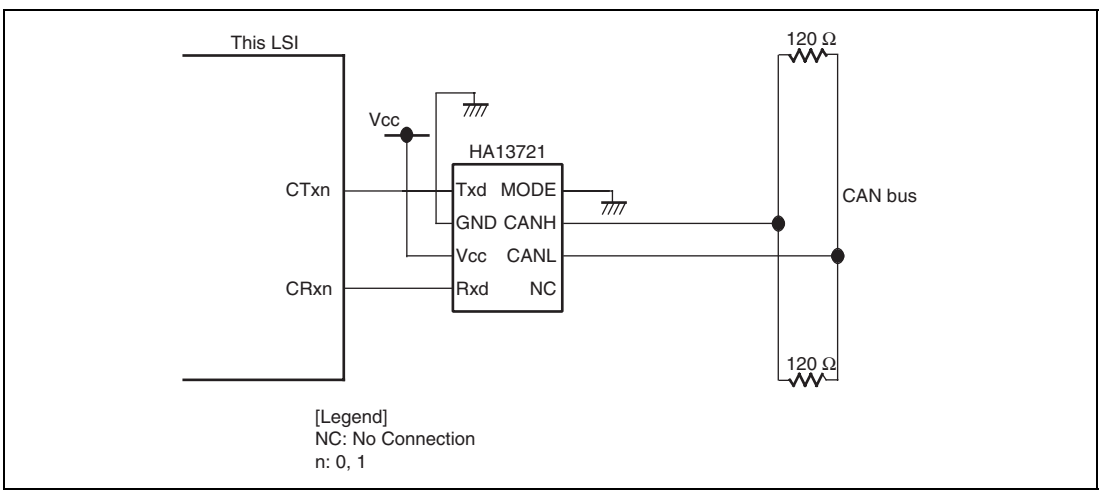


Figure 14.28 High-Speed CAN Interface Using HA13721

14.9 Setting Ports for RCAN-TL1

The ports for the RCAN-TL1 must be specified before or during the configuration mode. For details on the settings of ports, see section 9.3.1, Port Function Control Register 5 (PFCR5). Two methods are available using two channels of the RCAN-TL1 in this LSI.

- Using RCAN-TL1 as a 2-channel module
  Each of RCAN0 and RCAN1 has 32 Mailboxes.
- Using RCAN-TL1 as a 1-channel module
  RCAN0 and RCAN1 as a single channel has 64 Mailboxes.

When 64 Mailboxes are used, caution is required. See section 14.10.1, Notes on Port Setting for Two Channels Used as Single Channel of 64 Mailboxes.

Figures 14.29 and 14.30 show connection examples for individual port settings.

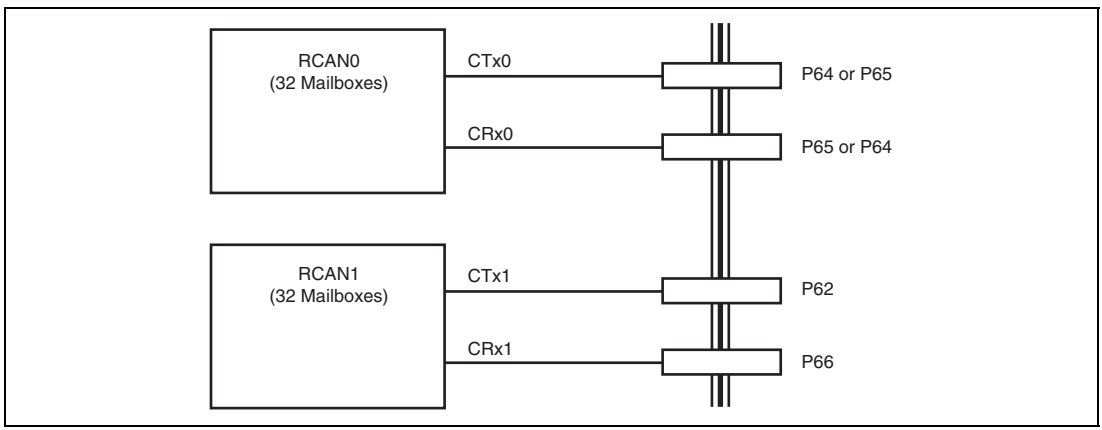


Figure 14.29 Connection Example when Using RCAN-TL1 as 2-Channel Module (32 Mailboxes × 2 Channels)

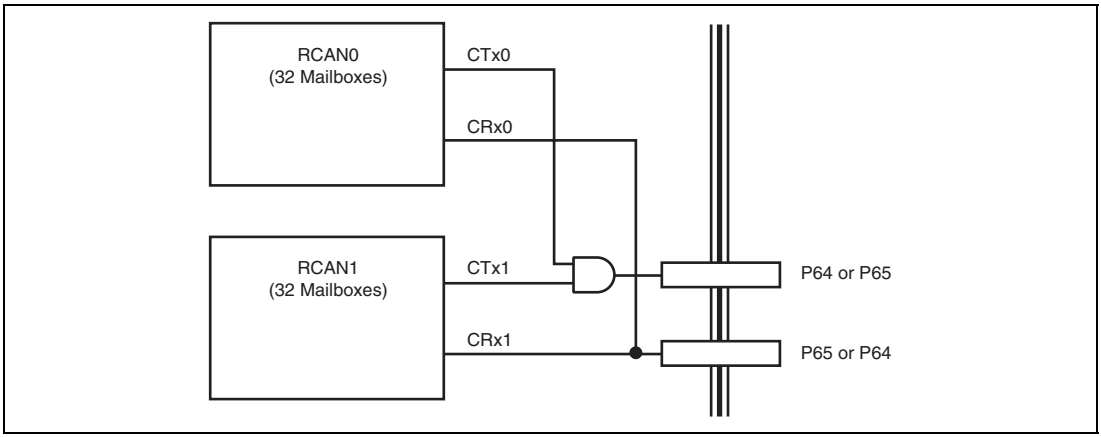


Figure 14.30 Connection Example when Using RCAN-TL1 as 1-Channel Module (64 Mailboxes × 1 Channel)

14.10 Usage Notes

14.10.1 Notes on Port Setting for Two Channels Used as Single Channel of 64 Mailboxes

The RCAN-TL1 in this LSI has two channels. When two channels are used as a single channel that has 64 Mailboxes, keep the following in mind.

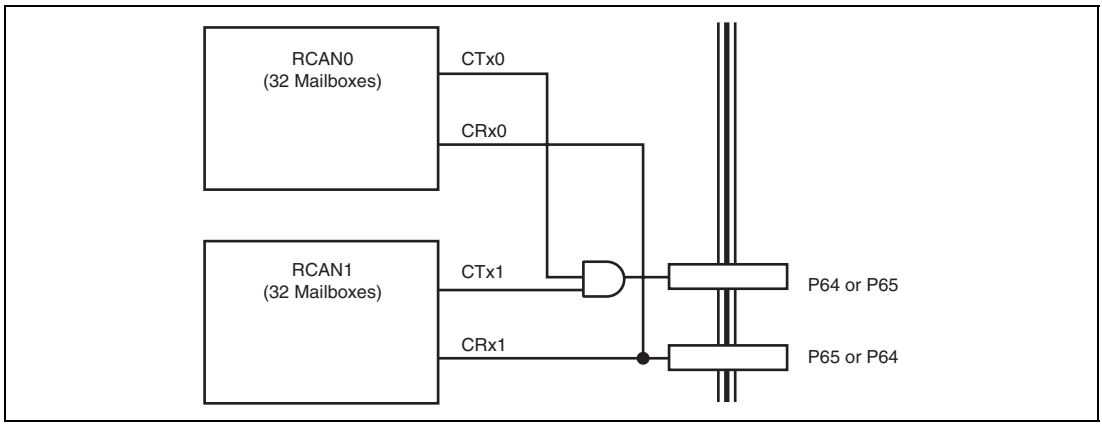


Figure 14.31 Connection Example when Using RCAN-TL1 as 1-Channel Module (64 Mailboxes × 1 Channel)

1. No ACK error is detected even when any other nodes are not connected to the CAN bus. This occurs when RCAN1 transmits an ACK in the ACK field in response to a message RCAN0 has transmitted.

   RCAN1 receives a message which channel RCAN0 has transmitted on the CAN bus and then transmits an ACK in the ACK field. After that, RCAN0 receives the ACK.

   To avoid this, make RCAN1 which is not currently used for transmission the listen-only mode (TST[2:0] = B'001) or the reset state (MCR0 = 1). With this setting, only a channel which transmits a message transmits an ACK.
2. Internal arbitration for RCAN0 and RCAN1 is independently controlled to determine the order of transmission.

   Although the internal arbitration is performed on 31 Mailboxes at a time, it is not performed on 62 Mailboxes at a time even though multiple channels function as a single channel.
3. Do not set the same transmission message ID in both RCAN0 and RCAN1.

   Two messages may be transmitted from the two channels after arbitration on the CAN bus.

14.10.2 Module Stop Mode

The clock supply to RCAN-TL1 can be stopped or started by using the Module Stop Control Register (MSTPCRE). With the initial value, the clock supply is stopped. Access to the RCAN-TL1 registers should be made only after releasing RCAN-TL1 from module stop mode.

14.10.3 Reset

RCAN-TL1 can be reset by hardware reset or software reset.

- Hardware reset
  RCAN-TL1 is reset to the initial state by power-on reset or on entering module stop mode or software standby mode.
- Software reset
  By setting the MCR0 bit in Master Control Register (MCR), RCAN-TL1 registers, excluding the MCR0 bit, and the CAN communication circuitry are initialized.

Since the IRR0 bit in Interrupt Request Register (IRR) is set by the initialization upon reset, it should be cleared while RCAN-TL1 is in configuration mode during the reset sequence.

The areas except for message control field 1 (CONTROL1), timestamp (TIMESTAMP), transmit trigger time (TTT), and time trigger control (TTCONTROL) of mailboxes are not initialized by reset because they are in RAM. After power-on reset, all mailboxes should be initialized while RCAN-TL1 is in configuration mode during the reset sequence.

14.10.4 CAN Sleep Mode

In CAN sleep mode, the clock supply to the major parts in the module is stopped. Therefore, do not make access in CAN sleep mode except for access to the MCR, GSR, IRR, and IMR registers.

14.10.5 Register Access

If the mailbox area is accessed while the CAN communication circuitry in RCAN-TL1 is storing a received CAN bus frame in a mailbox, a 0 to five peripheral clock cycles of wait state is generated.

14.10.6 Interrupts

As shown in table 14.2, a Mailbox 0 receive interrupt can activate the DMAC. If configured such that the DMAC is activated by a Mailbox 0 receive interrupt and clearing of the interrupt source flag upon DMA transfer is enabled, use block transfer mode and read the whole Mailbox 0 message up to the timestamp (TIMESTAMP).

When clearing of the interrupt source flags shown in table 14.2 is by the CPU, read the flag after clearing it in the interrupt service routine. This is necessary to prevent the CPU from executing the RTE instruction during the period after clearing of the interrupt source flag until clearing of the interrupt within the interrupt controller.

Section 15 Renesas Serial Peripheral Interface (RSPI)

This LSI includes the Renesas serial peripheral interface (RSPI) consisting of four independent channels.

The RSPI is capable of full-duplex synchronous serial communications and transmit-only operations. It has a function for performing high-speed serial communications with multiple processors and peripheral devices.

15.1 Features

The RSPI of this LSI has the following features.

- RSPI transfer functions

  Use of MOSI (master out/slave in), MISO (master in/slave out), SSL (slave select), and RSPCK (RSPI clock) signals allows serial communications through SPI operation (four-wire method) or clock synchronous operation (three-wire method).

  Capable of transmit-only operations

  Capable of serial communications in master/slave mode

  Supports mode fault error detection

  Supports overrun error detection

  Switching of the polarity of the serial transfer clock

  Switching of the clock phase of serial transfer
- Data format

  MSB-first/LSB-first selectable

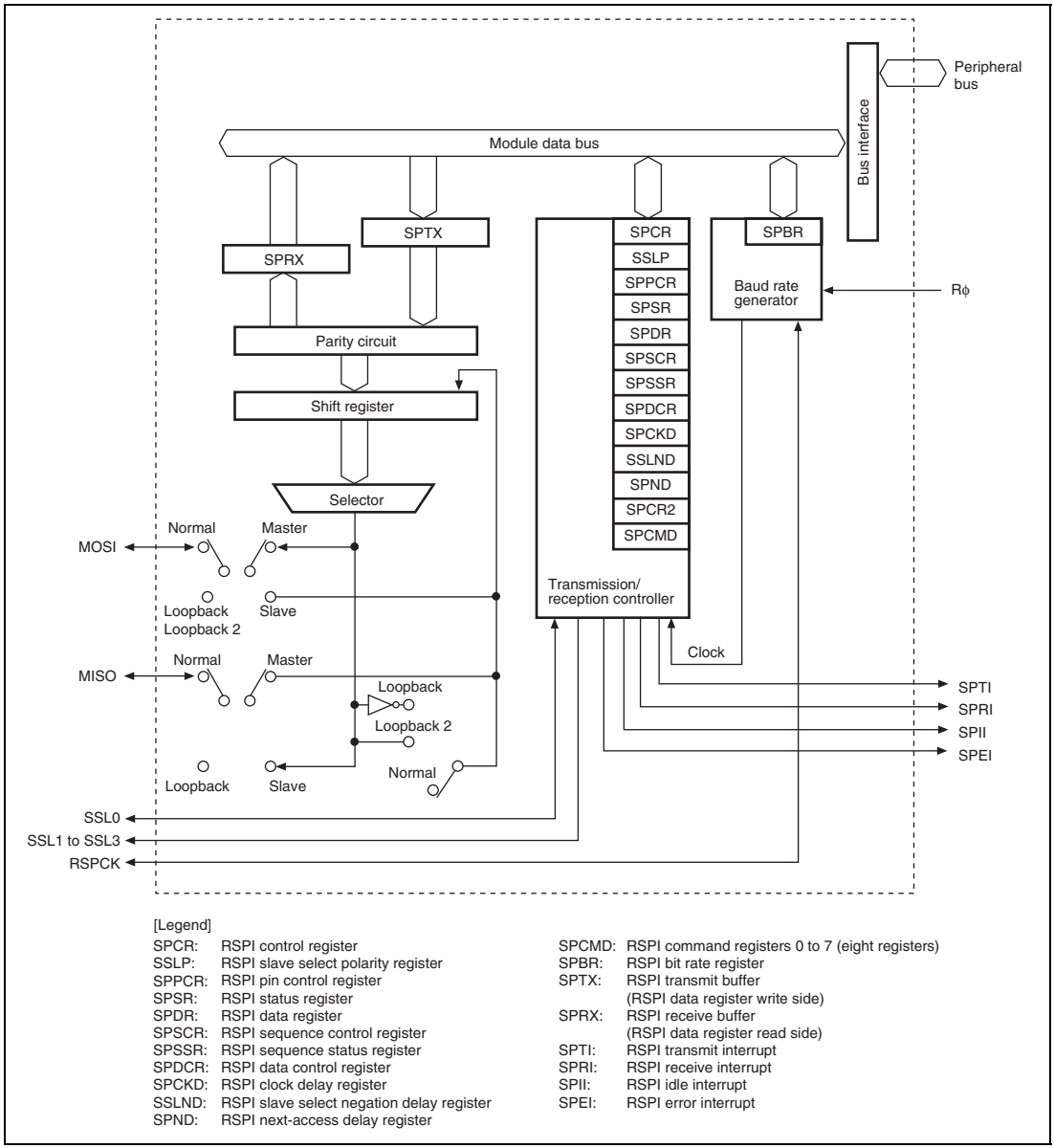
  Transfer bit length is selectable as 8, 9, 10, 11, 12, 13, 14, 15, 16, 20, 24, or 32 bits.

  128-bit transmit/receive buffers

  Up to four frames can be transferred in one round of transmission/reception (each frame consisting of up to 32 bits).
- Buffer configuration

  Double buffer configuration for the transmit/receive buffers

- SSL control function
  Four SSL signals (SSL0 to SSL3) for each channel of RSPI
  In single-master mode, outputs SSL0 to SSL3 signals
  In multi-master mode, SSL0 signal for input, and SSL1 to SSL3 signals for either output or Hi-Z.
  In slave mode, SSL0 signal for input, and SSL1 to SSL3 signals for Hi-Z.
  Controllable delay from SSL output assertion to RSPCK operation (RSPCK delay)
  Range: 1 to 8 RSPCK cycles (set in RSPCK-cycle units)
  Controllable delay from RSPCK stoppage to SSL output negation (SSL negation delay)
  Range: 1 to 8 RSPCK cycles (set in RSPCK-cycle units)
  Controllable wait for next-access SSL output assertion (next-access delay)
  Range: 1 to 8 RSPCK cycles (set in RSPCK-cycle units)
  Function for changing SSL polarity
- Control in master transfer
  A transfer of up to eight commands can be executed sequentially in looped execution.
  For each command, the following can be set:
  SSL signal value, bit rate, RSPCK polarity/phase, transfer data length, LSB/MSB first, burst, RSPCK delay, SSL negation delay, and next-access delay.
  A transfer can be initiated by writing to the transmit buffer.
  A transfer can be initiated by the CPU clearing the SPTEF bit.
  MOSI signal value specifiable in SSL negation
- Interrupt sources
  Maskable interrupt sources:
  RSPI receive interrupt (receive buffer full)
  RSPI transmit interrupt (transmit buffer empty)
  RSPI error interrupts (mode fault, overrun, and parity error)
  RSPI idle interrupt (RSPI idle)
- Others
  Provides loopback mode.
  Provides a function for switching between CMOS output and open-drain output.
  Provides a function for disabling (initializing) the RSPI.



[Legend]

SPCR:	RSPI control register	SPCMD:	RSPI command registers 0 to 7 (eight registers)
SSLP:	RSPI slave select polarity register	SPBR:	RSPI bit rate register
SPPCR:	RSPI pin control register	SPTX:	RSPI transmit buffer (RSPI data register write side)
SPSR:	RSPI status register	SPRX:	RSPI receive buffer (RSPI data register read side)
SPDR:	RSPI data register	SPTI:	RSPI transmit interrupt
SPSCR:	RSPI sequence control register	SPRI:	RSPI receive interrupt
SPSSR:	RSPI sequence status register	SPII:	RSPI idle interrupt
SPDCR:	RSPI data control register	SPEI:	RSPI error interrupt
SPCKD:	RSPI clock delay register		
SSLND:	RSPI slave select negation delay register		
SPND:	RSPI next-access delay register		

Figure 15.1 Block Diagram of RSPI (for One Channel)

15.2 Input/Output Pins

The serial pins listed in table 15.1 are provided for each channel of the RSPI. The RSPI automatically switches the input/output direction of the SSL0 pin. SSL0 is set as an output when the RSPI is a single master and as an input when the RSPI is a multi-master or a slave. Pins RSPCK, MOSI, and MISO are automatically set as inputs or outputs according to the setting of master or slave and the level input on SSL0 (see section 15.4.2, Controlling RSPI Pins).

Table 15.1 Pin Configuration

Channel	Pin Name	Pin Name	I/O	Function
A	RSPI clock pin	RSPCKA	I/O	RSPI clock input/output
	Master transmit data pin	MOSIA	I/O	RSPI master transmit data
	Slave transmit data pin	MISOA	I/O	RSPI slave transmit data
	Slave select 0 pin	SSLA0	I/O	RSPI slave selection
	Slave select 1 pin	SSLA1	Output	RSPI slave selection
	Slave select 2 pin	SSLA2	Output	RSPI slave selection
	Slave select 3 pin	SSLA3	Output	RSPI slave selection
B	RSPI clock pin	RSPCKB	I/O	RSPI clock input/output
	Master transmit data pin	MOSIB	I/O	RSPI master transmit data
	Slave transmit data pin	MISOB	I/O	RSPI slave transmit data
	Slave select 0 pin	SSLB0	I/O	RSPI slave selection
	Slave select 1 pin	SSLB1	Output	RSPI slave selection
	Slave select 2 pin	SSLB2	Output	RSPI slave selection
	Slave select 3 pin	SSLB3	Output	RSPI slave selection
C	RSPI clock pin	RSPCKC	I/O	RSPI clock input/output
	Master transmit data pin	MOSIC	I/O	RSPI master transmit data
	Slave transmit data pin	MISOC	I/O	RSPI slave transmit data
	Slave select 0 pin	SSLC0	I/O	RSPI slave selection
	Slave select 1 pin	SSLC1	Output	RSPI slave selection
	Slave select 2 pin	SSLC2	Output	RSPI slave selection
	Slave select 3 pin	SSLC3	Output	RSPI slave selection

Channel	Pin Name	Pin Name	I/O	Function
D	RSPI clock pin	RSPCKD	I/O	RSPI clock input/output
	Master transmit data pin	MOSID	I/O	RSPI master transmit data
	Slave transmit data pin	MISOD	I/O	RSPI slave transmit data
	Slave select 0 pin	SSLD0	I/O	RSPI slave selection
	Slave select 1 pin	SSLD1	Output	RSPI slave selection
	Slave select 2 pin	SSLD2	Output	RSPI slave selection
	Slave select 3 pin	SSLD3	Output	RSPI slave selection

Note: In the description of the pins, the channel is omitted and pin names are described as RSPICK, MOSI, MISO, and SSL0 to SSL3.

15.3 Register Descriptions

The RSPI has the registers shown in table 15.2. These registers enable the RSPI to perform the following controls: specifying master/slave modes, specifying a transfer format, and controlling the transmitter and receiver.

Table 15.2 Register Configuration

Channel	Register Name	Abbreviation*1	R/W	Initial Value	Address	Access Size
A	RSPI control register A	SPCRA	R/W	H'00	H'FFD800	8, 16
	RSPI slave select polarity register A	SSLPA	R/W	H'00	H'FFD801	8, 16
	RSPI pin control register A	SPPCRA	R/W	H'00	H'FFD802	8, 16
	RSPI status register A	SPSRA	R/(W)*2	H'20	H'FFD803	8, 16
	RSPI data register A	SPDRA	R/W	H'00000000	H'FFD804	16, 32
	RSPI sequence control register A	SPSCRA	R/W	H'00	H'FFD808	8, 16
	RSPI sequence status register A	SPSSRA	R	H'00	H'FFD809	8, 16
	RSPI bit rate register A	SPBRA	R/W	H'FF	H'FFD80A	8, 16
	RSPI data control register A	SPDCRA	R/W	H'00	H'FFD80B	8, 16
	RSPI clock delay register A	SPCKDA	R/W	H'00	H'FFD80C	8, 16
	RSPI slave select negation delay register A	SSLNDA	R/W	H'00	H'FFD80D	8, 16

Channel	Register Name	Abbreviation*1	R/W	Initial Value	Address	Access Size
A	RSPI next-access delay register A	SPNDA	R/W	H'00	H'FFD80E	8, 16
	RSPI control register 2 A	SPCR2A	R/W	H'00	H'FFD80F	8, 16
	RSPI command register A0	SPCMDA0	R/W	H'070D	H'FFD810	16
	RSPI command register A1	SPCMDA1	R/W	H'070D	H'FFD812	16
	RSPI command register A2	SPCMDA2	R/W	H'070D	H'FFD814	16
	RSPI command register A3	SPCMDA3	R/W	H'070D	H'FFD816	16
	RSPI command register A4	SPCMDA4	R/W	H'070D	H'FFD818	16
	RSPI command register A5	SPCMDA5	R/W	H'070D	H'FFD81A	16
	RSPI command register A6	SPCMDA6	R/W	H'070D	H'FFD81C	16
	RSPI command register A7	SPCMDA7	R/W	H'070D	H'FFD81E	16
B	RSPI control register B	SPCRB	R/W	H'00	H'FFD820	8, 16
	RSPI slave select polarity register B	SSLPB	R/W	H'00	H'FFD821	8, 16
	RSPI pin control register B	SPPCRB	R/W	H'00	H'FFD822	8, 16
	RSPI status register B	SPSRB	R/(W)*2	H'20	H'FFD823	8, 16
	RSPI data register B	SPDRB	R/W	H'00000000	H'FFD824	16, 32
	RSPI sequence control register B	SPSCRB	R/W	H'00	H'FFD828	8, 16
	RSPI sequence status register B	SPSSRB	R	H'00	H'FFD829	8, 16
	RSPI bit rate register B	SPBRB	R/W	H'FF	H'FFD82A	8, 16
	RSPI data control register B	SPDCRB	R/W	H'00	H'FFD82B	8, 16
	RSPI clock delay register B	SPCKDB	R/W	H'00	H'FFD82C	8, 16
	RSPI slave select negation delay register B	SSLNDB	R/W	H'00	H'FFD82D	8, 16
	RSPI next-access delay register B	SPNDB	R/W	H'00	H'FFD82E	8, 16
	RSPI control register 2 B	SPCR2B	R/W	H'00	H'FFD82F	8, 16
	RSPI command register B0	SPCMDB0	R/W	H'070D	H'FFD830	16
	RSPI command register B1	SPCMDB1	R/W	H'070D	H'FFD832	16
	RSPI command register B2	SPCMDB2	R/W	H'070D	H'FFD834	16

Channel	Register Name	Abbreviation*1	R/W	Initial Value	Address	Access Size
B	RSPI command register B3	SPCMDB3	R/W	H'070D	H'FFD836	16
	RSPI command register B4	SPCMDB4	R/W	H'070D	H'FFD838	16
	RSPI command register B5	SPCMDB5	R/W	H'070D	H'FFD83A	16
	RSPI command register B6	SPCMDB6	R/W	H'070D	H'FFD83C	16
	RSPI command register B7	SPCMDB7	R/W	H'070D	H'FFD83E	16
C	RSPI control register C	SPCRC	R/W	H'00	H'FFD840	8, 16
	RSPI slave select polarity register C	SSLPC	R/W	H'00	H'FFD841	8, 16
	RSPI pin control register C	SPPCRC	R/W	H'00	H'FFD842	8, 16
	RSPI status register C	SPSRC	R/(W)*2	H'20	H'FFD843	8, 16
	RSPI data register C	SPDRC	R/W	H'00000000	H'FFD844	16, 32
	RSPI sequence control register C	SPSCRC	R/W	H'00	H'FFD848	8, 16
	RSPI sequence status register C	SPSSRC	R	H'00	H'FFD849	8, 16
	RSPI bit rate register C	SPBRC	R/W	H'FF	H'FFD84A	8, 16
	RSPI data control register C	SPDCRC	R/W	H'00	H'FFD84B	8, 16
	RSPI clock delay register C	SPCKDC	R/W	H'00	H'FFD84C	8, 16
	RSPI slave select negation delay register C	SSLNDC	R/W	H'00	H'FFD84D	8, 16
	RSPI next-access delay register C	SPNDC	R/W	H'00	H'FFD84E	8, 16
	RSPI control register 2 C	SPCR2C	R/W	H'00	H'FFD84F	8, 16
	RSPI command register C0	SPCMDC0	R/W	H'070D	H'FFD850	16
	RSPI command register C1	SPCMDC1	R/W	H'070D	H'FFD852	16
	RSPI command register C2	SPCMDC2	R/W	H'070D	H'FFD854	16
	RSPI command register C3	SPCMDC3	R/W	H'070D	H'FFD856	16
	RSPI command register C4	SPCMDC4	R/W	H'070D	H'FFD858	16
	RSPI command register C5	SPCMDC5	R/W	H'070D	H'FFD85A	16
	RSPI command register C6	SPCMDC6	R/W	H'070D	H'FFD85C	16
	RSPI command register C7	SPCMDC7	R/W	H'070D	H'FFD85E	16

Channel	Register Name	Abbreviation*1	R/W	Initial Value	Address	Access Size
D	RSPI control register D	SPCRD	R/W	H'00	H'FFD860	8, 16
	RSPI slave select polarity register D	SSLPD	R/W	H'00	H'FFD861	8, 16
	RSPI pin control register D	SPPCRD	R/W	H'00	H'FFD862	8, 16
	RSPI status register D	SPSRD	R/(W)*2	H'20	H'FFD863	8, 16
	RSPI data register D	SPDRD	R/W	H'00000000	H'FFD864	16, 32
	RSPI sequence control register D	SPSCRD	R/W	H'00	H'FFD868	8, 16
	RSPI sequence status register D	SPSSRD	R	H'00	H'FFD869	8, 16
	RSPI bit rate register D	SPBRD	R/W	H'FF	H'FFD86A	8, 16
	RSPI data control register D	SPDCRD	R/W	H'00	H'FFD86B	8, 16
	RSPI clock delay register D	SPCKDD	R/W	H'00	H'FFD86C	8, 16
	RSPI slave select negation delay register D	SSLNDD	R/W	H'00	H'FFD86D	8, 16
	RSPI next-access delay register D	SPNDD	R/W	H'00	H'FFD86E	8, 16
	RSPI control register 2 D	SPCR2D	R/W	H'00	H'FFD86F	8, 16
	RSPI command register D0	SPCMDD0	R/W	H'070D	H'FFD870	16
	RSPI command register D1	SPCMDD1	R/W	H'070D	H'FFD872	16
	RSPI command register D2	SPCMDD2	R/W	H'070D	H'FFD874	16
	RSPI command register D3	SPCMDD3	R/W	H'070D	H'FFD876	16
	RSPI command register D4	SPCMDD4	R/W	H'070D	H'FFD878	16
	RSPI command register D5	SPCMDD5	R/W	H'070D	H'FFD87A	16
	RSPI command register D6	SPCMDD6	R/W	H'070D	H'FFD87C	16
	RSPI command register D7	SPCMDD7	R/W	H'070D	H'FFD87E	16

Notes: 1. In the description of the register names, the channel is omitted.
2. Only 0 can be written to clear the flag.

15.3.1 RSPI Control Register (SPCR)

SPCR sets the operating mode of the RSPI. SPCR can be read from or written to by the CPU. If the MSTR, MODFEN, and TXMD bits are changed while the RSPI function is enabled by setting the SPE bit to 1, subsequent operations cannot be guaranteed.

Bit	7	6	5	4	3	2	1	0
Bit Name	SPRIE	SPE	SPTIE	SPEIE	MSTR	MODFEN	TXMD	SPMS
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	SPRIE	0	R/W	RSPI Receive Interrupt Enable If the RSPI has detected a receive buffer write after completion of a serial transfer and the SPRF bit in the RSPI status register (SPSR) is set to 1, this bit enables or disables the generation of an RSPI receive interrupt request. 0: Disables the generation of RSPI receive interrupt requests. 1: Enables the generation of RSPI receive interrupt requests.
6	SPE	0	R/W	RSPI Function Enable Setting this bit to 1 enables the RSPI function. When the MODF bit in the RSPI status register (SPSR) is 1, the SPE bit cannot be set to 1 (see section 15.4.8, Error Detection). Setting the SPE bit to 0 disables the RSPI function, and initializes a part of the module function (see section 15.4.9, Initializing RSPI). 0: Disables the RSPI function 1: Enables the RSPI function

Bit	Bit Name	Initial Value	R/W	Description
5	SPTIE	0	R/W	RSPI Transmit Interrupt Enable Enables or disables the generation of RSPI transmit interrupt requests when the RSPI detects transmit buffer empty and sets the SPTEF bit in the RSPI status register (SPSR) to 1. In the RSPI disabled (with the SPE bit 0) status, the SPTEF bit is 1. Therefore, note that setting the SPTIE bit to 1 when the RSPI is in the disabled status generates an RSPI transmit interrupt request. 0: Disables the generation of RSPI transmit interrupt requests. 1: Enables the generation of RSPI transmit interrupt requests.
4	SPEIE	0	R/W	RSPI Error Interrupt Enable Enables or disables the generation of RSPI error interrupt requests when the RSPI detects a mode fault error and sets the MODF bit in the RSPI status register (SPSR) to 1, or when the RSPI detects and sets the OVRF bit in SPSR to 1 (see section 15.4.8, Error Detection). 0: Disables the generation of RSPI error interrupt requests. 1: Enables the generation of RSPI error interrupt requests.
3	MSTR	0	R/W	RSPI Master/Slave Mode Select Selects master/slave mode of RSPI. According to MSTR bit settings, the RSPI determines the direction of pins RSPCK, MOSI, MISO, and SSL1 to SSL3. 0: Slave mode 1: Master mode
2	MODFEN	0	R/W	Mode Fault Error Detection Enable Enables or disables the detection of mode fault error (see section 15.4.8, Error Detection). In addition, the RSPI determines the input/output directions of the SSL0 pin based on combinations of the MODFEN and MSTR bits (see section 15.4.2, Controlling RSPI Pins). 0: Disables the detection of mode fault error 1: Enables the detection of mode fault error

Bit	Bit Name	Initial Value	R/W	Description
1	TXMD	0	R/W	Communication Operating Mode Select Selects full-duplex synchronous serial communications or transmit-only operations. When communications is performed with the TXMD bit set to 1, only transmit operations are performed and receive operations are not performed (see section 15.4.6, Communicaton Operating Mode). When the TXMD bit is set to 1, a receive buffer full interrupt request cannot be generated. 0: Full-duplex synchronous serial communications 1: Transmit-only operations
0	SPMS	0	R/W	RSPI Mode Select Selects SPI operation (four-wire method) or clock synchronous operation (three-wire method). The SSL pins are not used in clock synchronous operation. The three pins RSPCK, MOSI, and MISO handle communications. If clock-synchronous operation is to proceed in master mode (MSTR = 1), the CPHA bit in the RSPI command register (SPCMD) can be set to either 0 or 1. Set the CPHA bit to 1 if clock-synchronous operation is to proceed in slave mode (MSTR = 0). Operation is not guaranteed if CPHA is set to 0 when clock-synchronous operation is to proceed in slave mode (MSTR = 0). 0: SPI operation (four-wire method) 1: Clock synchronous operation (three-wire method)

15.3.2 RSPI Slave Select Polarity Register (SSLP)

SSLP sets the polarity of the SSL0 to SSL3 signals of the RSPI. SSLP can be read from or written to by the CPU. If the contents of SSLP are changed by the CPU while the RSPI function is enabled by setting the SPE bit in the RSPI control register (SPCR) to 1, subsequent operations are not guaranteed.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	SSL3P	SSL2P	SSL1P	SSL0P
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 4	—	All 0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.
3	SSL3P	0	R/W	SSL Signal Polarity Setting These bits set the polarity of the SSL signals. The setting of SSLiP (i = 3 to 0) indicates the active polarity of the SSLi signal. 0: SSLi signal 0-active 1: SSLi signal 1-active
2	SSL2P	0	R/W	
1	SSL1P	0	R/W	
0	SSL0P	0	R/W	

15.3.3 RSPI Pin Control Register (SPPCR)

SPPCR sets the modes of the RSPI pins. SPPCR can be read from or written to by the CPU. If the contents of this register are changed by the CPU while the RSPI function is enabled by setting the SPE bit in the RSPI control register (SPCR) to 1, subsequent operations cannot be guaranteed.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	MOIFE	MOIFV	—	SPOM	SPLP2	SPLP
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R/W	R/W	R	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7, 6	—	All 0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.
5	MOIFE	0	R/W	MOSI Idle Value Fixing Enable Fixes the MOSI output value when the RSPI in master mode is in an SSL negation period (including the SSL retention period during a burst transfer). When MOIFE is 0, the RSPI outputs the last data from the previous serial transfer during the SSL negation period. When MOIFE is 1, the RSPI outputs the fixed value set in the MOIFV bit to the MOSI bit. 0: MOSI output value equals final data from previous transfer 1: MOSI output value equals the value set in the MOIFV bit
4	MOIFV	0	R/W	MOSI Idle Fixed Value If the MOIFE bit is 1 in master mode, the RSPI, according to MOIFV bit settings, determines the MOSI signal value during the SSL negation period (including the SSL retention period during a burst transfer). 0: MOSI idle fixed value equals 0 1: MOSI idle fixed value equals 1

Bit	Bit Name	Initial Value	R/W	Description
3	—	0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.
2	SPOM	0	R/W	RSPI Output Pin Mode Sets the RSPI output pins to CMOS output/open drain output. 0: CMOS output 1: Open-drain output
1	SPLP2	0	R/W	RSPI Loopback 2 When the SPLP2 bit is set to 1, the RSPI shuts off the path between the MISO pin and the shift register, and between the MOSI pin and the shift register, and connects the input path and the output path for the shift register (loopback mode). 0: Normal mode 1: Loopback mode (transmit data = received data)
0	SPLP	0	R/W	RSPI Loopback When the SPLP bit is set to 1, the RSPI shuts off the path between the MISO pin and the shift register, and between the MOSI pin and the shift register, and connects (reverses) the input path and the output path for the shift register. 0: Normal mode 1: Loopback mode (reversed transmit data = received data)

15.3.4 RSPI Status Register (SPSR)

SPSR indicates the operating status of the RSPI. SPSR can be read by the CPU. Writing to SPSR can only be performed by the CPU under certain conditions.

Bit	7	6	5	4	3	2	1	0
Bit Name	SPRF	—	SPTEF	—	PERF	MODF	IDLNF	OVRF
Initial Value:	0	0	1	0	0	0	0	0
R/W:	R/(W)*	R	R/(W)*	R	R/(W)*	R/(W)*	R	R/(W)*

Note: * Only 0 can be written to clear the flag after reading 1.

Bit	Bit Name	Initial Value	R/W	Description
7	SPRF	0	R/(W)*	RSPI Receive Buffer Full Flag Indicates the status of the receive buffer for the RSPI data register (SPDR). If a serial transfer ends while the communication operating mode select bit (TXMD) in the RSPI control register (SPCR) is 0 and the SPRF bit is 0, the RSPI transfers the receive data from the shift register to SPDR, and sets this bit to 1. As the RSPI handles full-duplex synchronous serial communications when the TXMD bit is 0, this means that the last bit of transmit data has been transmitted. The SPRF bit is cleared to 0 under the following conditions: • The CPU reads SPSR when the SPRF bit is 1, and then the CPU writes a 0 to the SPRF bit. • Received data is read from the SPDR. • System reset If a serial transfer ends while the SPRF bit is 1, the RSPI does not transfer the received data from the shift register to the SPDR. When the OVRF bit in SPSR is 1, the SPRF bit cannot be changed from 0 to 1 (see section 15.4.8, Error Detection). 0: No valid data in SPDR 1: Valid data found in SPDR

Bit	Bit Name	Initial Value	R/W	Description
6	—	0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.
5	SPTEF	1	R/(W)*	RSPI Transmit Buffer Empty Flag Indicates the status of the transmit buffer for the RSPI data register (SPDR). After the initialization of RSPI or after transmit data is transferred from the transmit buffer to the shift register, the RSPI sets the SPTEF bit to 1. The SPTEF bit is cleared to 0 under the following conditions. If the SPTEF bit is cleared and the shift register is empty, the data is copied from the transmit buffer to the shift register. • The CPU reads SPRF when the SPTEF bit is 1, and then the CPU writes 0 to the SPTEF bit. • The transmit data is written to SPDR. Data can be written to SPDR only when the SPTEF bit is 1. If data is written to the transmit buffer of SPDR when the SPTEF bit is 0, the data in the transmit buffer is not updated. 0: Data found in the transmit buffer 1: No data in the transmit buffer
4	—	0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.
3	PERF	0	R/(W)*	Parity Error Flag Indicates the occurrence of a parity error. If a serial transfer ends while the TXMD bit in SPCR is 0 and the SPPE bit in SPCR2 is 1, the RSPI detects a parity error, and sets the PERF bit to 1. The PERF bit is cleared to 0 under the following conditions. • The CPU reads SPSR when the PERF bit is 1, and then writes 0 to the PERF bit. • System reset 0: No parity error occurs 1: A parity error occurs

Bit	Bit Name	Initial Value	R/W	Description
2	MODF	0	R/(W)*	Mode Fault Error Flag Indicates the occurrence of a mode fault error. When the input level of the SSL0 pin changes to the active level while the MSTR bit in the RSPI control register (SPCR) is 1 and the MODFEN bit is 1 with the RSPI being in multi-master mode, the RSPI detects a mode fault error and sets the MODF bit to 1. Similarly, if the MODFEN bit is set to 1 when the MSTR bit is 0 and the RSPI is in slave mode, and the SSL0 pin is negated before the RSPCK cycle necessary for data transfer ends, the RSPI detects a mode fault error. The active level of the SSL0 signal is determined by the SSL0P bit in the RSPI slave select polarity register (SSLP). The MODF bit is cleared to 0 under the following conditions. • The CPU reads SPSR when the MODF bit is 1, and then writes 0 to the MODF bit. • System reset 0: No mode fault error occurs 1: A mode fault error occurs

Bit	Bit Name	Initial Value	R/W	Description
1	IDLNF	0	R	RSPI Idle Flag Indicates the transfer status of the RSPI. In master mode, if the SPCP bits in SPSSR are 000 (at the beginning of the sequence) while the SPTEF bit in SPSR is 1, data for the next transfer is not set and the RSPI will not perform transfer, thus this bit is cleared to 0. In both master mode and slave mode, if the SPE bit in SPCR is 0 (RSPI function is disabled), this bit is cleared to 0. In master mode (single-master or multi-master mode), the IDLNF bit is cleared to 0 under the following conditions. • SPE bit in SPCR is 0 (RSPI is initialized) • All of the conditions below are satisfied — SPTEF bit in SPSR is 1 (data for the next transfer is not set) — SPCP bits in SPSSR are 000 (sequence control is at the top command pointer in the loop) — RSPI internal sequencer has entered the idle state (operation up to next-access delay is completed) The IDLNF bit is set to 1 when the above conditions are not satisfied. In slave mode, the IDLNF bit is cleared to 0 under the following condition. • SPE bit in SPCR is 0 (RSPI is initialized) When the SPE bit is set to 1, the IDLNF bit is set to 1.

Bit	Bit Name	Initial Value	R/W	Description
0	OVRF	0	R/(W)*	Overrun Error Flag Indicates the occurrence of an overrun error. If a serial transfer ends while the communication operating mode select bit (TXMD) in the RSPI control register (SPCR) is 0 and the SPRF bit is 1, the RSPI detects an overrun error, and sets the OVRF bit to 1. The OVRF bit is cleared to 0 under the following conditions. • The CPU reads SPSR when the OVRF bit is 1, and then writes 0 to the OVRF bit. • System reset 0: No overrun error occurs 1: An overrun error occurs

Note: * Only 0 can be written to clear the flag after reading 1.

15.3.5 RSPI Data Register (SPDR)

SPDR is a buffer that holds data for transmission and reception by the RSPI.

The transmit buffer (SPTX) and receive buffer (SPRX) are independent and are mapped to SPDR.

When reading from or writing to SPDR, set the RSPI longword access/word access specification bit (SPLW) in the RSPI data control register (SPDCR) according to whether access is to be in longword or word units. When SPLW is 0, SPDR is a 64-bit buffer that consists of up to four 16-bit frames. When SPLW is 1, SPDR is a 128-bit buffer that consists of up to four 32-bit frames.

The frame length used by SPDR is determined by the number of frames specification bits (SPFC1 and SPFC0) in the RSPI data control register (SPDCR), and the bit length to be used is determined by the RSPI data length specification bits (SPB3 to SPB0) in the RSPI command register (SPCMD).

If the SPTEF bit in the RSPI status register (SPSR) is 1 when data is written to SPDR, the RSPI allows writing of the data to the SPDR transmit buffer. If the SPTEF bit is 0, the RSPI does not allow updating of the SPDR transmit buffer.

If the RSPI receive/transmit data selection bit (SPRDTD) in the RSPI data control register (SPDCR) is 0 when data is read from SPDR, the RSPI allows reading of the receive buffer. If SPPCR is 1, the RSPI allows reading of the transmit buffer.

When reading from the transmit buffer, the value written to the buffer immediately before the read operation is read. Reading from the transmit buffer is not possible while the SPTEF bit in the RSPI status register (SPSR) is 0.

In normal operation method, data is read from the receive buffer when the SPRDTD bit is 0, and the SPRF bit in SPSR is 1 (a condition in which the receive buffer holds data that has not yet been read out). When the SPRF bit or the OVRF bit in SPSR is 1, the RSPI does not update the receive buffer of SPSR at the end of a serial transfer.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Bit Name	SPD31	SPD30	SPD29	SPD28	SPD27	SPD26	SPD25	SPD24	SPD23	SPD22	SPD21	SPD20	SPD19	SPD18	SPD17	SPD16
Initial Value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Name	SPD15	SPD14	SPD13	SPD12	SPD11	SPD10	SPD9	SPD8	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0
Initial Value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

15.3.6 RSPI Sequence Control Register (SPSCR)

SPSCR sets the sequence control method when the RSPI operates in master mode. SPSCR can be read from or written to by the CPU. If the contents of SPSCR are changed by the CPU while the MSTR and SPE bits in the RSPI control register (SPCR) are 1 with the RSPI function enabled, the subsequent operation cannot be guaranteed.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	SPSLN2	SPSLN1	SPSLN0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 3	—	All 0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.

Bit	Bit Name	Initial Value	R/W	Description
2 to 0	SPSLN2 to SPSLN0	All 0	R/W	RSPI Sequence Length Specification These bits specify a sequence length when the RSPI in master mode performs sequential operations. The RSPI in master mode changes RSPI command registers 0 to 7 (SPCMD0 to SPCMD7) to be referenced and the order in which they are referenced according to the sequence length that is set in the SPSLN2 to SPSLN0 bits. The relationship among the setting of bits SPSLN2 to SPSLN0, sequence length, and SPCMD0 to SPCMD7 referenced by the RSPI is shown below. Note that in slave mode, the RSPI always references SPCMD0.

	Sequence Length	Referenced SPCMD (No.)
000	1	0 → 0 → …
001	2	0 → 1 → 0 → …
010	3	0 → 1 → 2 → 0 → …
011	4	0 → 1 → 2 → 3 → 0 → …
100	5	0 → 1 → 2 → 3 → 4 → 0 → …
101	6	0 → 1 → 2 → 3 → 4 → 5 → 0 → …
110	7	0 → 1 → 2 → 3 → 4 → 5 → 6 → 0 → …
111	8	0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 0 → …

15.3.7 RSPI Sequence Status Register (SPSSR)

SPSSR indicates the sequence control status when the RSPI operates in master mode. SPSSR can be read by the CPU. Any writing to SPSSR by the CPU is ignored.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	SPECM2	SPECM1	SPECM0	—	SPCP2	SPCP1	SPCP0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R

Bit	Bit Name	Initial Value	R/W	Description
7	—	0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.
6 to 4	SPECM2 to SPECM0	All 0	R	RSPI Error Command These bits indicate RSPI command registers 0 to 7 (SPCMD0 to SPCMD7) that are pointed to by command pointers (SPCP2 to SPCP0 bits) when an error is detected during sequence control by the RSPI. The RSPI updates the SPECM2 to SPECM0 bits only when an error is detected. If both the OVRF and MODF bits in the RSPI status register (SPSR) are 0 and there is no error, the values of the SPECM2 to SPECM0 bits have no meaning. The relationship between the setting of bits SPECM2 to SPECM0 and SPCMD7 to SPCMD0 is shown below. For the RSPI's error detection function, see section 15.4.8, Error Detection. For the RSPI's sequence control, see section 15.4.10 (1), Master Mode Operation. 000: SPCMD0 001: SPCMD1 010: SPCMD2 011: SPCMD3 100: SPCMD4 101: SPCMD5 110: SPCMD6 111: SPCMD7

Bit	Bit Name	Initial Value	R/W	Description
3	—	0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.
2 to 0	SPCP2 to SPCP0	All 0	R	RSPI Command Pointer During RSPI sequence control, these bits indicate RSPI command registers 7 to 0 (SPCMD7 to SPCMD0), which are currently pointed to by the pointers. The relationship between the setting of bits SPCP2 to SPCP0 and SPCMD7 to SPCMD0 is shown below. For the RSPI's sequence control, see section 15.4.10 (1), Master Mode Operation. 000: SPCMD0 001: SPCMD1 010: SPCMD2 011: SPCMD3 100: SPCMD4 101: SPCMD5 110: SPCMD6 111: SPCMD7

15.3.8 RSPI Bit Rate Register (SPBR)

SPBR sets the bit rate in master mode. SPBR can be read from or written to by the CPU. If the contents of SPBR are changed by the CPU while the MSTR and SPE bits in the RSPI control register (SPCR) are 1 with the RSPI function in master mode enabled, operation cannot be guaranteed.

Bit	7	6	5	4	3	2	1	0
Bit Name	SPR7	SPR6	SPR5	SPR4	SPR3	SPR2	SPR1	SPR0
Initial Value:	1	1	1	1	1	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

When the RSPI is used in slave mode, the bit rate depends on the bit rate of the input clock regardless of the settings of SPBR and BRDV.

The bit rate is determined by combinations of SPBR settings and the bit settings in the BRDV1 and BRDV0 bits in the RSPI command registers (SPCMD0 to SPCMD7). The equation for calculating the bit rate is given below. In the equation, n denotes an SPBR setting (0, 1, 2, …, 255), and N denotes bit settings in the bits BRDV1 and BRDV0 (0, 1, 2, 3).

$$\text{Bit rate} = \frac{f\,(R\phi)}{2 \times (n + 1) \times 2^{N}}$$

Table 15.3 shows examples of the relationship between the SPBR register and BRDV1 and BRDV0 bit settings.

Table 15.3 Relationship between SPBR and BRDV1 and BRDV0 Settings

			Bit Rate	
SPBR (n)	**BRDV[1:0] (N)**	**Division Ratio**	**Rϕ = 32 MHz**	**Rϕ = 40 MHz**
0	0	2	16.0 Mbps*	20.0 Mbps*
1	0	4	8.00 Mbps	10.0 Mbps
2	0	6	5.33 Mbps	6.67 Mbps
3	0	8	4.00 Mbps	5.00 Mbps
4	0	10	3.20 Mbps	4.00 Mbps
5	0	12	2.67 Mbps	3.33 Mbps
5	1	24	1.33 Mbps	1.67 Mbps
5	2	48	667 kbps	833 kbps
5	3	96	333 kbps	417 kbps
255	3	4096	7.81 kbps	9.78 kbps

Note: * Cannot be set in this LSI.

15.3.9 RSPI Data Control Register (SPDCR)

SPDCR is a register used to set the number of frames that can be stored in SPDR, control the SSL pin output and reading from SPDR, and select the width (longword or word) for access to SPDR.

Up to four frames can be transmitted or received in one round of transmission or reception activation. The amount of data in each transfer is controlled by the combination of the RSPI data length specification bits (SPB3 to SPB0) in the RSPI command register (SPCMD), the RSPI sequence length specification bits (SPSLN2 to SPSLN0) in the RSPI sequence control register (SPSCR), and the number of frames specification bits (SPFC1 and SPFC0) in the RSPI data control register (SPDCR).

SPDCR can be read from or written to by the CPU. If the contents of SPDCR are changed by the CPU while the SPE bit in the RSPI control register (SPCR) is 1 with the RSPI function enabled, subsequent operation cannot be guaranteed.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	SPLW	SPRDTD	SLSEL1	SLSEL0	SPFC1	SPFC0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7, 6	—	All 0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.
5	SPLW	0	R/W	RSPI Longword Access/Word Access Specification Specifies the access width for the RSPI data register (SPDR). Access to SPDR is in words when SPLW is 0 and in longwords when SPLW is 1. Also, when SPLW is 0, set the RSPI data length specification bits (SPB3 to SPB0) in the RSPI command register (SPCMD) to 8 to 16 bits. When 20, 24, or 32 bit is specified, operation is not guaranteed. 0: SPDR is accessed in words. 1: SPDR is accessed in longwords.

1Bit	Bit Name	Initial Value	R/W	Description
4	SPRDTD	0	R/W	RSPI Receive/Transmit Data Selection Selects whether the RSPI data register (SPDR) reads values from the receive buffer or from the transmit buffer. If reading is from the transmit buffer, the value written to SPDR immediately beforehand is read. Reading should only be from the transmit buffer when the SPTEF bit in the RSPI status register (SPSR) is 1. 0: SPDR reads from the receive buffer. 1: SPDR reads from the transmit buffer (but only if the SPTEF bit is 1).
3	SLSEL1	0	R/W	SSl Pin Output Selection
2	SLSEL0	0	R/W	These bits control the SSL pin output in master mode. Whether the SSL pin is used for general I/O or SSL output is selected according to the combination of these bits as shown below. Operation is not guaranteed if SLSEL1 = SLSEL0 = 1 is set.

	SLSEL1 = 0 SLSEL0 = 0	SLSEL1 = 0 SLSEL0 = 1	SLSEL1 = 1 SLSEL0 = 0	SLSEL1 = 1 SLSEL0 = 1
SSL3	Output	IO	IO	Setting Prohibited
SSL2	Output	IO	IO	
SSL1	Output	IO	Output	
SSL0	Output	Output	Output	

Bit	Bit Name	Initial Value	R/W	Description
1	SPFC1	0	R/W	Number of Frames Specification
0	SPFC0	0	R/W	These bits specify the number of frames that can be stored in SPDR. Up to four frames can be transmitted or received in one round of transmission or reception, and the amount of data is determined by the combination of the RSPI data length specification bits (SPB3 to SPB0) in the RSPI command register (SPCMD), the RSPI sequence length specification bits (SPSLN2 to SPSLN0) in the RSPI sequence control register (SPSCR), and the number of frames specification bits (SPFC1 and SPFC0) in the RSPI data control register (SPDCR). Also, SPFC1 and SPFC0 specify the number of received data at which the RSPI receive buffer full flag (SPRF) in the RSPI status register (SPSR) is set. Table 15.4 and figure 15.2 show the frame configurations that can be stored in SPDR and examples of combinations of settings for transmission and reception. If combinations of settings other than those shown in the examples are made, subsequent operation is not guaranteed.

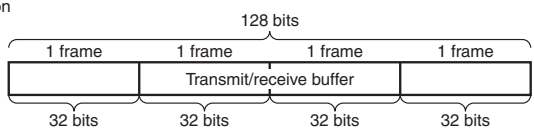
Table 15.4 Frame Settings by All Relevant Bits

Setting	SPB3 to SPB0	SPSLN2 to SPSLN0	SPFC1 and SPFC0	Number of Frames for Transfer	Number of Frames at Which SPRF is Set to 1 and SPTEF is Cleared to 0
1-1	M	000	00	1	1
1-2	M	000	01	2	2
1-3	M	000	10	3	3
1-4	M	000	11	4	4
2-1	M, N	001	01	2	2
2-2	M, N	001	11	4	4
3	M, N, O	010	10	3	3
4	M, N, O, P	011	11	4	4
5	M, N, O, P, Q	100	00	5	1
6	M, N, O, P, Q, R	101	00	6	1
7	M, N, O, P, Q, R, S	110	00	7	1
8	M, N, O, P, Q, R, S, T	111	00	8	1

[Legend]

M, N, O, P, Q, R, S, T: Data length that can be specified by bits SPB3 to SPB0

- Frame configuration



- Examples of combination of settings for transmission and reception
  With one activation, data can be transmitted or received as shown below when settings 1-1 to 8 in table 15.4 are made.

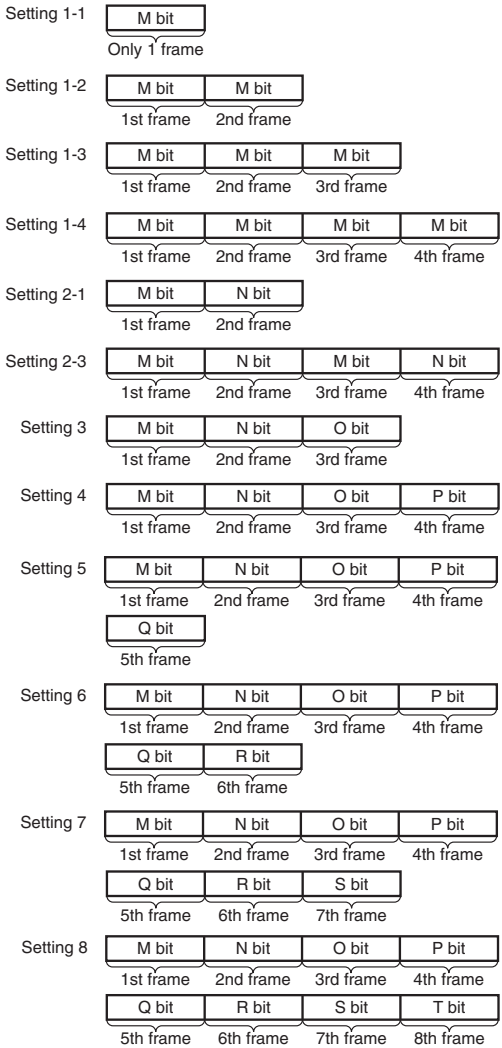


Figure 15.2 Frame Configurations and Examples of Combinations of Transmission and Reception Settings

15.3.10 RSPI Clock Delay Register (SPCKD)

SPCKD sets a period from the beginning of SSL signal assertion to RSPCK oscillation (RSPCK delay) when the SCKDEN bit in the RSPI command register (SPCMD) is 1. SPCKD can be read from or written to by the CPU. If the contents of SPCKD are changed by the CPU while the MSTR and SPE bits in the RSPI control register (SPCR) are 1 with the RSPI function in master mode enabled, operation cannot be guaranteed.

When using the RSPI in slave mode, set B'000 to the SCKDL2 to SCKDL0 bits.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	SCKDL2	SCKDL1	SCKDL0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 3	—	All 0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.
2	SCKDL2	0	R/W	RSPCK Delay Setting
1	SCKDL1	0	R/W	These bits set an RSPCK delay value when the SCKDEN bit in SPCMD is 1.
0	SCKDL0	0	R/W	The relationship between the setting of SCKDL2 to SCKDL0 and the RSPCK delay value is shown below. 000: 1 RSPCK 001: 2 RSPCK 010: 3 RSPCK 011: 4 RSPCK 100: 5 RSPCK 101: 6 RSPCK 110: 7 RSPCK 111: 8 RSPCK

15.3.11 RSPI Slave Select Negation Delay Register (SSLND)

SSLND sets a period (SSL negation delay) from the transmission of a final RSPCK edge to the negation of the SSL signal during a serial transfer by the RSPI in master mode. SSLND can be read from or written to by the CPU. If the contents of SSLND are changed by the CPU while the MSTR and SPE bits in the RSPI control register (SPCR) are 1 with the RSPI function in master mode enabled, operation cannot be guaranteed.

When using the RSPI in slave mode, set B'000 to SLNDL2 to SLNDL0.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	SLNDL2	SLNDL1	SLNDL0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 3	—	All 0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.
2	SLNDL2	0	R/W	SSL Negation Delay Setting
1	SLNDL1	0	R/W	These bits set an SSL negation delay value when the RSPI is in master mode.
0	SLNDL0	0	R/W	The relationship between the setting of SLNDL2 to SLNDL0 and the SSL negation delay value is shown below. 000: 1 RSPCK 001: 2 RSPCK 010: 3 RSPCK 011: 4 RSPCK 100: 5 RSPCK 101: 6 RSPCK 110: 7 RSPCK 111: 8 RSPCK

15.3.12 RSPI Next-Access Delay Register (SPND)

SPND sets a non-active period (next-access delay) after termination of a serial transfer when the SPNDEN bit in the RSPI command register (SPCMD) is 1. SPND can be read from or written to by the CPU. If the contents of SPND are changed by the CPU while the MSTR and SPE bits in the RSPI control register (SPCR) are 1 with the RSPI function in master mode enabled, operation cannot be guaranteed.

When using the RSPI in slave mode, set B'000 to SPNDL2 to SPNDL0.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	SPNDL2	SPNDL1	SPNDL0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 3	—	All 0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.
2	SPNDL2	0	R/W	RSPI Next-Access Delay Setting
1	SPNDL1	0	R/W	These bits set a next-access delay when the SPNDEN bit in SPCMD is 1.
0	SPNDL0	0	R/W	The relationship between the setting of SPNDL2 to SPNDL0 and the next-access delay value is shown below. 000: 1 RSPCK + 2 Rϕ 001: 2 RSPCK + 2 Rϕ 010: 3 RSPCK + 2 Rϕ 011: 4 RSPCK + 2 Rϕ 100: 5 RSPCK + 2 Rϕ 101: 6 RSPCK + 2 Rϕ 110: 7 RSPCK + 2 Rϕ 111: 8 RSPCK + 2 Rϕ

15.3.13 RSPI Control Register 2 (SPCR2)

SPCR2 sets the operating mode of the RSPI. SPCR2 can be read from or written to by the CPU. If the SPOE and SPPE bits are changed while the RSPI function is enabled by setting the SPE bit to 1, subsequent operations cannot be guaranteed.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	PTE	SPIIE	SPOE	SPPE
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 4	—	All 0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.
3	PTE	0	R/W	Parity Self Diagnosis Enables the self diagnosis function of the parity circuit to confirm that the parity function has been enabled correctly. 0: Disables the self diagnosis function of the parity circuit. 1: Enables the self diagnosis function of the parity circuit.
2	SPIIE	0	R/W	RSPI Idle Interrupt Enable If the RSPI being in the idle state is detected and the IDLNF bit in the RSPI status register (SPSR) is set to 1, this bit enables or disables the generation of an RSPI idle interrupt request. 0: Disables the generation of RSPI idle interrupt requests. 1: Enables the generation of RSPI idle interrupt requests.

Bit	Bit Name	Initial Value	R/W	Description
1	SPOE	0	R/W	Parity Mode When even parity is set, parity bit addition is performed so that the total number of 1-bits in the transmit/receive character plus the parity bit is even. In the same manner, when odd parity is set, parity bit addition is performed so that the total number of 1-bits in the transmit/receive character plus the parity bit is odd. The SPOE bit is valid only when the SPPE bit in SPCR2 is 1. 0: Performs transmission/reception with even parity 0: Performs transmission/reception with odd parity
0	SPPE	0	R/W	Parity Enable Enables or disables the parity function. When the communication operating mode select bit (TXMD) in the RSPI control register (SPCR) is 0 and the SPPE bit is 1, the parity bit is added to the transmit data, and the receive data is checked for the parity bit. When the TXMD bit is 1 and the SPPE bit is 1, the parity bit is added to the transmit data, but the receive data is not checked for the parity bit. 0: Parity bit is not added to the transmit data, and the receive data is not checked for the parity bit. 1: When TXMD = 0, the parity bit is added to the transmit data, and the receive data is checked for the parity bit. When TXMD = 1, the parity bit is added to the transmit data, but the receive data is not checked for the parity bit.

15.3.14 RSPI Command Register (SPCMD)

Each channel has eight RSPI command registers (SPCMD0 to SPCMD7). SPCMD0 to SPCMD7 are used to set a transfer format for the RSPI in master mode. Some of the bits in SPCMD0 are used to set a transfer mode for the RSPI in slave mode. The RSPI in master mode sequentially references SPCMD0 to SPCMD7according to the settings in bits SPSLN2 to SPSLN0 in the RSPI sequence control register (SPSCR), and executes the serial transfer that is set in the referenced SPCMD.

SPCMD can be read from or written to by the CPU. SPCMD must be set while the SPTEF bit in the RSPI status register (SPSR) is 1 and also before the data to be transmitted referencing that SPCMD has been set. SPCMD that is referenced by the RSPI in master mode can be checked by means of bits SPCP2 to SPCP0 in the RSPI sequence status register (SPSSR). When the RSPI function in master mode is enabled, operation cannot be guaranteed if the value set in SPCMD0 is changed by the CPU.

Bit	15	14	13	12	11	10	9	8
Bit Name	SCKDEN	SLNDEN	SPNDEN	LSBF	SPB3	SPB2	SPB1	SPB0
Initial Value:	0	0	0	0	0	1	1	1
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	SSLKP	—	SSLA1	SSLA0	BRDV1	BRDV0	CPOL	CPHA
Initial Value:	0	0	0	0	1	1	0	1
R/W:	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
15	SCKDEN	0	R/W	RSPCK Delay Setting Enable Sets the period from the point when the RSPI in master mode activates the SSL signal until the RSPCK starts oscillation (RSPI clock delay). If the SCKDEN bit is 0, the RSPI sets the RSPCK delay to 1 RSPCK. If the SCKDEN bit is 1, the RSPI starts the oscillation of RSPCK at an RSPCK delay in compliance with RSPI clock delay register (SPCKD) settings. To use the RSPI in slave mode, the SCKDEN bit should be set to 0. 0: An RSPCK delay of 1 RSPCK 1: An RSPCK delay is equal to SPCKD settings.
14	SLNDEN	0	R/W	SSL Negation Delay Setting Enable Sets the period (SSL negation delay) from the time the master mode RSPI stops RSPCK oscillation until the RSPI sets the SSL signal inactive. If the SLNDEN bit is 0, the RSPI sets the SSL negation delay to 1 RSPCK. If the SLNDEN bit is 1, the RSPI negates the SSL signal at an SSL negation delay in compliance with slave select negation delay register (SSLND) settings. To use the RSPI in slave mode, the SLNDEN bit should be set to 0. 0: An SSL negation delay of 1 RSPCK 1: An SSL negation delay is equal to SSLND settings.
13	SPNDEN	0	R/W	RSPI Next-Access Delay Enable Sets the period from the time the RSPI in master mode terminates a serial transfer and sets the SSL signal inactive until the RSPI enables the SSL signal assertion for the next access (next-access delay). If the SPNDEN bit is 0, the RSPI sets the next-access delay to 1 RSPCK+ 2 Rϕ. If the SPNDEN bit is 1, the RSPI inserts a next-access delay in compliance with RSPI next-access delay register (SPND) settings. To use the RSPI in slave mode, the SPNDEN bit should be set to 0. 0: A next-access delay of 1 RSPCK + 2 Rϕ 1: A next-access delay is equal to SPND settings.

Bit	Bit Name	Initial Value	R/W	Description
12	LSBF	0	R/W	RSPI LSB First Sets the data format of the RSPI in master mode or slave mode to MSB first or LSB first. 0: MSB first 1: LSB first
11	SPB3	0	R/W	SRPI Data Length Setting These bits set a transfer data length for the RSPI in master mode or slave mode. 0100 to 0111: 8 bits 1000: 9 bits 1001: 10 bits 1010: 11 bits 1011: 12 bits 1100: 13 bits 1101: 14 bits 1110: 15 bits 1111: 16 bits 0000: 20 bits 0001: 24 bits 0010, 0011: 32 bits
10	SPB2	1	R/W	
9	SPB1	1	R/W	
8	SPB0	1	R/W	
7	SSLKP	0	R/W	SSL Signal Level Keeping When the RSPI in master mode performs a serial transfer, this bit specifies whether the SSL signal level for the current command is to be kept or negated between the SSL negation timing associated with the current command and the SSL assertion timing associated with the next command. To use the RSPI in slave mode, the SSLKP bit should be set to 0. 0: Negates all SSL signals upon completion of transfer. 1: Keeps the SSL signal level from the end of the transfer until the beginning of the next access.

Bit	Bit Name	Initial Value	R/W	Description
6	—	0	R	Reserved The write value should always be 0. Otherwise, operation cannot be guaranteed.
5	SSLA1	0	R/W	SSL Signal Assertion Setting
4	SSLA0	0	R/W	These bits control the SSL signal assertion when the RSPI performs serial transfers in master mode. Setting the SSLAi (i = 1 or 0) controls the assertion for the signals SSL3 to SSL0. When an SSL signal is asserted, its polarity is determined by the set value in the corresponding SSLP (RSPI slave select polarity register). When SSLA1 and SSLA0 are set to B'00 in multi-master mode, serial transfers are performed with all the SLL signals in the negated state (as SSL0 acts as input). When using the RSPI in slave mode, set B'00 to SSLA1 and SSLA0. 00: SSL0 01: SSL1 10: SSL2 11: SSL3
3	BRDV1	1	R/W	Bit Rate Division Setting
2	BRDV0	1	R/W	These bits are used to determine the bit rate. A bit rate is determined by combinations of bits BRDV1 and BRDV 0 and the settings in the RSPI bit rate register (SPBR) (see section 15.3.8, RSPI Bit Rate Register (SPBR)). The settings in SPBR determine the base bit rate. The settings in bits BRDV1 and BRDV0 are used to select a bit rate which is obtained by dividing the base bit rate by 1, 2, 4, or 8. In SPCMD0 to SPCMD7, different BRDV1 and BRDV0 settings can be specified. This permits the execution of serial transfers at a different bit rate for each command. 00: Select the base bit rate 01: Select the base bit rate divided by 2 10: Select the base bit rate divided by 4 11: Select the base bit rate divided by 8

Bit	Bit Name	Initial Value	R/W	Description
1	CPOL	0	R/W	RSPCK Polarity Setting Sets an RSPCK polarity of the RSPI in master or slave mode. Data communications between RSPI modules require the same RSPCK polarity setting between the modules. 0: RSPCK = 0 when idle 1: RSPCK = 1 when idle
0	CPHA	1	R/W	RSPCK Phase Setting Sets an RSPCK phase of the RSPI in master or slave mode. Data communications between RSPI modules require the same RSPCK phase setting between the modules. 0: Data sampling on odd edge, data variation on even edge 1: Data variation on odd edge, data sampling on even edge

15.4 Operation

In this section, the serial transfer period means a period from the beginning of driving valid data to the fetching of the final valid data.

15.4.1 Overview of RSPI Operations

The RSPI is capable of synchronous serial transfers in slave mode (SPI operation), single-master mode (SPI operation), multi-master mode (SPI operation), slave mode (clock synchronous operation), and master mode (clock synchronous operation). A particular mode of the RSPI can be selected by using the MSTR, MODFEN, and SPMS bits in the RSPI control register (SPCR). Table 15.5 gives the relationship between RSPI modes and SPCR settings, and a description of each mode.

Table 15.5 Relationship between RSPI Modes and SPCR and Description of Each Mode

Mode	Slave (SPI Operation)	Single-Master (SPI Operation)	Multi-Master (SPI Operation)	Slave (Clock Synchronous Operation)	Master (Clock Synchronous Operation)
MSTR bit setting	0	1	1	0	1
MODFEN bit setting	0 or 1	0	1	0	0
SPMS bit setting	0	0	0	1	1
RSPCK signal	Input	Output	Output/Hi-Z	Input	Output
MOSI signal	Input	Output	Output/Hi-Z	Input	Output
MISO signal	Output/Hi-Z	Input	Input	Output	Input
SSL0 signal	Input	Output	Input	Hi-Z	Hi-Z
SSL1 to SSL3 signals	Hi-Z	Output/Hi-Z	Output/Hi-Z	Hi-Z	Hi-Z
Output pin mode	CMOS/ open-drain	CMOS/ open-drain	CMOS/ open-drain	CMOS/ open-drain	CMOS/ open-drain
SSL polarity modification function	Supported	Supported	Supported	—	—
Transfer rate	Up to Rϕ/8	Up to Rϕ/2	Up to Rϕ/2	Up to Rϕ/8	Up to Rϕ/2
Clock source	RSPCK input	On-chip baud rate generator	On-chip baud rate generator	RSPCK input	On-chip baud rate generator
Clock polarity	Two	Two	Two	Two	Two

Mode	Slave (SPI Operation)	Single-Master (SPI Operation)	Multi-Master (SPI Operation)	Slave (Clock Synchronous Operation)	Master (Clock Synchronous Operation)
Clock phase	Two	Two	Two	One (CPHA = 1)	Two
First transfer bit	MSB/LSB	MSB/LSB	MSB/LSB	MSB/LSB	MSB/LSB
Transfer data length	8 to 32 bits	8 to 32 bits	8 to 32 bits	8 to 32 bits	8 to 32 bits
Burst transfer	Possible (CPHA = 1)	Possible (CPHA = 0,1)	Possible (CPHA = 0,1)	—	—
RSPCK delay control	Not supported	Supported	Supported	Not supported	Supported
SSL negation delay control	Not supported	Supported	Supported	Not supported	Supported
Next-access delay control	Not supported	Supported	Supported	Not supported	Supported
Transfer activation method	SSL input active or RSPCK oscillation	Transmit buffer is written when SPTEF = 1	Transmit buffer is written when SPTEF = 1	RSPCK oscillation	Transmit buffer is written when SPTEF = 1
Sequence control	Not supported	Supported	Supported	Not supported	Supported
Transmit buffer empty detection	Supported	Supported	Supported	Supported	Supported
Receive buffer full detection*1	Supported	Supported	Supported	Supported	Supported
Overrun error detection*1	Supported	Supported	Supported	Supported	Supported
Parity error detection*1*2	Supported	Supported	Supported	Supported	Supported
Mode fault error detection	Supported (MODFEN = 1)	Not supported	Supported	Not supported	Not supported

Notes: 1. When the TXMD bit in SPCR is 1, receive buffer full, overrun errors, and parity errors are not detected.
2. When the SPPE bit in SPCR2 is 0, parity errors are not detected.

15.4.2 Controlling RSPI Pins

According to the MSTR, MODFEN, and SPMS bits in the RSPI control register (SPCR) and the SPOM bit in the RSPI pin control register (SPPCR), the RSPI can automatically switch pin directions and output modes. Table 15.6 shows the relationship between pin states and bit settings.

Table 15.6 Relationship between Pin States and Bit Settings

		Pin State*1	
Mode	**Pin**	**SPOM = 0**	**SPOM = 1**
Single-master mode (SPI operation) (MSTR = 1, MODFEN = 0, SPMS = 0)	RSPCK	CMOS output	Open-drain output
	SSL0	CMOS output	Open-drain output
	SSL1 to SSL3	CMOS output/Hi-Z	Open-drain output/Hi-Z
	MOSI	CMOS output	Open-drain output
	MISO	Input	Input
Multi-master mode (SPI operation) (MSTR = 1, MODFEN = 1, SPMS = 0)	RSPCK*2	CMOS output/Hi-Z	Open-drain output/Hi-Z
	SSL0	Input	Input
	SSL1 to SSL3*2	CMOS output/Hi-Z	Open-drain output/Hi-Z
	MOSI*2	CMOS output/Hi-Z	Open-drain output/Hi-Z
	MISO	Input	Input
Slave mode (SPI operation) (MSTR = 0, SPMS = 0)	RSPCK	Input	Input
	SSL0	Input	Input
	SSL1 to SSL3	Hi-Z	Hi-Z
	MOSI	Input	Input
	MISO*3	CMOS output/Hi-Z	Open-drain output/Hi-Z
Master mode (Clock synchronous operation) (MSTR = 1, MODFEN = 0, SPMS = 1)	RSPCK	CMOS output	Open-drain output
	SSL0 to SSL3*4	Hi-Z	Hi-Z
	MOSI	CMOS output	Open-drain output
	MISO	Input	Input
Slave mode (Clock synchronous operation) (MSTR = 0, SPMS = 1)	RSPCK	Input	Input
	SSL0 to SSL3*4	Hi-Z	Hi-Z
	MOSI	Input	Input
	MISO	CMOS output	Open-drain output

Notes: 1. RSPI settings are not indicated in the multiplex pins for which the RSPI function is not selected.
2. When SSL0 is at the active level, the pin state is Hi-Z.
3. When SSL0 is at the non-active level or the SPE bit in SPCR is cleared (= 0), the pin state is Hi-Z.
4. In clock synchronous operation, SSL0 to SSL3 are available for use as I/O port pins.

The RSPI in single-master mode (SPI operation) or multi-master mode (SPI operation) determines MOSI signal values during the SSL negation period (including the SSL retention period during a burst transfer) according to MOIFE and MOIFV bit settings in SPPCR, as shown in table 15.7.

Table 15.7 MOSI Signal Value Determination during SSL Negation Period

MOIFE	MOIFV	MOSI Signal Value during SSL Negation Period
0	0, 1	Final data from previous transfer
1	0	Always 0
1	1	Always 1

15.4.3 RSPI System Configuration Example

(1) Single Master/Single Slave (with This LSI Acting as Master)

Figure 15.3 shows a single-master/single-slave RSPI system configuration example when this LSI is used as a master. In the single-master/single-slave configuration, the SSL0 to SSL3 output of this LSI (master) are not used. The SSL input of the RSPI slave is fixed to the low level, and the RSPI slave is always maintained in a select state. In the transfer format corresponding to the case where the CPHA bit in the RSPI control register (SPCR) is 0, there are slave devices for which the SSL signal cannot be fixed to the active level. In situations where the SSL signal cannot be fixed, the SSL output of this LSI should be connected to the SSL input of the slave device.

This LSI (master) always drives the RSPCK and MOSI. The RSPI slave always drives the MISO.

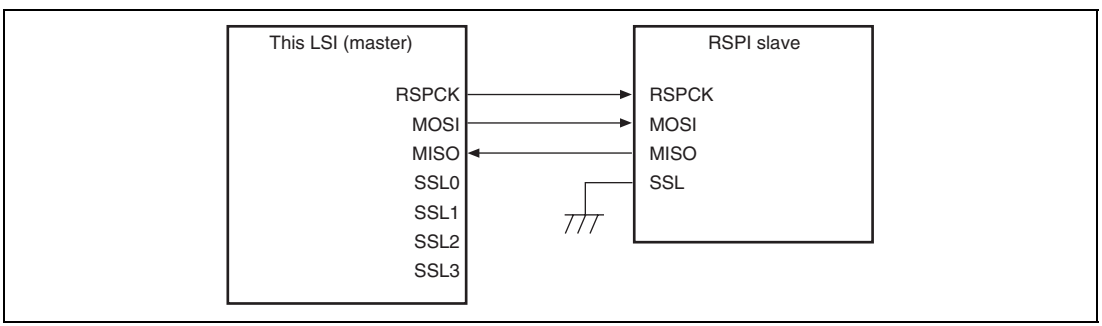


Figure 15.3 Single-Master/Single-Slave Configuration Example (This LSI = Master)

(2) Single Master/Single Slave (with This LSI Acting as Slave)

Figure 15.4 shows a single-master/single-slave RSPI system configuration example when this LSI is used as a slave. When this LSI is to operate as a slave, the SSL0 pin is used as SSL input. The RSPI master always drives the RSPCK and MOSI. This LSI (slave) always drives the MISO. When SSL0 is at the active level, the pin state is Hi-Z.

In the single-slave configuration in which the CPHA bit in the RSPI command register (SPCMD) is set to 1, the SSL0 input of this LSI (slave) is fixed to the 0 level, this LSI (slave) is always maintained in a selected state, and in this manner it is possible to execute serial transfer (figure 15.5).

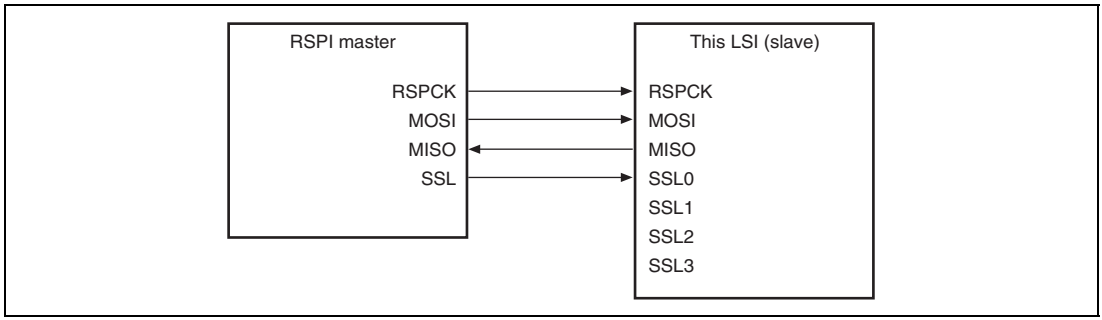


Figure 15.4 Single-Master/Single-Slave Configuration Example (This LSI = Slave)

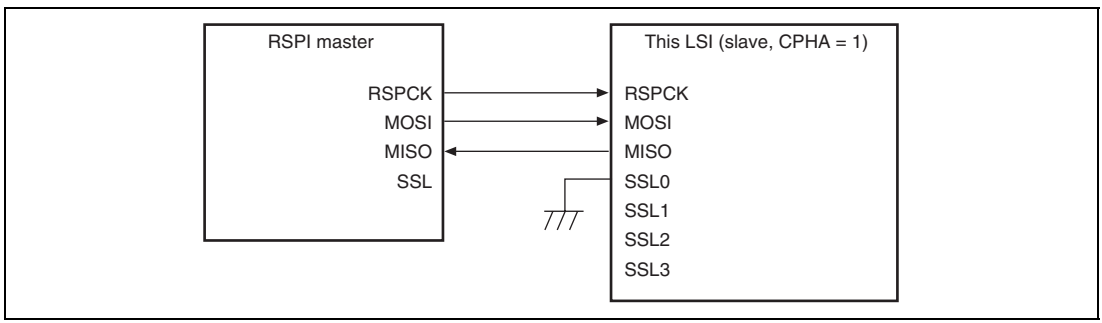


Figure 15.5 Single-Master/Single-Slave Configuration Example (This LSI = Slave, CPHA = 1)

(3) Single Master/Multi-Slave (with This LSI Acting as Master)

Figure 15.6 shows a single-master/multi-slave RSPI system configuration example when this LSI is used as a master. In the example of figure 15.6, the RSPI system is comprised of this LSI (master) and four slaves (RSPI slave 0 to RSPI slave 3).

The RSPCK and MOSI outputs of this LSI (master) are connected to the RSPCK and MOSI inputs of RSPI slave 0 to RSPI slave 3. The MISO outputs of RSPI slave 0 to RSPI slave 3 are all connected to the MISO input of this LSI (master). SSL0 to SSL3 outputs of this LSI (master) are connected to the SSL inputs of RSPI slave 0 to RSPI slave 3, respectively.

This LSI (master) always drives RSPCK, MOSI, and SSL0 to SSL3. Of the RSPI slave 0 to RSPI slave 3, the slave that receives low-level input into the SSL input drives MISO.

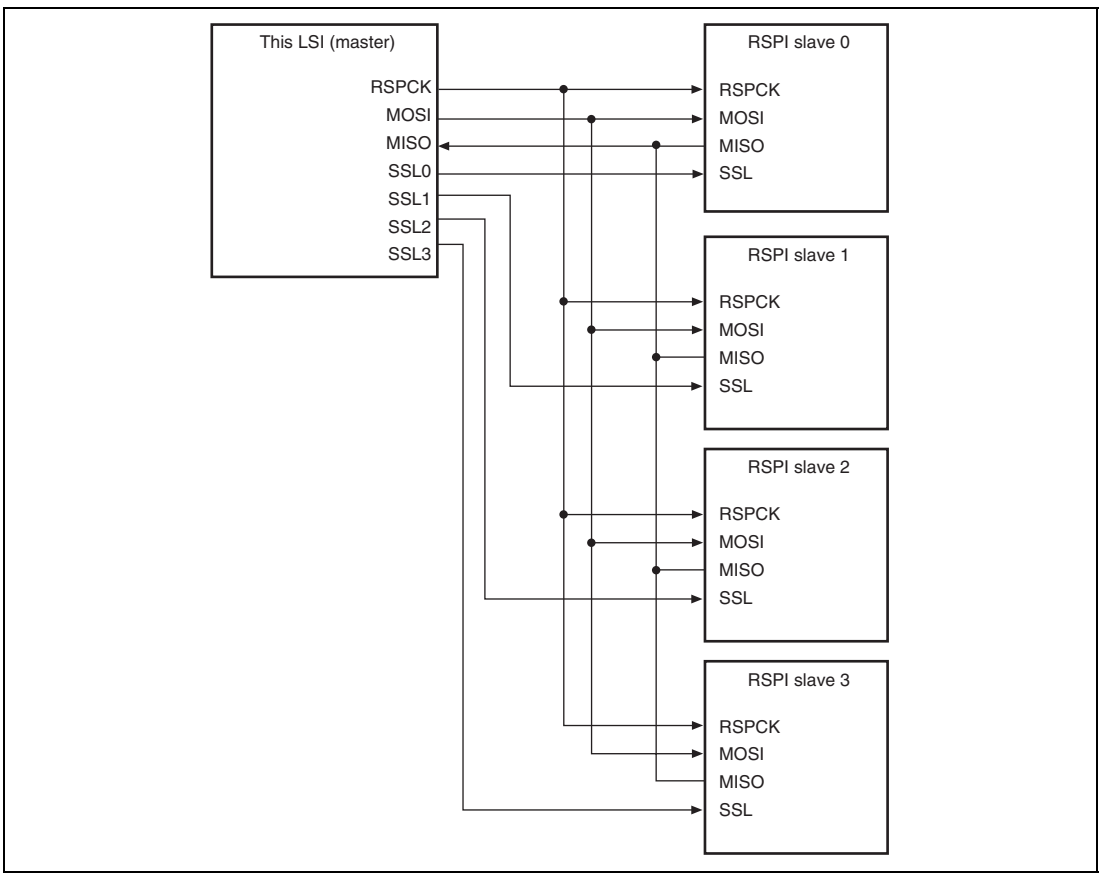


Figure 15.6 Single-Master/Multi-Slave Configuration Example (This LSI = Master)

(4) Single Master/Multi-Slave (with This LSI Acting as Slave)

Figure 15.7 shows a single-master/multi-slave RSPI system configuration example when this LSI is used as a slave. In the example of figure 15.7, the RSPI system is comprised of an RSPI master and two LSIs (slave X and slave Y).

The RSPCK and MOSI outputs of the RSPI master are connected to the RSPCK and MOSI inputs of the LSIs (slave X and slave Y). The MISO outputs of the LSIs (slave X and slave Y) are all connected to the MISO input of the RSPI master. SSLX and SSLY outputs of the RSPI master are connected to the SSL0 inputs of the LSIs (slave X and slave Y), respectively.

The RSPI master always drives RSPCK, MOSI, SSLX, and SSLY. Of the LSIs (slave X and slave Y), the slave that receives low-level input into the SSL0 input drives MISO.

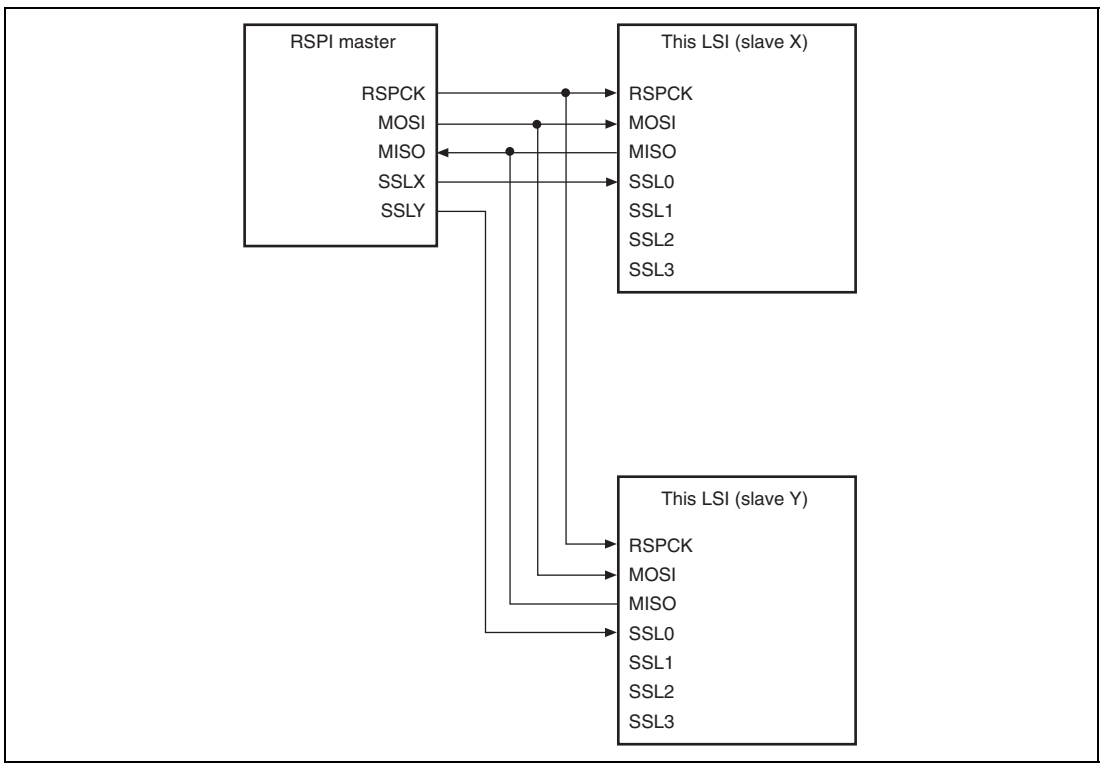


Figure 15.7 Single-Master/Multi-Slave Configuration Example (This LSI = Slave)

(5) Multi-Master/Multi-Slave (with This LSI Acting as Master)

Figure 15.8 shows a multi-master/multi-slave RSPI system configuration example when this LSI is used as a master. In the example of figure 15.8, the RSPI system is comprised of two LSIs (master X, master Y) and two RSPI slaves (RSPI slave 1, RSPI slave 2).

The RSPCK and MOSI outputs of this LSI (master X, master Y) are connected to the RSPCK and MOSI inputs of RSPI slaves 1 and 2. The MISO outputs of RSPI slaves 1 and 2 are connected to the MISO inputs of this LSI (master X, master Y). Any generic port Y output from this LSI (master X) is connected to the SSL0 input of this LSI (master Y). Any generic port X output of this LSI (master Y) is connected to the SSL0 input of this LSI (master X). The SSL1 and SSL2 outputs of this LSI (master X, master Y) are connected to the SSL inputs of the RSPI slaves 1 and 2. In this configuration example, because the system can be comprised solely of SSL0 input, and SSL1 and SSL2 outputs for slave connections, SSL3 output of this LSI is not required.

This LSI drives RSPCK, MOSI, SSL1, and SSL2 when the SSL0 input level is 1. When the SSL0 input level is 0, this LSI detects a mode fault error, sets RSPCK, MOSI, SSL1, and SSL2 to Hi-Z, and releases the RSPI bus right to the other master. Of the RSPI slaves 1 and 2, the slave that receives a level-0 input into the SSL input drives MISO.

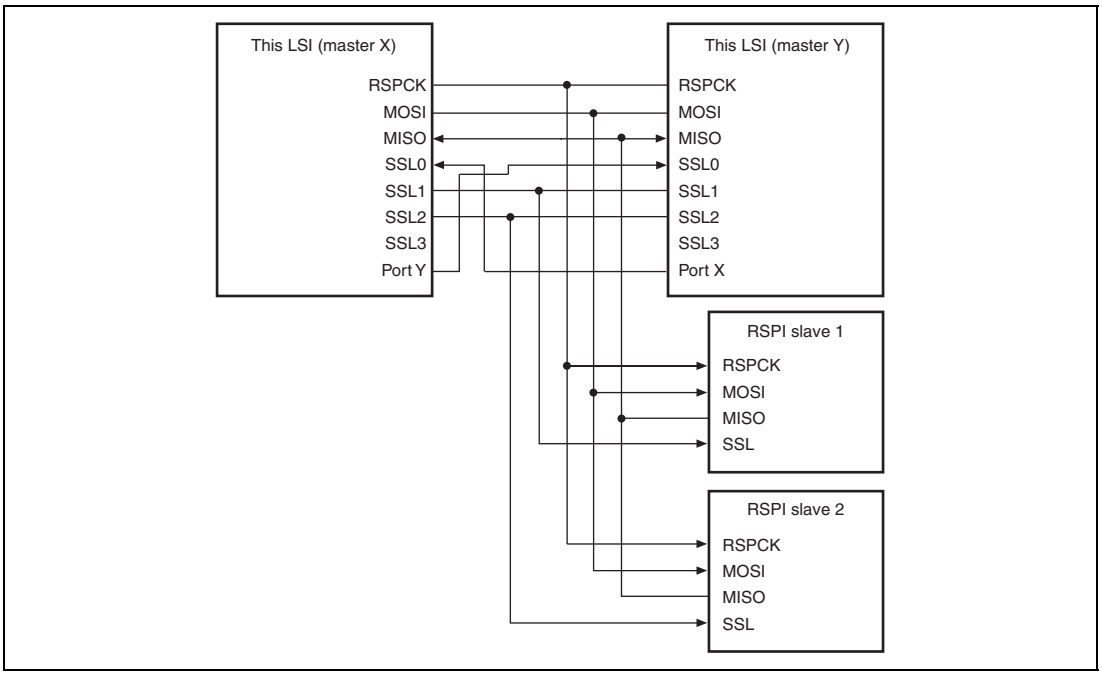


Figure 15.8 Multi-Master/Multi-Slave Configuration Example (This LSI = Master)

(6) Master (Clock Synchronous Operation)/Slave (Clock Synchronous Operation) (with This LSI Acting as Master)

Figure 15.9 shows a master (clock synchronous operation)/slave (clock synchronous operation) RSPI system configuration example when this LSI is used as a master. In the master (clock synchronous operation)/slave (clock synchronous operation) configuration, SSL0 to SSL3 of this LSI (master) are not used.

This LSI (master) always drives the RSPCK and MOSI. The RSPI slave always drives the MISO.

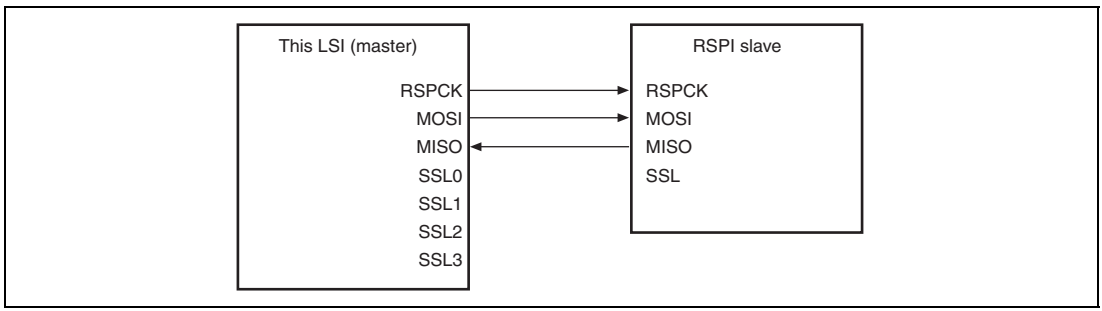


Figure 15.9 Master (Clock Synchronous Operation)/Slave (Clock Synchronous Operation) Configuration Example (This LSI = Master)

(7) Master (Clock Synchronous Operation)/Slave (Clock Synchronous Operation) (with This LSI Acting as Master)

Figure 15.10 shows a master (clock synchronous operation)/slave (clock synchronous operation) RSPI system configuration example when this LSI is used as a master. When this LSI is to operate as a slave (clock synchronous operation), this LSI (slave) always drives the MISO and the RSPI master always drives the RSPCK and MOSI. In addition, SSL0 to SSL3 of this LSI (slave) are not used.

Only in the single-slave configuration in which the CPHA bit in the RSPI command register (SPCMD) is set to 1, this LSI (slave) can execute serial transfer.

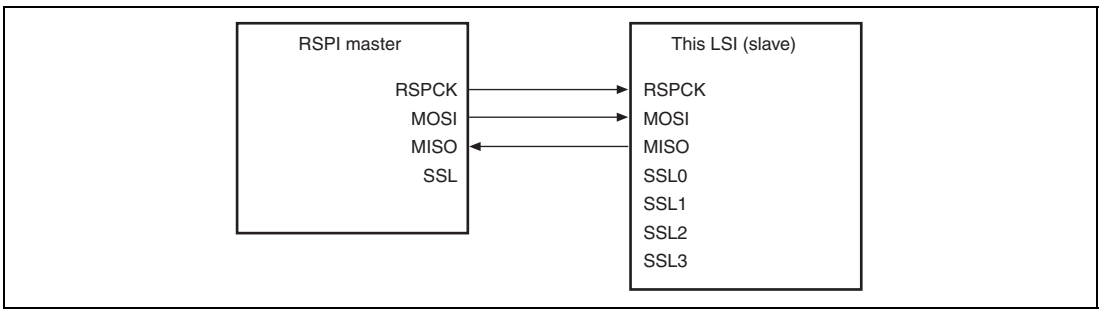


Figure 15.10 Master (Clock Synchronous Operation)/Slave (Clock Synchronous Operation) Configuration Example (This LSI = Slave, CPHA = 1)

15.4.4 Transfer Format

(1) CPHA = 0

Figure 15.11 shows a sample transfer format for the serial transfer of 8-bit data when the CPHA bit in the RSPI command register (SPCMD) is 0. Note that clock synchronous operation (the SPMS bit in the RSPI control register (SPCR) is 1) is not guaranteed when the RSPI operates in slave mode (MSTR = 0) and the CPHA bit is 0. In figure 15.11, RSPCK (CPOL = 0) indicates the RSPCK signal waveform when the CPOL bit in SPCMD is 0; RSPCK (CPOL = 1) indicates the RSPCK signal waveform when the CPOL bit is 1. The sampling timing represents the timing at which the RSPI fetches serial transfer data into the shift register. The input/output directions of the signals depend on the RSPI settings. For details, see section 15.4.2, Controlling RSPI Pins.

When the CPHA bit is 0, the driving of valid data to the MOSI and MISO signals commences at an SSL signal assertion timing. The first RSPCK signal change timing that occurs after the SSL signal assertion becomes the first transfer data fetch timing. After this timing, data is sampled at every 1 RSPCK cycle. The change timing for MOSI and MISO signals is always 1/2 RSPCK cycle after the transfer data fetch timing. The settings in the CPOL bit do not affect the RSPCK signal operation timing; they only affect the signal polarity.

t1 denotes a period from an SSL signal assertion to RSPCK oscillation (RSPCK delay). t2 denotes a period from the cessation of RSPCK oscillation to an SSL signal negation (SSL negation delay). t3 denotes a period in which SSL signal assertion is suppressed for the next transfer after the end of serial transfer (next-access delay). t1, t2, and t3 are controlled by a master device running on the RSPI system. For a description of t1, t2, and t3 when the RSPI of this LSI is in master mode, see section 15.4.10 (1), Master Mode Operation.

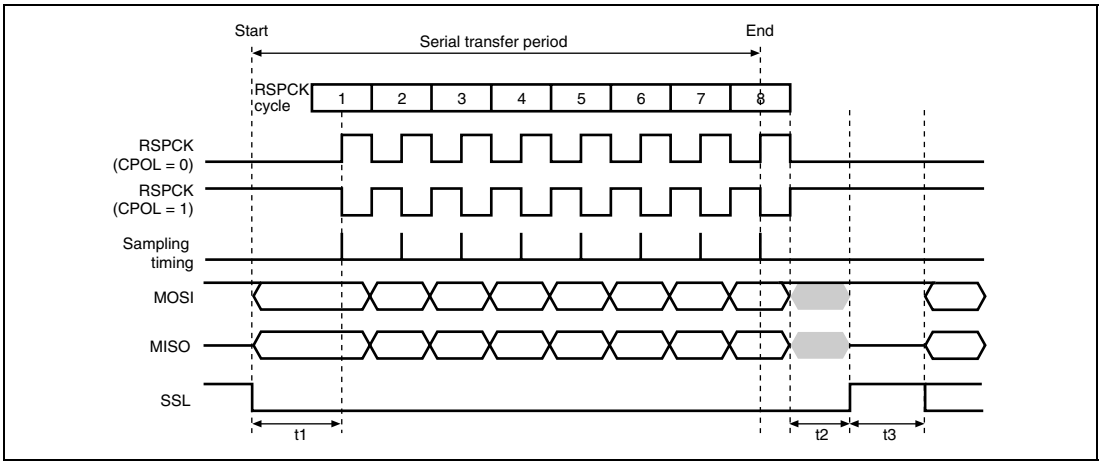


Figure 15.11 RSPI Transfer Format (CPHA = 0)

(2) CPHA = 1

Figure 15.12 shows a sample transfer format for the serial transfer of 8-bit data when the CPHA bit in the RSPI command register (SPCMD) is 1. However, when the SPMS bit in the RSPI control register (SPCR) is 1, the SSL signals are not used, and only the three signals RSPCK, MOSI, and MISO handle communications. In figure 15.12, RSPCK (CPOL = 0) indicates the RSPCK signal waveform when the CPOL bit in SPCMD is 0; RSPCK (CPOL = 1) indicates the RSPCK signal waveform when the CPOL bit is 1. The sampling timing represents the timing at which the RSPI fetches serial transfer data into the shift register. The input/output directions of the signals depend on the RSPI modes (master or slave). For details, see section 15.4.2, Controlling RSPI Pins.

When the CPHA bit is 1, the driving of invalid data to the MOSI and MISO signals commences at an SSL signal assertion timing. The driving of valid data to the MOSI and MISO signals commences at the first RSPCK signal change timing that occurs after the SSL signal assertion. After this timing, data is updated at every 1 RSPCK cycle. The transfer data fetch timing is always 1/2 RSPCK cycle after the data update timing. The settings in the CPOL bit do not affect the RSPCK signal operation timing; they only affect the signal polarity.

t1, t2, and t3 are the same as those in the case of CPHA = 0. For a description of t1, t2, and t3 when the RSPI of this LSI is in master mode, see section 15.4.10 (1), Master Mode Operation.

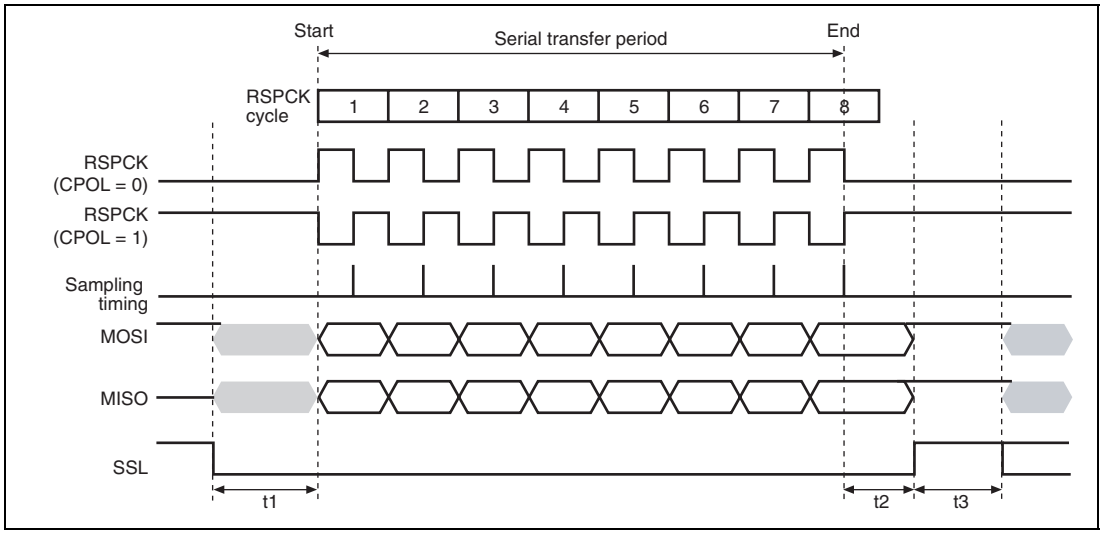


Figure 15.12 RSPI Transfer Format (CPHA = 1)

15.4.5 Data Format

The RSPI's data format depends on the settings in the RSPI command register (SPCMD) and the parity enable bit (SPPE) in RSPI control register 2 (SPCR2). Irrespective of MSB/LSB first, the RSPI treats the range from the LSB of the RSPI data register (SPDR) to the assigned data length as transfer data.

(1) MSB First Transfer (32-Bit Data)

(a) When Parity Function is Disabled (SPPE = 0)

Figure 15.13 shows the operation of the RSPI data register (SPDR) and the shift register when the RSPI performs a 32-bit data length MSB-first data transfer with the parity function disabled.

T31 to T00 is written to the transmit buffer of SPDR by the CPU or DMAC. If the SPTEF bit in the RSPI status register (SPSR) is 0 and the shift register is empty, the RSPI copies the data in the transmit buffer of SPDR to the shift register, and fully populates the shift register. When serial transfer starts, the RSPI outputs data from the MSB (bit 31) of the shift register, and shifts in the data from the LSB (bit 0) of the shift register. When the RSPCK cycle required for the serial transfer of 32 bits has passed, data R31 to R00 is stored in the shift register. In this state, in the case of full-duplex synchronous serial communications (TXMD = 0), the RSPI copies the data from the shift register to the receive buffer of SPDR, and empties the shift register.

If another serial transfer is started before the CPU or the DMAC writes to the transmit buffer of SPDR, received data R31 to R00 is shifted out from the shift register.

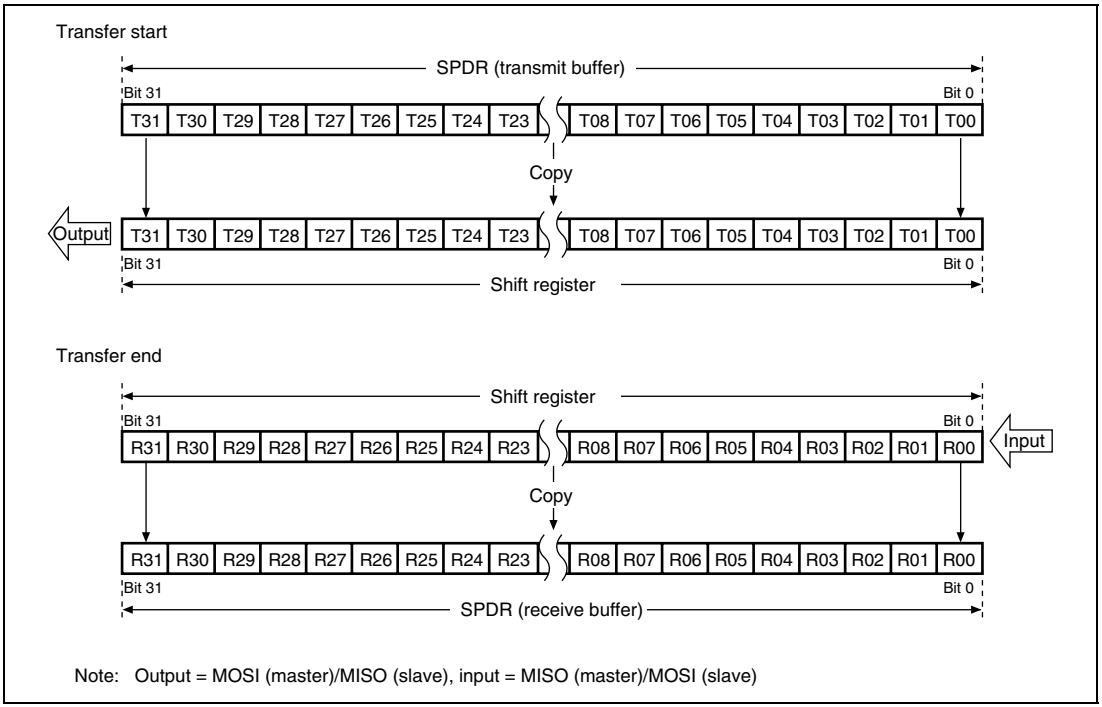


Figure 15.13 MSB First Transfer (32-Bit Data, Parity Function Disabled)

(b) When Parity Function is Enabled (SPPE = 1)

Figure 15.14 shows the operation of the RSPI data register (SPDR) and the shift register when the RSPI performs a 32-bit data length MSB-first data transfer with the parity function enabled.

T31 to T00 is written to the transmit buffer of SPDR by the CPU or DMAC. If the SPTEF bit in the RSPI status register (SPSR) is 0 and the shift register is empty, the RSPI converts T00 of the data stored in the transmit buffer of SPDR into the parity bit (P). Then, the RSPI copies the data with the parity bit (P) added to the shift register, and fully populates the shift register. When serial transfer starts, the RSPI outputs data from the MSB (bit 31) of the shift register, and shifts in the data from the LSB (bit 0) of the shift register. When the RSPCK cycle required for the serial transfer of 32 bits has passed, data R31 to P is stored in the shift register. In this state, in the case of full-duplex synchronous serial communications (TXMD = 0), the RSPI copies the data from the shift register to the receive buffer of SPDR, and empties the shift register.

If another serial transfer is started before data is written to the transmit buffer of SPDR by the CPU or DMAC, received data R31 to P is shifted out from the shift register.

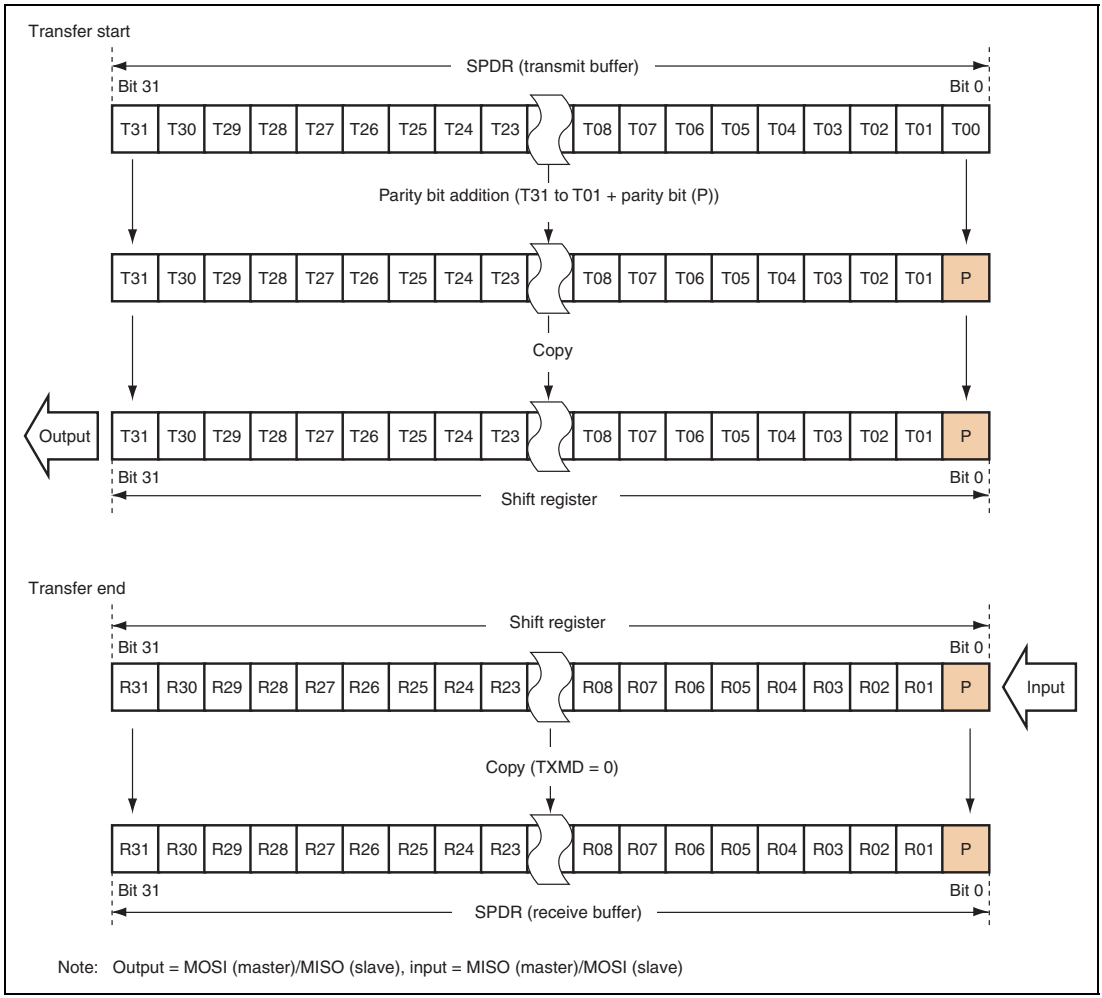


Figure 15.14 MSB First Transfer (32-Bit Data, Parity Function Enabled)

(2) MSB First Transfer (24-Bit Data)

(a) When Parity Function is Disabled (SPPE = 0)

Figure 15.15 shows the operation of the RSPI data register (SPDR) and the shift register when the RSPI performs a 24-bit data length MSB-first data transfer (as an example of operation in the transfer of data with lengths other than 32 bits) with the parity function disabled.

The CPU or the DMAC writes T31 to T00 to the transmit buffer of SPDR. If the SPTEF bit in the RSPI status register (SPSR) is 0 and the shift register is empty, the RSPI copies the data in the transmit buffer of SPDR to the shift register, and fully populates the shift register. When serial transfer starts, the RSPI outputs data from bit 23 of the shift register, and shifts in the data from the LSB (bit 0) of the shift register. When the RSPCK cycle required for the serial transfer of 24 bits has passed, received data R23 to R00 is stored in bits 23 to 0 of the shift register. After completion of the serial transfer, data that existed before the transfer is retained in bits 31 to 24 in the shift register. In this state, in the case of full-duplex synchronous serial communications (TXMD = 0), the RSPI copies the data from the shift register to the receive buffer of SPDR, and empties the shift register.

If another serial transfer is started before the CPU or the DMAC writes to the transmit buffer of SPDR, received data R23 to R00 is shifted out from the shift register.

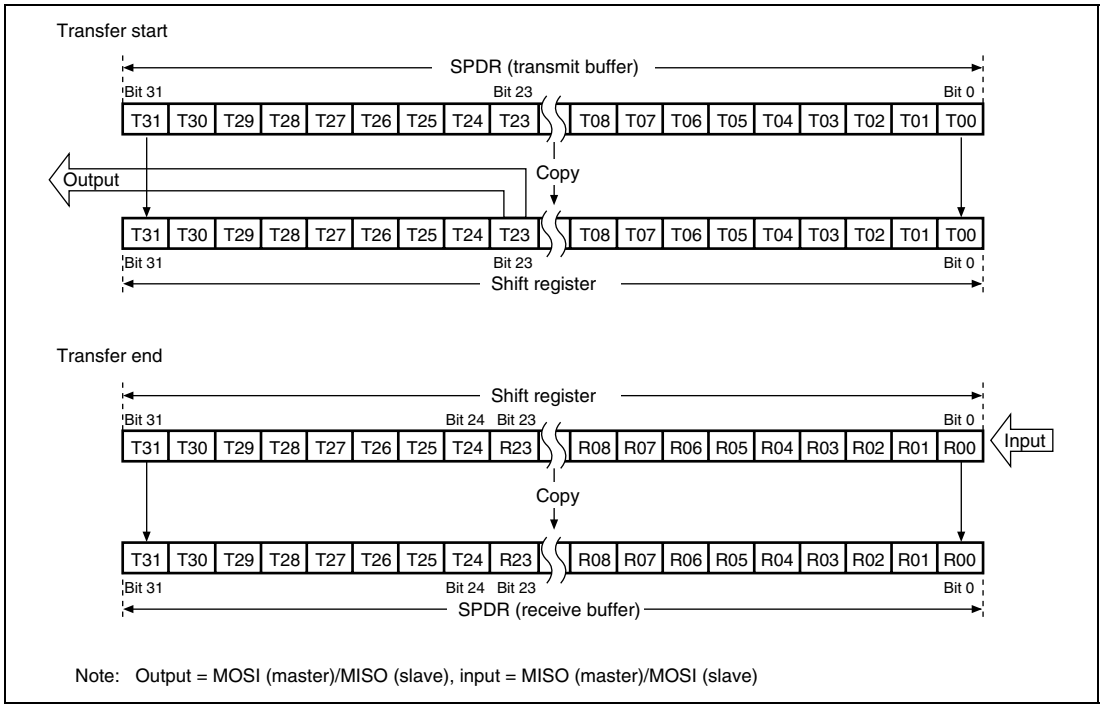


Figure 15.15 MSB First Transfer (24-Bit Data, Parity Function Disabled)

(b) When Parity Function is Enabled (SPPE = 1)

Figure 15.16 shows the operation of the RSPI data register (SPDR) and the shift register when the RSPI performs a 24-bit data length MSB-first data transfer (as an example of operation in the transfer of data with lengths other than 32 bits) with the parity function enabled.

The CPU or the DMAC writes T31 to T00 to the transmit buffer of SPDR. If the SPTEF bit in the RSPI status register (SPSR) is 0 and the shift register is empty, the RSPI converts T00 of the data stored in the transmit buffer of SPDR into the parity bit (P). Then, the RSPI copies the data with the parity bit (P) added to the shift register, and fully populates the shift register. When serial transfer starts, the RSPI outputs data from bit 23 of the shift register, and shifts in the data from the LSB (bit 0) of the shift register. When the RSPCK cycle required for the serial transfer of 24 bits has passed, received data R23 to P is stored in bits 23 to 0 of the shift register. After completion of the serial transfer, data that existed before the transfer is retained in bits 24 to 31 in the shift register. In this state, in the case of full-duplex synchronous serial communications (TXMD = 0), the RSPI copies the data from the shift register to the receive buffer of SPDR, and empties the shift register.

If another serial transfer is started before the CPU or the DMAC writes to the transmit buffer of SPDR, received data R23 to P is shifted out from the shift register.

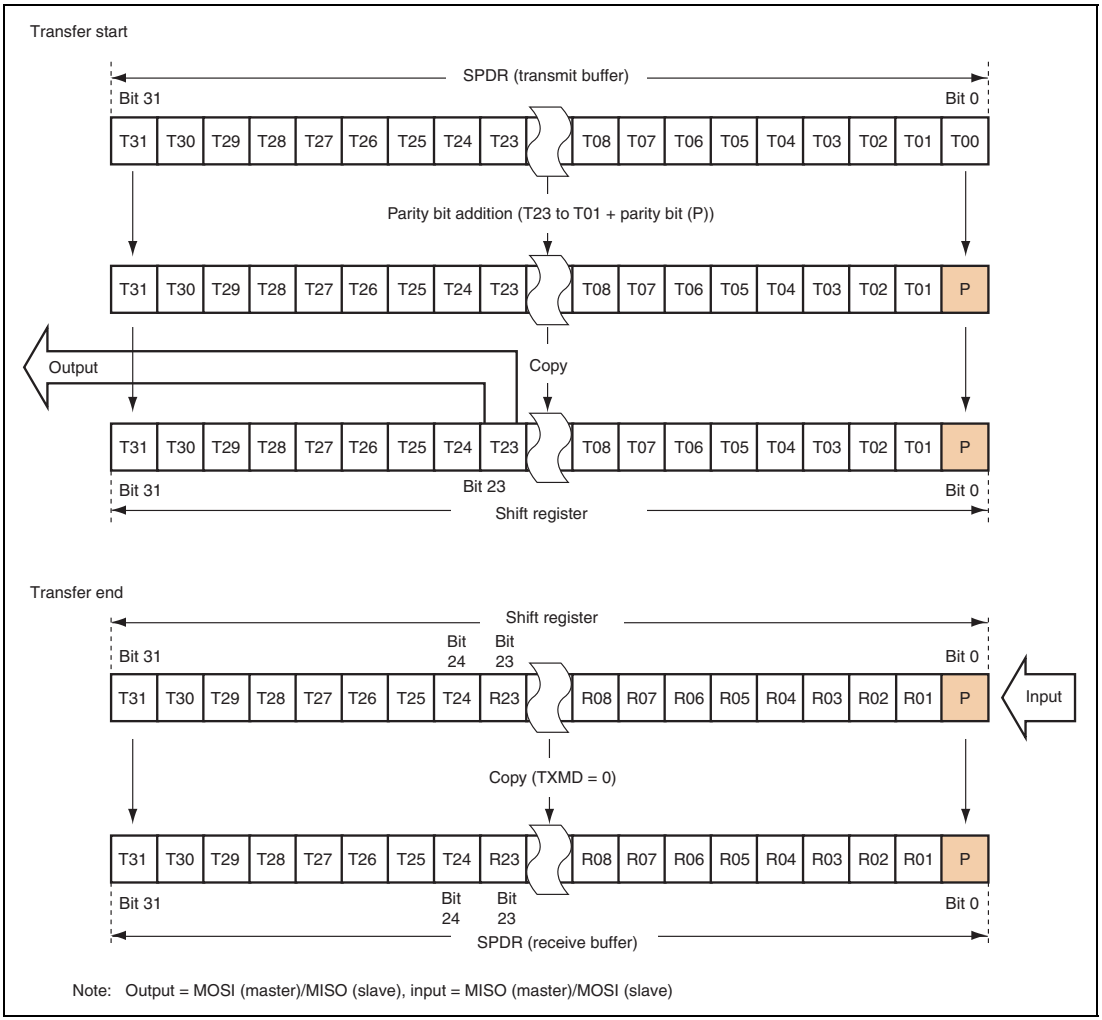


Figure 15.16 MSB First Transfer (24-Bit Data, Parity Function Enabled)

(3) LSB First Transfer (32-Bit Data)

(a) When Parity Function is Disabled (SPPE = 0)

Figure 15.17 shows the operation of the RSPI data register (SPDR) and the shift register when the RSPI performs a 32-bit data length LSB-first data transfer with the parity function disabled.

The CPU or the DMAC writes T31 to T00 to the transmit buffer of SPDR. If the SPTEF bit in the RSPI status register (SPSR) is 0 and the shift register is empty, the RSPI reverses the order of the bits of the data in the transmit buffer of SPDR, copies it to the shift register, and fully populates the shift register. When serial transfer starts, the RSPI outputs data from the MSB (bit 31) of the shift register, and shifts in the data from the LSB (bit 0) of the shift register. When the RSPCK cycle required for the serial transfer of 32 bits has passed, data R00 to R31 is stored in the shift register. In this state, in the case of full-duplex synchronous serial communications (TXMD = 0), the RSPI copies the data, in which the order of the bits is reversed, from the shift register to the receive buffer of SPDR, and empties the shift register.

If another serial transfer is started before the CPU or the DMAC writes to the transmit buffer of SPDR, received data R00 to R31 is shifted out from the shift register.

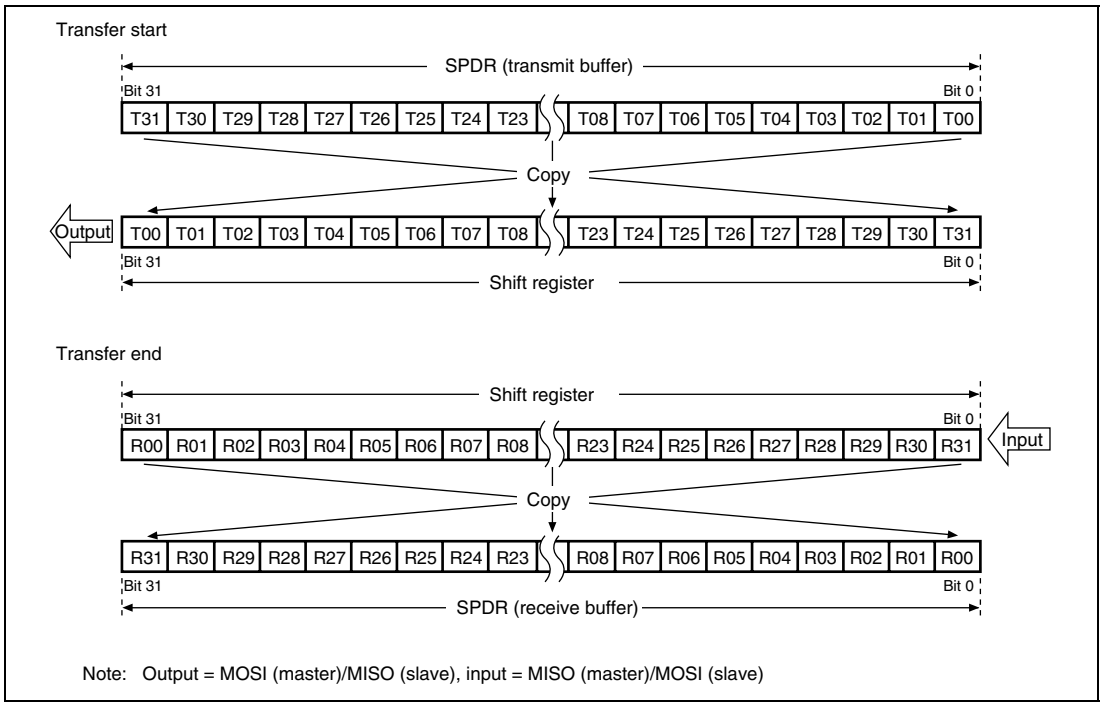


Figure 15.17 LSB First Transfer (32-Bit Data, Parity Function Disabled)

(b) When Parity Function is Enabled (SPPE = 1)

Figure 15.18 shows the operation of the RSPI data register (SPDR) and the shift register when the RSPI performs a 32-bit data length LSB-first data transfer with the parity function enabled.

The CPU or the DMAC writes T31 to T00 to the transmit buffer of SPDR. The RSPI converts T31 of the data stored in the transmit buffer of SPDR into the parity bit (P). If the SPTEF bit in the RSPI status register (SPSR) is 0 and the shift register is empty, the RSPI reverses the order of the bits of the data with the parity bit (P) added, copies it to the shift register, and fully populates the shift register. When serial transfer starts, the RSPI outputs data from the MSB (bit 31) of the shift register, and shifts in the data from the LSB (bit 0) of the shift register. When the RSPCK cycle required for the serial transfer of 32 bits has passed, data R00 to P is stored in the shift register. In this state, in the case of full-duplex synchronous serial communications (TXMD = 0), the RSPI copies the data, in which the order of the bits is reversed, from the shift register to the receive buffer of SPDR, and empties the shift register.

If another serial transfer is started before the CPU or the DMAC writes to the transmit buffer of SPDR, received data R00 to P is shifted out from the shift register.

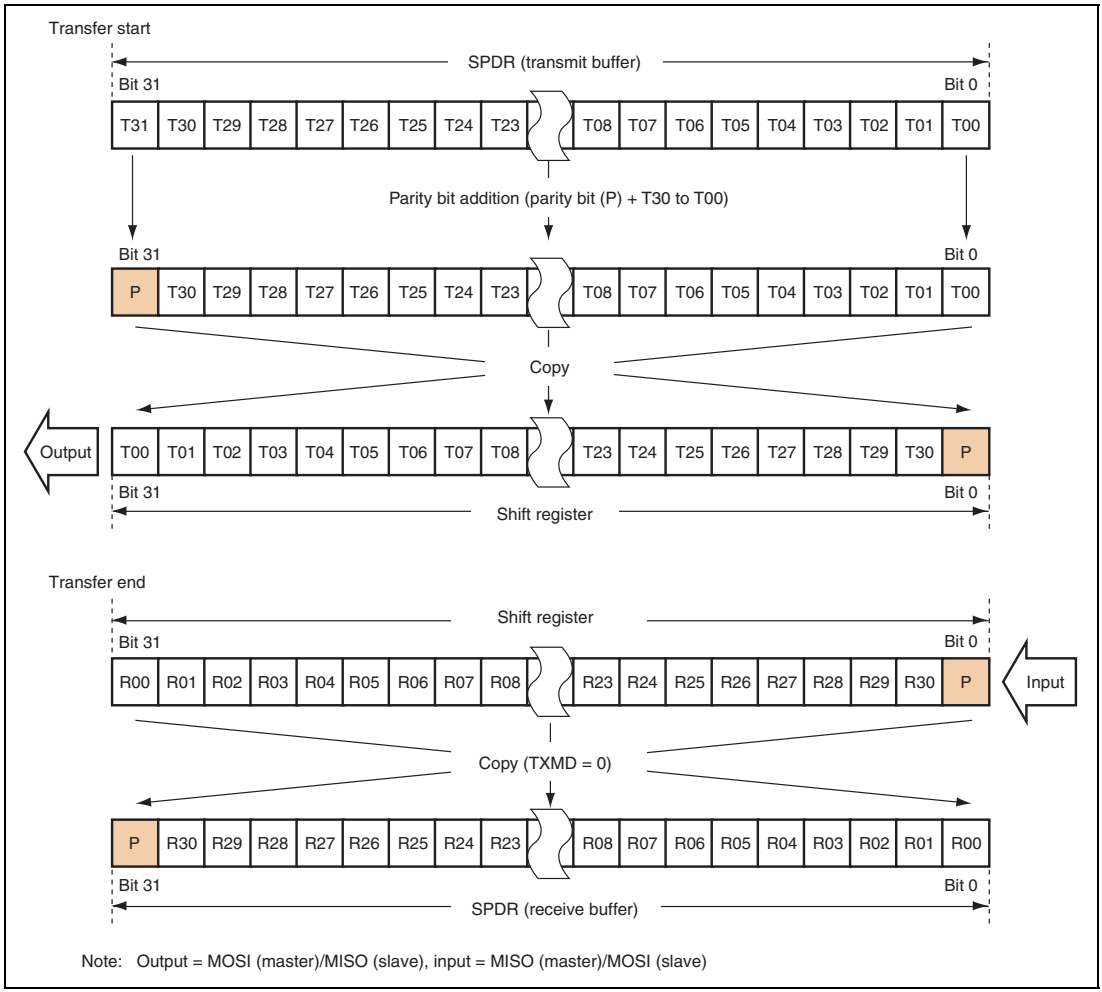


Figure 15.18 LSB First Transfer (32-Bit Data, Parity Function Enabled)

(4) LSB First Transfer (24-Bit Data)

(a) When Parity Function is Disabled (SPPE = 0)

Figure 15.19 shows the operation of the RSPI data register (SPDR) and the shift register when the RSPI performs a 24-bit data length LSB-first data transfer (as an example of operation in the transfer of data with lengths other than 32 bits) with the parity function disabled.

The CPU or the DMAC writes T31 to T00 to the transmit buffer of SPDR. If the SPTEF bit in the RSPI status register (SPSR) is 0 and the shift register is empty, the RSPI reverses the order of the bits of the data in the transmit buffer of SPDR, copies it to the shift register, and fully populates the shift register. When serial transfer starts, the RSPI outputs data from the MSB (bit 31) of the shift register, and shifts in the data from bit 8 of the shift register. When the RSPCK cycle required for the serial transfer of 24 bits has passed, received data R00 to R23 is stored in bits 31 to 8 of the shift register. After completion of the serial transfer, data that existed before the transfer is retained in bits 7 to 0 of the shift register. In this state, in the case of full-duplex synchronous serial communications (TXMD = 0), the RSPI copies the data, in which the order of the bits is reversed, from the shift register to the receive buffer of SPDR, and empties the shift register.

If another serial transfer is started before the CPU or the DMAC writes to the transmit buffer of SPDR, received data R00 to R23 is shifted out from the shift register.

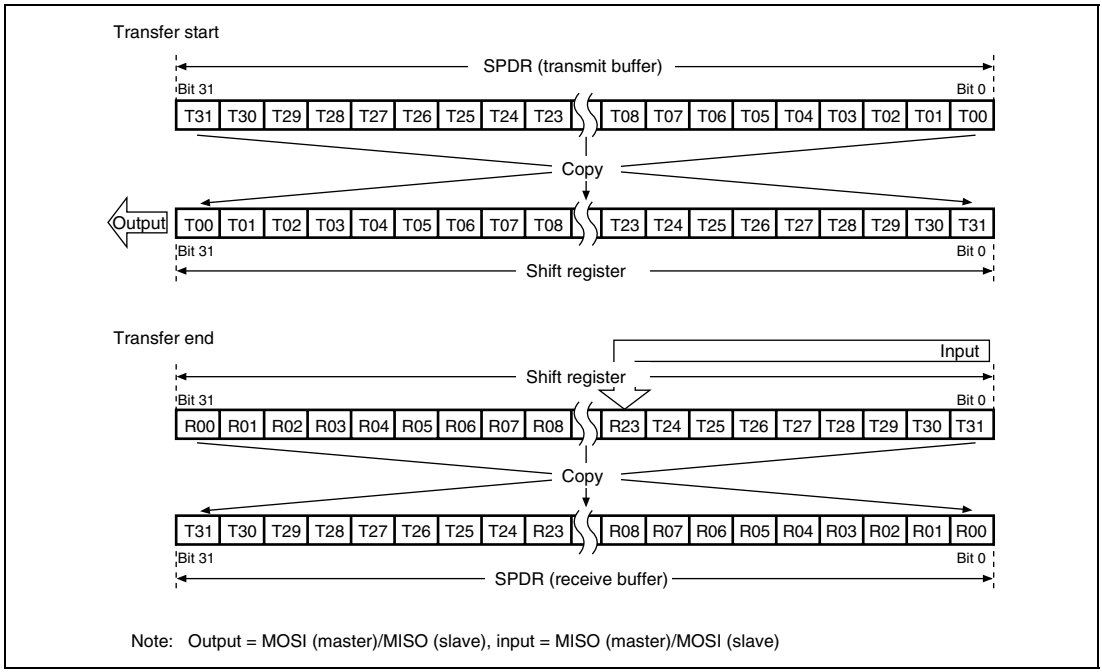


Figure 15.19 LSB First Transfer (24-Bit Data, Parity Function Disabled)

(b) When Parity Function is Enabled (SPPE = 1)

Figure 15.20 shows the operation of the RSPI data register (SPDR) and the shift register when the RSPI performs a 24-bit data length LSB-first data transfer (as an example of operation in the transfer of data with lengths other than 32 bits) with the parity function enabled.

The CPU or the DMAC writes T31 to T00 to the transmit buffer of SPDR. The RSPI converts T23 of the data stored in the transmit buffer of SPDR into the parity bit (P). If the SPTEF bit in the RSPI status register (SPSR) is 0 and the shift register is empty, the RSPI reverses the order of the bits of the data with the parity bit (P) added, copies it to the shift register, and fully populates the shift register. When serial transfer starts, the RSPI outputs data from the MSB (bit 31) of the shift register, and shifts in the data from bit 8 of the shift register. When the RSPCK cycle required for the serial transfer of 24 bits has passed, received data R00 to P is stored in bits 31 to 8 of the shift register. After completion of the serial transfer, data that existed before the transfer is retained in bits 7 to 0 of the shift register. In this state, in the case of full-duplex synchronous serial communications (TXMD = 0), the RSPI copies the data, in which the order of the bits is reversed, from the shift register to the receive buffer of SPDR, and empties the shift register.

If another serial transfer is started before the CPU or the DMAC writes to the transmit buffer of SPDR, received data R00 to P is shifted out from the shift register.

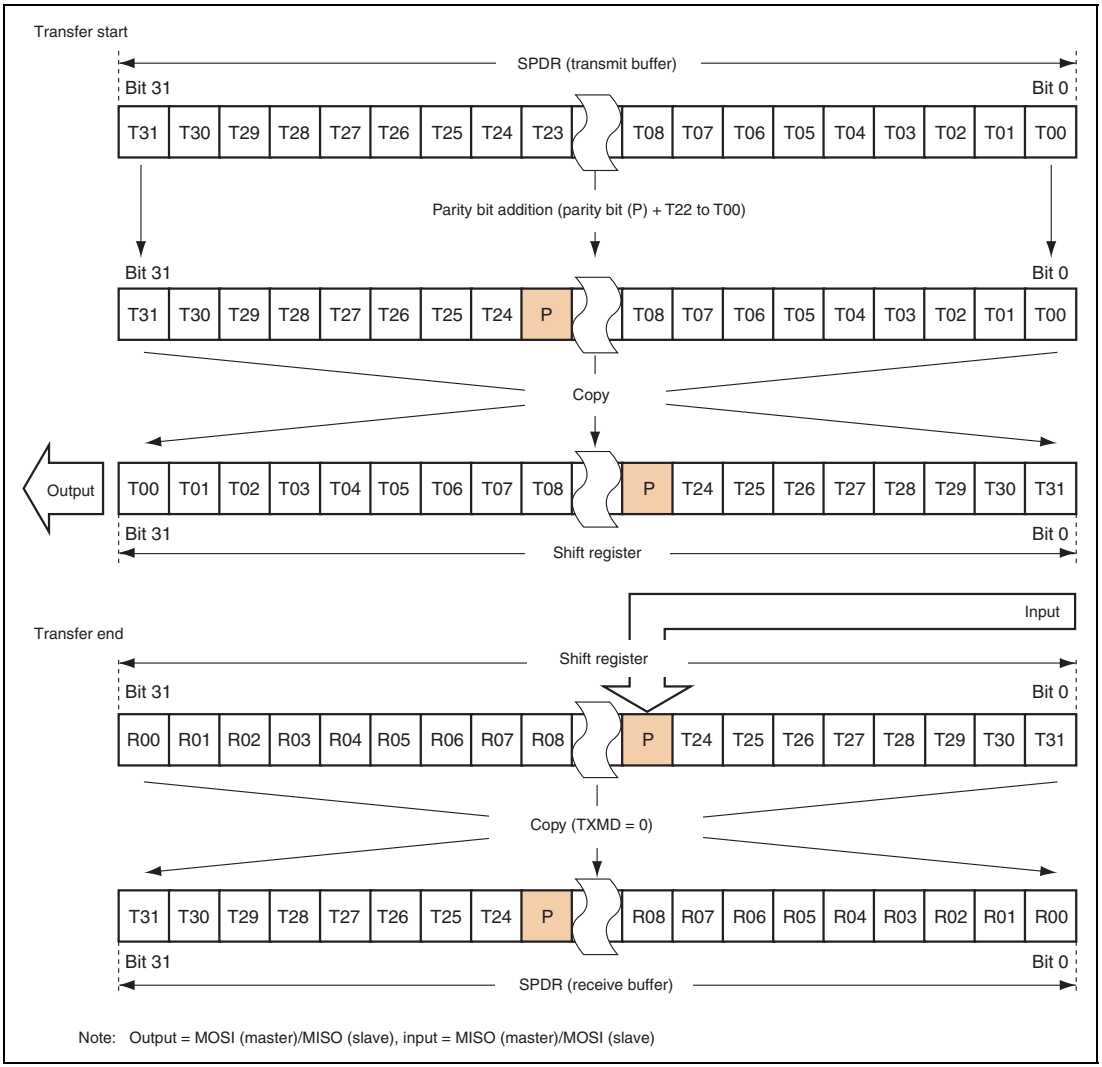


Figure 15.20 LSB First Transfer (24-Bit Data, Parity Function Enabled)

15.4.6 Communication Operating Mode

Full-duplex synchronous serial communications or transmit-only operations can be selected by the communication operating mode select bit (TXMD) in the RSPI control register (SPCR). The SPDR access shown in figures15.21 and 15.22 indicates the condition of access to the RSPI data register (SPDR), where I denotes an idle cycle and W a write cycle.

(1) Full-Duplex Synchronous Serial Communications (TXMD = 0)

Figure 15.21 shows an example of operation in which the communication operating mode select bit (TXMD) in the RSPI control register (SPCR) is set to 0. In the example of figure 15.21, the RSPI performs an 8-bit serial transfer in which the SPFC1 and SPFC0 bits in the RSPI data control register (SPDCR) are 00, and the CPHA bit is 1 and the CPOL bit is 0 in the RSPI command register (SPCMD). The numbers given under the RSPCK waveform represent the number of RSPCK cycles (i.e., the number of transferred bits).

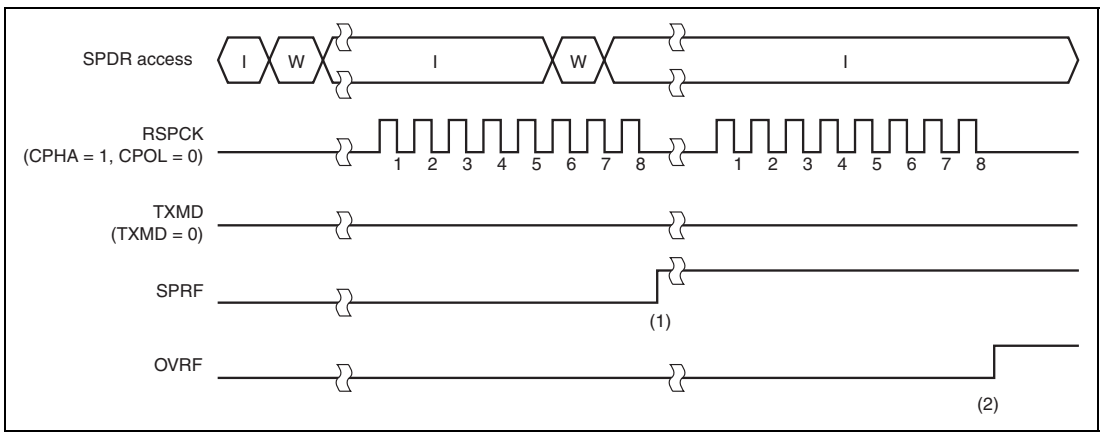


Figure 15.21 Operation Example of TXMD = 0

The operation of the flags at timings shown in steps (1) and (2) in the figure is described below.

1. When the serial transfer ends with the receive buffer of SPDR being empty, the RSPI sets the SPRF bit to 1, and copies the receive data in the shift register to the receive buffer.
2. If the serial transfer ends with the receive data from the previous serial transfer stored in the receive buffer of SPDR, the RSPI sets the OVRF bit to 1 and discards the receive data in the shift register.

In full-duplex synchronous serial communications (TXMD = 0), the transmit data is transmitted and the receive data is received. Therefore, the SPRF and OVRF bits are set to 1 at the timings of (1) and (2), respectively.

(2) Transmit-Only Operations (TXMD = 1)

Figure 15.22 shows an example of operation in which the communication operating mode select bit (TXMD) in the RSPI control register (SPCR) is set to 1. In the example of figure 15.22, the RSPI performs an 8-bit serial transfer in which the SPFC1 and SPFC0 bits in the RSPI data control register (SPDCR) are 00, and the CPHA bit is 1 and the CPOL bit is 0 in the RSPI command register (SPCMD). The numbers given under the RSPCK waveform represent the number of RSPCK cycles (i.e., the number of transferred bits).

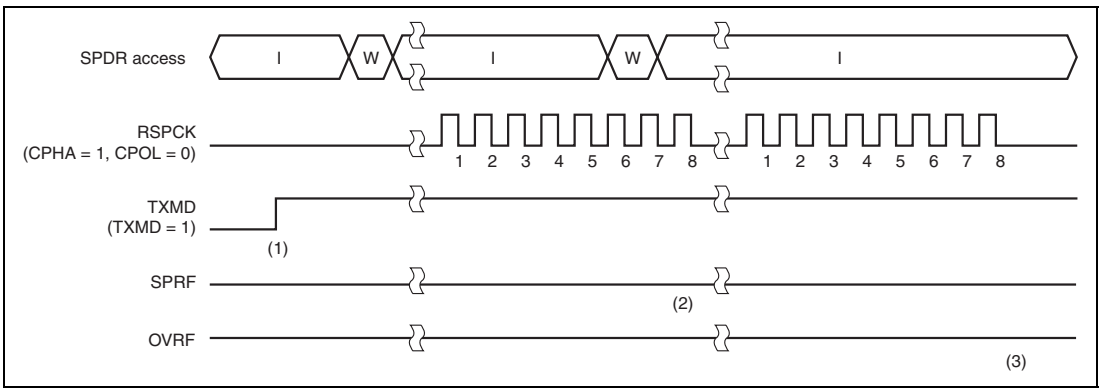


Figure 15.22 Operation Example of TXMD = 1

The operation of the flags at timings shown in steps (1) to (3) in the figure is described below.

1. Confirm that the SPRF and OVRF bits are 0 before shifting to transmit-only operations (TXMD = 1).
2. When the serial transfer ends with the receive buffer of SPDR being empty in transmit-only operations (TXMD = 1), the SPRF bit retains the value of 0, and the data in the shift register is not copied to the receive buffer.
3. Since the receive buffer of SPDR does not contain the receive data from the previous serial transfer, even though the serial transfer ends, the OVRF bit retains the value of 0, and the data in the shift register is not copied to the receive buffer.

In transmit-only operations (TXMD = 1), the transmit data is transmitted but the receive data is not received. Therefore, the SPRF and OVRF bits retain the value of 0 at the timings of (1), (2), and (3).

15.4.7 Transmit Buffer Empty/Receive Buffer Full Flags

Figure 15.23 shows an example of operation of the RSPI transmit buffer empty flag (SPTEF) and the RSPI receive buffer full flag in the RSPI status register (SPSR). The SPDR access shown in figure 15.23 indicates the condition of access to the RSPI data register (SPDR), where I denotes an idle cycle, W a write cycle, and R a read cycle. In the example of figure 15.23, the RSPI performs an 8-bit serial transfer in which the TXMD bit in the RSPI control register (SPCR) is 0, the SPFC1 and SPFC0 bits in the RSPI data control register (SPDCR) are 00, and the CPHA bit is 1 and the CPOL bit is 0 in the RSPI command register (SPCMD). The numbers given under the RSPCK waveform represent the number of RSPCK cycles (i.e., the number of transferred bits).

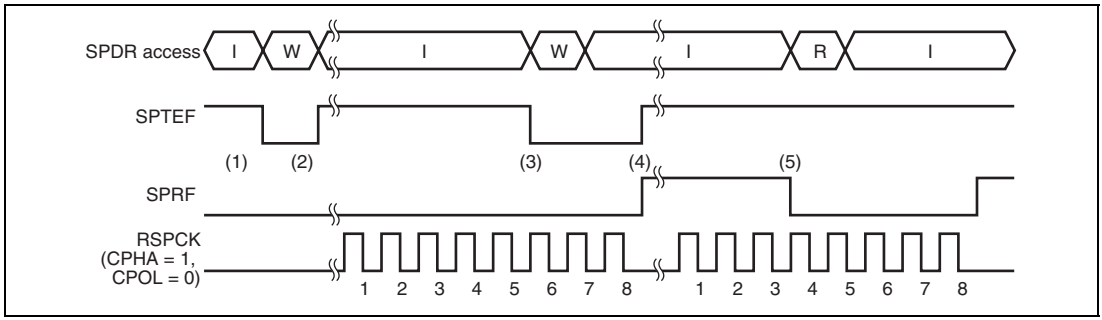


Figure 15.23 SPTEF and SPRF Bit Operation Example

The operation of the flags at timings shown in steps (1) to (5) in the figure is described below.

1. When transmit data is written to SPDR when the transmit buffer of SPDR is empty, the RSPI sets the SPTEF bit to 0, and writes data to the transmit buffer, with no change in the SPRF flag.
2. If the shift register is empty, the RSPI sets the SPTEF bit to 1, and copies the data in the transmit buffer to the shift register, with no change in the SPRF flag. How a serial transfer is started depends on the mode of the RSPI. For details, see section 15.4.10, SPI Operation, and section 15.4.11, Clock Synchronous Operation.
3. When transmit data is written to SPDR with the transmit buffer of SPDR being empty, the RSPI sets the SPTEF bit to 0, and writes data to the transmit buffer, while the SPRF flag remains unchanged. Because the data being transferred serially is stored in the shift register, the RSPI does not copy the data in the transmit buffer to the shift register.

4. When the serial transfer ends with the receive buffer of SPDR being empty, the RSPI sets the SPRF bit to 1, and copies the receive data in the shift register to the receive buffer. Because the shift register becomes empty upon completion of serial transfer, when the transmit buffer had been full before the serial transfer ended, the RSPI sets the SPTEF bit to 1, and copies the data in the transmit buffer to the shift register. Even when received data is not copied from the shift register to the receive buffer in an overrun error status, upon completion of the serial transfer the RSPI determines that the shift register is empty, and as a result data transfer from the transmit buffer to the shift register is enabled.
5. When SPDR is read with the receive buffer being full, the RSPI sets the SPRF bit to 0, and sends the data in the receive buffer to the bus inside the chip.

If SPDR is written to when the SPTEF bit is 0, the RSPI does not update the data in the transmit buffer. When writing to SPDR, make sure that the SPTEF bit is 1. The status that the SPTEF bit is 1 can be checked by reading SPSR or by using an RSPI transmit interrupt. To use an RSPI transmit interrupt, set the SPTIE bit in SPCR to 1.

If the RSPI is disabled (the SPE bit in SPCR being 0), the SPTEF bit is initialized to 1. For this reason, setting the SPTIE bit to 1 when the RSPI is disabled generates an RSPI transmit interrupt.

When serial transfer ends with the SPRF bit being 1, the RSPI does not copy data from the shift register to the receive buffer, and detects an overrun error (see section 15.4.8, Error Detection). To prevent a receive data overrun error, set the SPRF bit to 0 before the serial transfer ends. The status that the SPRF bit is 1 can be checked by either reading SPSR or by using an RSPI receive interrupt. To use an RSPI receive interrupt, set the SPRIE bit in SPCR to 1.

15.4.8 Error Detection

In the normal RSPI serial transfer, the data written from the RSPI data register (SPDR) to the transmit buffer is serially transmitted, and the serially received data can be read from the receive buffer of SPDR. If access is made to SPDR, depending on the status of the transmit buffer/receive buffer or the status of the RSPI at the beginning or end of serial transfer, in some cases non-normal transfers can be executed.

If a non-normal transfer operation occurs, the RSPI detects the event as an overrun error, a parity error, or a mode fault error. Table 15.8 shows the relationship between non-normal transfer operations and the RSPI's error detection function.

Table 15.8 Relationship between Non-Normal Transfer Operations and RSPI Error Detection Function

	Occurrence Condition	RSPI Operation	Error Detection
A	SPDR is written when the transmit buffer is full.	Keeps the contents of the transmit buffer. Missing write data.	None
B	Serial transfer is started in slave mode when transmit data is still not loaded on the shift register.	Data received in previous serial transfer is serially transmitted.	None
C	SPDR is read when the receive buffer is empty.	Previously received serial data is output.	None
D	Serial transfer terminates when the receive buffer is full.	Keeps the contents of the receive buffer. Missing serial receive data.	Overrun error detection
E	An incorrect parity bit is received when the parity function is enabled in full-duplex synchronous serial communications.	Parity error flag asserted.	Parity error detection
F	The SSL0 input signal is asserted when the serial transfer is idle in multi-master mode.	Driving of the RSPCK, MOSI, SSL1 to SSL3 output signals stopped. RSPI disabled.	Mode fault error detection
G	The SSL0 input signal is asserted during serial transfer in multi-master mode.	Serial transfer suspended. Missing send/receive data. Driving of the RSPCK, MOSI, SSL1 to SSL3 output signals stopped. RSPI disabled.	Mode fault error detection
H	The SSL0 input signal is negated during serial transfer in slave mode.	Serial transfer suspended. Missing send/receive data. Driving of the MISO output signal stopped. RSPI disabled.	Mode fault error detection

On operation A shown in table 15.8, the RSPI does not detect an error. To prevent data omission during the writing to SPDR, write operations to SPDR should be executed when the SPTEF bit in the RSPI status register (SPSR) is 1.

Likewise, the RSPI does not detect an error on operation B. In a serial transfer that was started before the shift register was updated, the RSPI sends the data that was received in the previous serial transfer, and does not treat the operation indicated in B as an error. Note that the received data from the previous serial transfer is retained in the receive buffer of SPDR, thus it can be correctly read (if SPDR is not read before the end of the serial transfer, an overrun error may result).

Similarly, the RSPI does not detect an error on operation C. To prevent extraneous data from being read, SPDR read operation should be executed when the SPRF bit in SPSR is 1.

An overrun error shown in D is described in detail in section 15.4.8 (1), Overrun Error. A parity error shown in E is described in detail in section 15.4.8 (2), Parity Error. A mode fault error shown in F to H is described in detail in section 15.4.8 (3), Mode Fault Error. On operations of the SPTEF and SPRF bits in SPSR, see section 15.4.7, Transmit Buffer Empty/Receive Buffer Full Flags.

(1) Overrun Error

If serial transfer ends when the receive buffer of the RSPI data register (SPDR) is full, the RSPI detects an overrun error, and sets the OVRF bit in SPSR to 1. When the OVRF bit is 1, the RSPI does not copy data from the shift register to the receive buffer so that the data prior to the occurrence of the error is retained in the receive buffer. To reset the OVRF bit in SPSR to 0, either perform a system reset, or write a 0 to the OVRF bit after the CPU has read SPSR with the OVRF bit set to 1.

Figure 15.24 shows an example of operation of the SPRF and OVRF bits in SPSR. The SPSR and SPDR accesses shown in figure 15.24 indicates the condition of accesses to SPSR and SPDR, respectively, where I denotes an idle cycle, W a write cycle, and R a read cycle. In the example of figure 15.24, the RSPI performs an 8-bit serial transfer in which the CPHA bit is 1 and the CPOL bit is 0 in the RSPI command register (SPCMD). The numbers given under the RSPCK waveform represent the number of RSPCK cycles (i.e., the number of transferred bits).

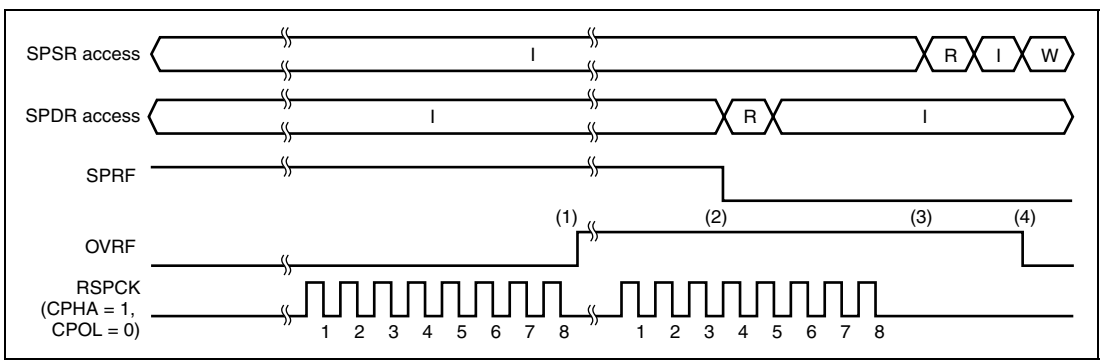


Figure 15.24 SPRF and OVRF Bit Operation Example

The operation of the flags at the timing shown in steps (1) to (4) in the figure is described below.

1. If a serial transfer terminates with the SPRF bit being 1 (receive buffer full), the RSPI detects an overrun error, and sets the OVRF bit to 1. The RSPI does not copy the data in the shift register to the receive buffer. Even if the SPPE bit is 1, parity errors are not detected. In master mode, the RSPI copies the value of the pointer to the RSPI command register (SPCMD) to bits SPECM2 to SPECM0 in the RSPI sequence status register (SPSSR).
2. When SPDR is read, the RSPI sets the SPRF bit to 0, and outputs the data in the receive buffer to an internal bus. The receive buffer becoming empty does not clear the OVRF bit.
3. If the serial transfer terminates with the OVRF bit being 1 (an overrun error), the RSPI keeps the SPRF bit at 0 and does not update it. Likewise, the RSPI does not copy the data in the shift register to the receive buffer. Even if the SPPE bit is 1, parity errors are not detected. When in master mode, the RSPI does not update bits SPECM2 to SPECM0 in SPSSR. If, in an overrun error state, the RSPI does not copy the received data from the shift register to the receive buffer, upon termination of the serial transfer, the RSPI determines that the shift register is empty; in this manner, data transfer is enabled from the transmit buffer to the shift register.
4. If the CPU writes a 0 to the OVRF bit after reading SPSR when the OVRF bit is 1, the RSPI clears the OVRF bit.

The occurrence of an overrun can be checked either by reading SPSR or by using an RSPI error interrupt and reading SPSR. When using an RSPI error interrupt, set the SPEIE bit in the RSPI control register (SPCR) to 1. When executing a serial transfer without using an RSPI error interrupt, measures should be taken to ensure the early detection of overrun errors, such as reading SPSR immediately after SPDR is read. When the RSPI is run in master mode, the pointer value to SPCMD can be checked by reading bits SPECM2 to SPECM0 in SPSSR.

If an overrun error occurs and the OVRF bit is set to 1, normal receive operations cannot be performed until such time as the OVRF bit is cleared. The OVRF bit is cleared to 0 under the following conditions:

- After reading SPSR in a condition in which the OVRF bit is set to 1, the CPU writes a 0 to the OVRF bit.
- System reset

(2) Parity Error

When full-duplex synchronous serial communications is performed with the TXMD bit in the RSPI control register (SPCR) set to 0 and the SPPE bit in RSPI control register 2 (SPCR2) set to 1, whether there is a parity error is determined on completion of transfer. If the RSPI detects a parity error in the received data, it sets the PERF bit in the RSPI status register (SPSR) to 1. Since the RSPI does not copy data in the shift register to the receive buffer while the OVRF bit is 1, parity error detection is not performed for the received data. To reset the PERF bit in SPSR to 0, either perform a system reset, or write a 0 to the PERF bit after the CPU has read SPSR with the PERF bit set to 1.

Figure 15.25 shows an example of operation of the OVRF and PERF bits in SPSR. The SPSR access shown in figure 15.25 indicates the condition of access to SPSR, where I denotes an idle cycle, W a write cycle, and R a read cycle. In the example of figure 15.25, full-duplex synchronous serial communications is performed with the TXMD bit in the RSPI control register (SPCR) set to 0 and the SPPE bit in RSPI control register 2 (SPCR2) set to 1. The RSPI performs an 8-bit serial transfer in which the CPHA bit is 1 and the CPOL bit is 0 in the RSPI command register (SPCMD). The numbers given under the RSPCK waveform represent the number of RSPCK cycles (i.e., the number of transferred bits).

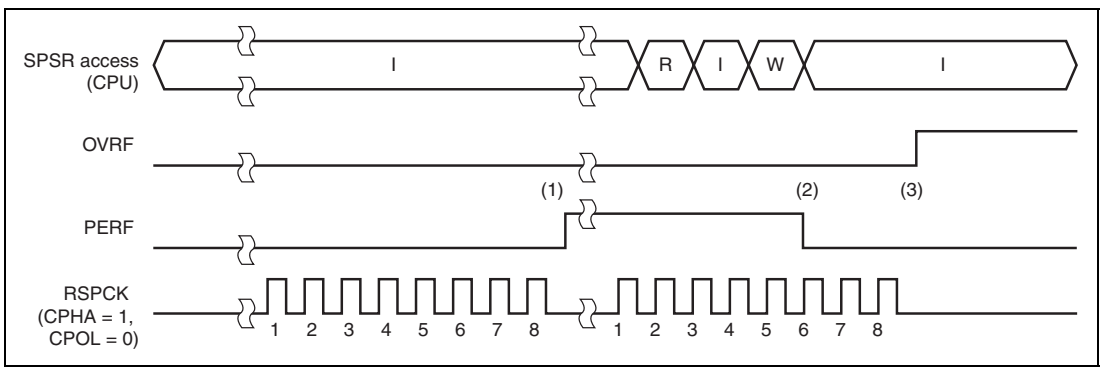


Figure 15.25 PERF Bit Operation Example

The operation of the flags at the timing shown in steps (1) to (3) in the figure is described below.

1. If a serial transfer terminates without the RSPI detecting an overrun error, the RSPI copies data in the shift register to the receive buffer. At this time, the RSPI tests the received data, and sets the PERF bit to 1 on detecting a parity error. In master mode, the RSPI copies the value of the pointer to the RSPI command register (SPCMD) to bits SPECM2 to SPECM0 in the RSPI sequence status register (SPSSR).
2. If the CPU writes a 0 to the PERF bit after reading SPSR when the PERF bit is 1, the RSPI clears the PERF bit.
3. If a serial transfer terminates with the RSPI detecting an overrun error, the RSPI does not copy data in the shift register to the receive buffer. At this time, the RSPI does not detect parity errors.

The occurrence of a parity error can be checked either by reading SPSR or by using an RSPI error interrupt and reading SPSR. When using an RSPI error interrupt, set the SPEIE bit in the RSPI control register (SPCR) to 1. When executing a serial transfer without using an RSPI error interrupt, measures should be taken to ensure the early detection of parity errors, such as reading SPSR. When the RSPI is run in master mode, the pointer value to SPCMD can be checked by reading bits SPECM2 to SPECM0 in SPSSR.

The PERF bit is cleared to 0 under the following conditions:

- After reading SPSR in a condition in which the PERF bit is set to 1, the CPU writes a 0 to the PERF bit.
- System reset

(3) Mode Fault Error

The RSPI operates in multi-master mode when the MSTR bit in the RSPI control register (SPCR) is 1, the SPMS bit is 0, and the MODFEN bit is also 1. If the active level is input with respect to the SSL0 input signal of the RSPI in multi-master mode, the RSPI detects a mode fault error irrespective of the status of the serial transfer, and sets the MODF bit in the RSPI status register (SPSR) to 1. Upon detecting the mode fault error, the RSPI copies the value of the pointer to the RSPI command register (SPCMD) to bits SPECM2 to SPECM0 in the RSPI sequence status register (SPSSR). The active level of the SSL0 signal is determined by the SSL0P bit in the RSPI slave select polarity register (SSLP).

When the MSTR bit is 0, the RSPI operates in slave mode. The RSPI detects a mode fault error if the MODFEN bit in the RSPI in slave mode is 1, and the SPMS bit is 0, and if the SSL0 input signal is negated during the serial transfer period (from the time the driving of valid data is started to the time the final valid data is fetched).

Upon detecting a mode fault error, the RSPI stops driving of the output signals and clears the SPE bit in SPCR to 0 (see section 15.4.9, Initializing RSPI). In the case of multi-master configuration, detection of a mode fault error is used to stop driving of the output signals and RSPI function, which allows the master right to be released.

The occurrence of a mode fault error can be checked either by reading SPSR or by using an RSPI error interrupt and reading SPSR. When using an RSPI error interrupt, set the SPEIE bit in the RSPI control register (SPCR) to 1. To detect a mode fault error without using an RSPI error interrupt, it is necessary to poll SPSR. When using the RSPI in master mode, reading the bits SPECM2 to SPECM0 in SPSSR allows to verify the value of the pointer to SPCMD when an error occurs.

When the MODF bit is 1, the RSPI ignores the writing of the value 1 to the SPE bit by the CPU. To enable the RSPI function after the detection of a mode fault error, the MODF bit must be set to 0. The MODF bit is cleared to 0 under the following conditions:

- After reading SPSR in a condition where the MODF bit is set to 1, the CPU writes a 0 to the MODF bit.
- System reset

15.4.9 Initializing RSPI

If the CPU writes a 0 to the SPE bit in the RSPI control register (SPCR) or the RSPI clears the SPE bit to 0 because of the detection of a mode fault error, the RSPI disables the RSPI function, and initializes a part of the module function. When a system reset is generated, the RSPI initializes all of the module function. The following describes initialization by the clearing of the SPE bit and initialization by a system reset.

(1) Initialization by Clearing the SPE Bit

When the SPE bit in SPCR is cleared, the RSPI performs the following initialization:

- Suspending any serial transfer that is being executed
- Stopping the driving of output signals (Hi-Z) in slave mode
- Initializing the internal state of the RSPI
- Initializing the SPTEF bit in the RSPI status register (SPSR)

Initialization by the clearing of the SPE bit does not initialize the control bits of the RSPI. For this reason, the RSPI can be started in the same transfer mode as prior to the initialization if the CPU resets the value 1 to the SPE bit.

The SPRF, OVRF, and MODF bits in SPSR are not initialized, nor is the value of the RSPI sequence status register (SPSSR) initialized. For this reason, even after the RSPI is initialized, data from the receive buffer can be read in order to check the status of error occurrence during an RSPI transfer.

The SPTEF bit in SPSR is initialized to 1. Therefore, if the SPTIE bit in SPCR is set to 1 after RSPI initialization, an RSPI transmit interrupt is generated. When the RSPI is initialized by the CPU, in order to disable any RSPI transmit interrupt, a 0 should be written to the SPTIE bit simultaneously with the writing of a 0 to the SPE bit. To disable any RSPI transmit interrupt after a mode fault error is detected, use an error handling routine to write a 0 to the SPTIE bit.

(2) System Reset

The initialization by a system reset completely initializes the RSPI through the initialization of all bits for controlling the RSPI, initialization of the status bits, and initialization of data registers, in addition to the requirements described in section 15.4.9 (1), Initialization by Clearing the SPE Bit.

15.4.10 SPI Operation

(1) Master Mode Operation

The only difference between single-master mode operation and multi-master mode operation lies in mode fault error detection (see section 15.4.8, Error Detection). When operating in single-master mode, the RSPI does not detect mode fault errors whereas the RSPI running in multi-master mode does detect mode fault errors. This section explains operations that are common to single-/multi-master modes.

(a) Starting a Serial Transfer

The RSPI updates the data in the transmit buffer when data is written to the RSPI data register (SPDR) with the SPTEF bit in the RSPI status register (SPSR) set to 1. If the shift register is empty in a condition where the SPTEF bit has been cleared to 0 due to the writing of 0 either after the writing to SPDR or by the writing of 0 after the value 1 is read from the SPTEF bit by the CPU, the RSPI copies the data in the transmit buffer to the shift register and starts a serial transfer. Upon copying transmit data to the shift register, the RSPI changes the status of the shift register to "full", and upon termination of serial transfer, it changes the status of the shift register to "empty". The status of the shift register cannot be referenced from the CPU.

For details on the RSPI transfer format, see section 15.4.4, Transfer Format.

(b) Terminating a Serial Transfer

Irrespective of the CPHA bit in the RSPI command register (SPCMD), the RSPI terminates the serial transfer after transmitting an RSPCK edge corresponding to the final sampling timing. If the SPRF bit in the RSPI status register (SPSR) is 0 and free space is available in the receive buffer, upon termination of serial transfer the RSPI copies data from the shift register to the receive buffer of the RSPI data register (SPDR).

It should be noted that the final sampling timing varies depending on the bit length of transfer data. In master mode, the RSPI data length depends on the settings in bits SPB3 to SPB0 in SPCMD. For details on the RSPI transfer format, see section 15.4.4, Transfer Format.

(c) Sequence Control

The transfer format that is employed in master mode is determined by the RSPI sequence control register (SPSCR), RSPI command registers 0 to 7 (SPCMD0 to SPCMD7), the RSPI bit rate register (SPBR), the RSPI clock delay register (SPCKD), the RSPI slave select negation delay register (SSLND), and the RSPI next-access delay register (SPND).

SPSCR is a register used to determine the sequence configuration for serial transfers that are executed by a master mode RSPI. The following items are set in RSPI command registers SPCMD0 to SPCMD7: SSL output signal value, MSB/LSB first, data length, some of the bit rate settings, RSPCK polarity/phase, whether SPCKD is to be referenced, whether SSLND is to be referenced, and whether SPND is to be referenced. SPBR holds some of the bit rate settings; SPCKD, an RSPI clock delay value; SSLND, an SSL negation delay; and SPND, a next-access delay value.

According to the sequence length that is assigned to SPSCR, the RSPI makes up a sequence comprised of a part or all of SPCMD0 to SPCMD7. The RSPI contains a pointer to the SPCMD that makes up the sequence. The CPU can check the value of this pointer by reading bits SPCP2 to SPCP0 in the RSPI sequence status register (SPSSR). When the SPE bit in the RSPI control register (SPCR) is set to 1 and the RSPI function is enabled, the RSPI loads the pointer to the commands in SPCMD0, and incorporates the SPCMD0 settings into the transfer format at the beginning of serial transfer. The RSPI increments the pointer each time the next-access delay period for a data transfer ends. Upon completion of the serial transfer that corresponds to the final command comprising the sequence, the RSPI sets the pointer in SPCMD0, and in this manner the sequence is executed repeatedly.

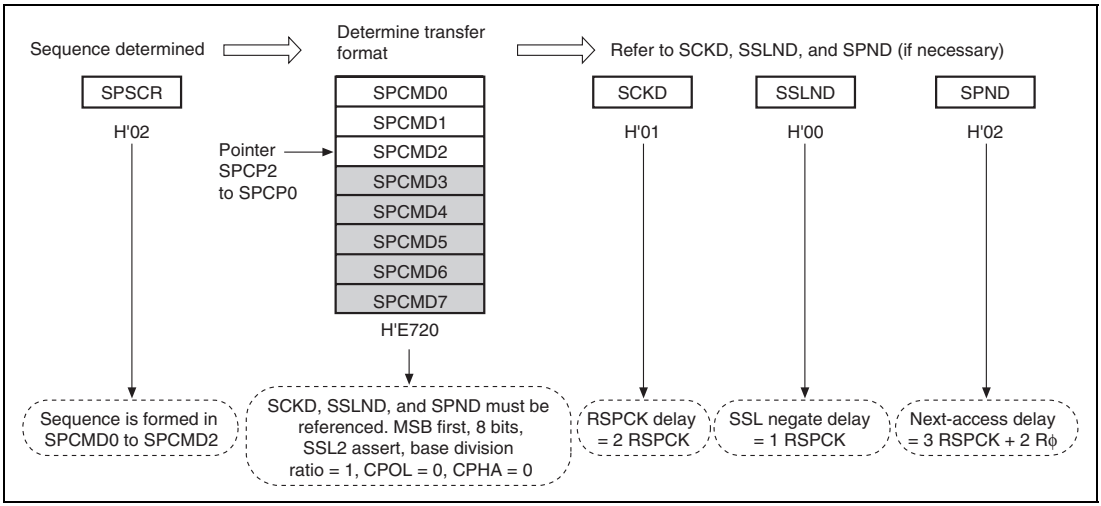


Figure 15.26 Determination Procedure of Serial Transfer Mode in Master Mode

(d) Burst Transfer

If the SSLKP bit in the RSPI command register (SPCMD) that the RSPI references during the current serial transfer is 1, the RSPI keeps the SSL signal level during the serial transfer until the beginning of the SSL signal assertion for the next serial transfer. If the SSL signal level for the next serial transfer is the same as the SSL signal level for the current serial transfer, the RSPI can execute continuous serial transfers while keeping the SSL signal assertion status (burst transfer).

Figure 15.27 shows an example of an SSL signal operation for the case where a burst transfer is implemented using SPCMD0 and SPCMD1 settings. The text below explains the RSPI operations (1) to (7) as shown in figure 15.27. It should be noted that the polarity of the SSL output signal depends on the settings in the RSPI slave select polarity register (SSLP).

1. Based on SPCMD0, the RSPI asserts the SSL signal and inserts RSPCK delays.
2. The RSPI executes serial transfers according to SPCMD0.
3. The RSPI inserts SSL negation delays.
4. Because the SSLKP bit in SPCMD0 is 1, the RSPI keeps the SSL signal value on SPCMD0. This period is sustained, at the shortest, for a period equal to the next-access delay of SPCMD0. If the shift register is empty after the passage of a minimum period, this period is sustained until such time as the transmit data is stored in the shift register for another transfer.
5. Based on SPCMD1, the RSPI asserts the SSL signal and inserts RSPCK delays.
6. The RSPI executes serial transfers according to SPCMD1.
7. Because the SSLKP bit in SPCMD1 is 0, the RSPI negates the SSL signal. In addition, a next-access delay is inserted according to SPCMD1.

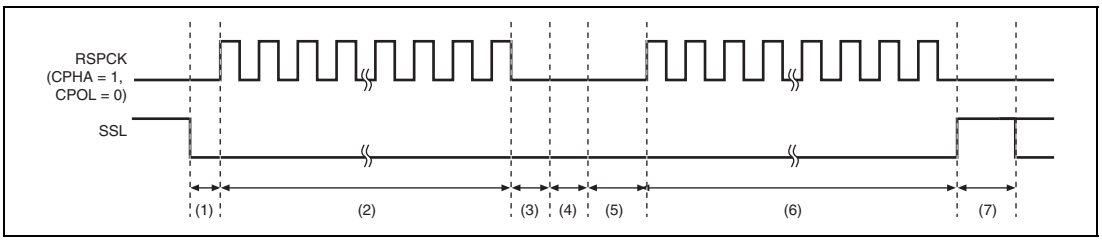


Figure 15.27 Example of Burst Transfer Operation using SSLKP Bit

If the SSL signal settings in the SPCMD in which 1 is assigned to the SSLKP bit are different from the SSL signal output settings in the SPCMD to be used in the next transfer, the RSPI switches the SSL signal status to SSL signal assertion ((5) in figure 15.23) corresponding to the command for the next transfer. Note that if such an SSL signal switching occurs, the slaves that drive the MISO signal compete, and the collision of signal levels may occur.

The RSPI in master mode references the SSL signal operation within the module for the case where the SSLKP bit is not used. Even when the CPHA bit in SPCMD is 0, the RSPI can accurately start serial transfers by asserting the SSL signal for the next transfer. For this reason, burst transfers in master mode can be executed irrespective of CPHA bit settings (see section 15.4.10 (2), Slave Mode Operation).

(e) RSPCK Delay (t1)

The RSPCK delay value of the RSPI in master mode depends on SCKDEN bit settings in the RSPI command register (SPCMD) and on RSPI clock delay register (SPCKD) settings. The RSPI determines the SPCMD to be referenced during serial transfer by pointer control, and determines an RSPCK delay value during serial transfer by using the SCKDEN bit in the selected SPCMD and SPCKD, as shown in table 15.9. For a definition of RSPCK delay, see section 15.4.4, Transfer Format.

Table 15.9 Relationship among SCKDEN and SPCKD Settings and RSPCK Delay Values

SCKDEN	SPCKD	RSPCK Delay Value
0	000 to 111	1 RSPCK
1	000	1 RSPCK
	001	2 RSPCK
	010	3 RSPCK
	011	4 RSPCK
	100	5 RSPCK
	101	6 RSPCK
	110	7 RSPCK
	111	8 RSPCK

(f) SSL Negation Delay (t2)

The SSL negation delay value of the RSPI in master mode depends on SLNDEN bit settings in the RSPI command register (SPCMD) and on SSL negation delay register (SSLND) settings. The RSPI determines the SPCMD to be referenced during serial transfer by pointer control, and determines an SSL negation delay value during serial transfer by using the SLNDEN bit in the selected SPCMD and SSLND, as shown in table 15.10. For a definition of SSL negation delay, see section 15.4.4, Transfer Format.

Table 15.10 Relationship among SLNDEN and SSLND Settings and SSL Negation Delay Values

SLNDEN	SSLND	SSL Negation Delay Value
0	000 to 111	1 RSPCK
1	000	1 RSPCK
	001	2 RSPCK
	010	3 RSPCK
	011	4 RSPCK
	100	5 RSPCK
	101	6 RSPCK
	110	7 RSPCK
	111	8 RSPCK

(g) Next-Access Delay (t3)

The next-access delay value of the RSPI in master mode depends on SPNDEN bit settings in the RSPI command register (SPCMD) and on next-access delay register (SPND) settings. The RSPI determines the SPCMD to be referenced during serial transfer by pointer control, and determines a next-access delay value during serial transfer by using the SPNDEN bit in the selected SPCMD and SPND, as shown in table 15.11. For a definition of next-access delay, see section 15.4.4, Transfer Format.

Table 15.11 Relationship among SPNDEN and SPND Settings and Next-Access Delay Values

SPNDEN	SPND	Next-Access Delay Value
0	000 to 111	1 RSPCK + 2 Rϕ
1	000	1 RSPCK + 2 Rϕ
	001	2 RSPCK + 2 Rϕ
	010	3 RSPCK + 2 Rϕ
	011	4 RSPCK + 2 Rϕ
	100	5 RSPCK + 2 Rϕ
	101	6 RSPCK + 2 Rϕ
	110	7 RSPCK + 2 Rϕ
	111	8 RSPCK + 2 Rϕ

(h) Initialization Flowchart

Figure 15.28 is a flowchart illustrating an example of initialization in SPI operation when the RSPI is used in master mode. For a description of how to set up an interrupt controller, the DMAC, and input/output ports, see the descriptions given in the individual blocks.

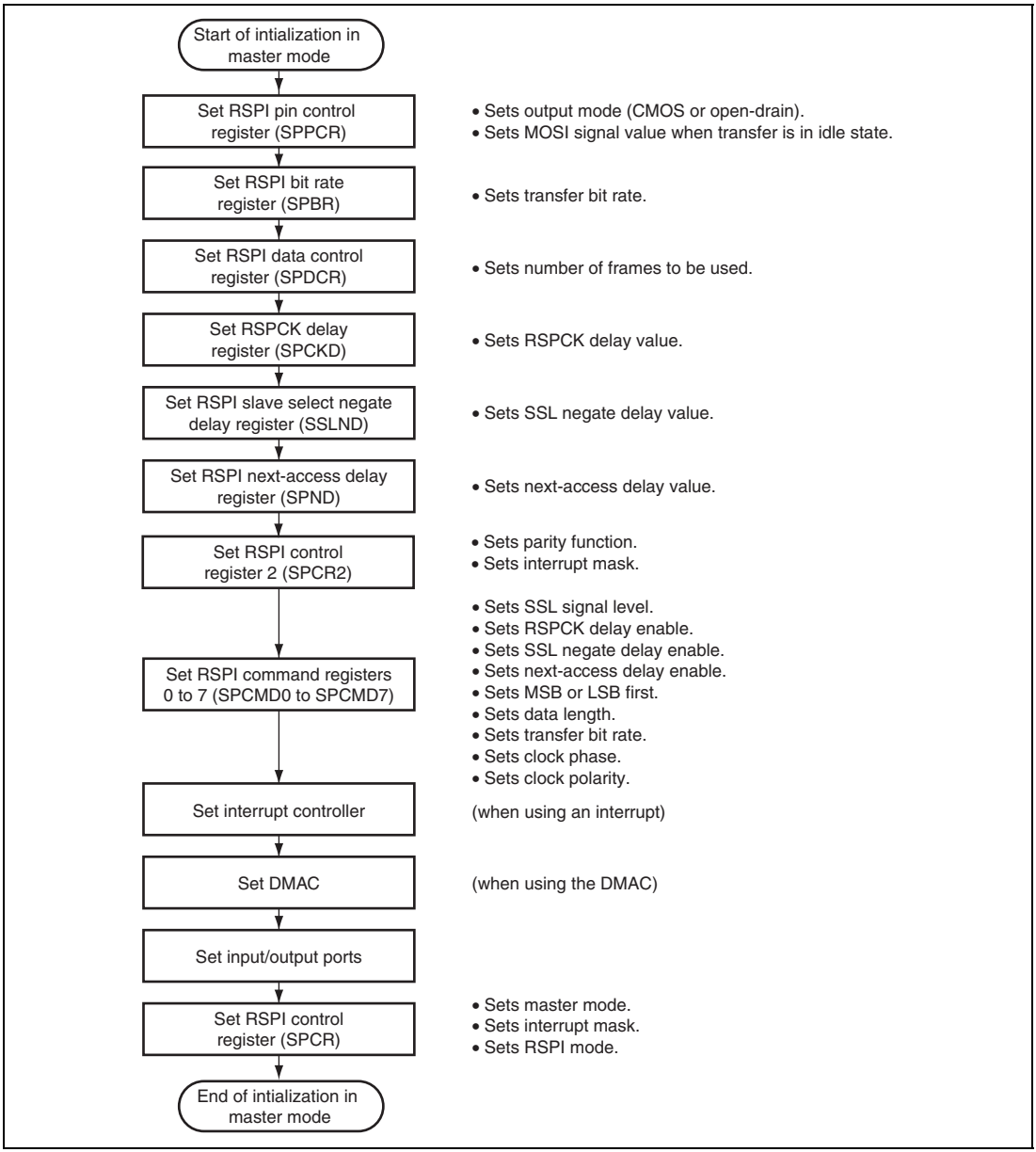
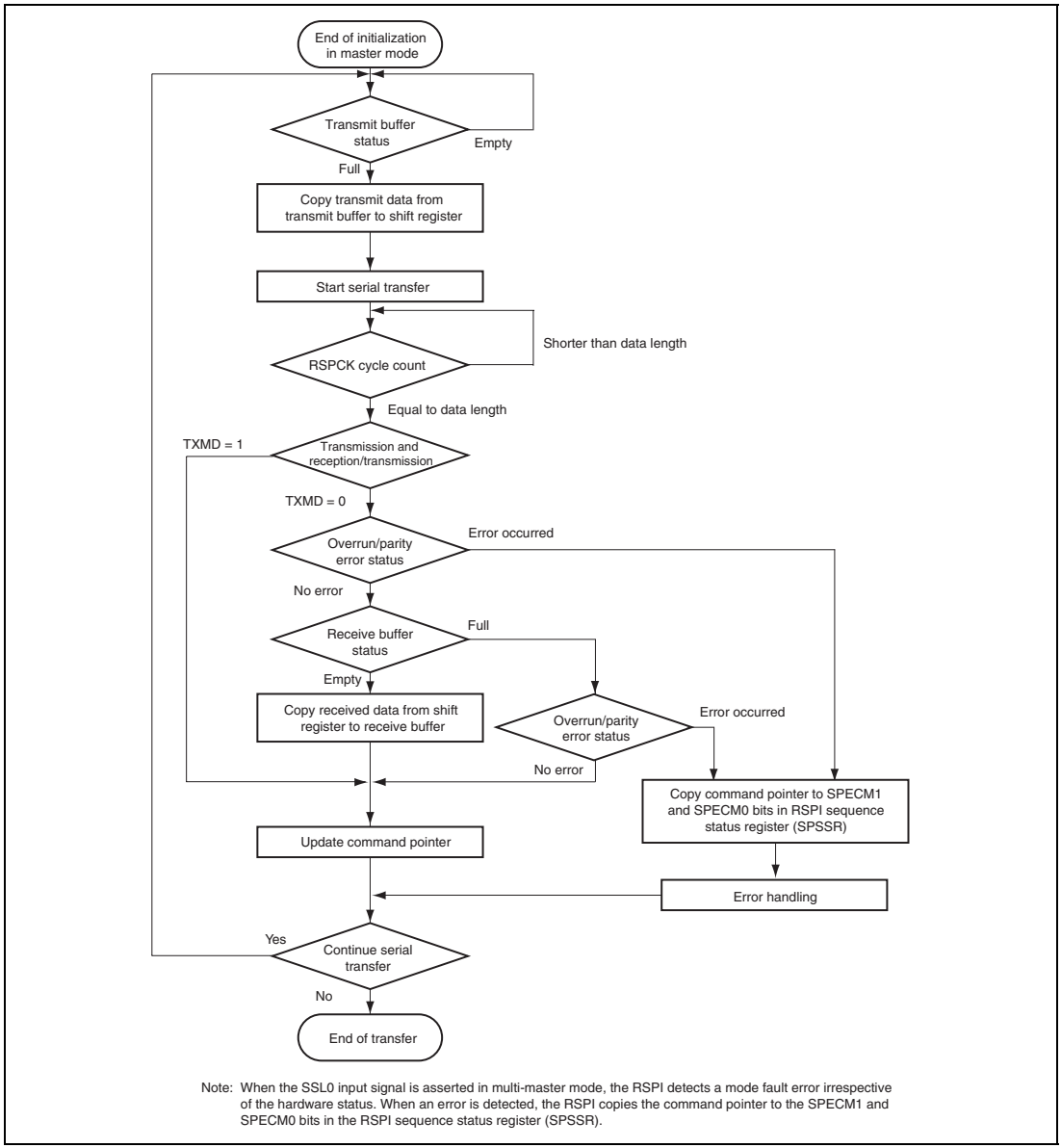


Figure 15.28 Example of Initialization Flowchart in Master Mode

(i) Transfer Operation Flowchart

Figure 15.29 is a flowchart illustrating a transfer in SPI operation when the RSPI is used in master mode.



Note: When the SSL0 input signal is asserted in multi-master mode, the RSPI detects a mode fault error irrespective of the hardware status. When an error is detected, the RSPI copies the command pointer to the SPECM1 and SPECM0 bits in the RSPI sequence status register (SPSSR).

Figure 15.29 Transfer Operation Flowchart in Master Mode

(2) Slave Mode Operation

(a) Starting a Serial Transfer

If the CPHA bit in RSPI command register 0 (SPCMD0) is 0, when detecting an SSL0 input signal assertion, the RSPI needs to start driving valid data to the MISO output signal. For this reason, when the CPHA bit is 0, the assertion of the SSL0 input signal triggers the start of a serial transfer.

If the CPHA bit is 1, when detecting the first RSPCK edge in an SSL0 signal asserted condition, the RSPI needs to start driving valid data to the MSO signal. For this reason, when the CPHA bit is 1, the first RSPCK edge in an SSL0 signal asserted condition triggers the start of a serial transfer.

When detecting the start of a serial transfer in a condition in which the shift register is empty, the RSPI changes the status of the shift register to "full", so that data cannot be copied from the transmit buffer to the shift register when serial transfer is in progress. If the shift register was full before the serial transfer started, the RSPI leaves the status of the shift register unchanged, in the full state.

Irrespective of CPHA bit settings, the timing at which the RSPI starts driving MISO output signals is the SSL0 signal assertion timing. The data which is output by the RSPI is either valid or invalid, depending on CPHA bit settings.

For details on the RSPI transfer format, see section 15.4.4, Transfer Format. The polarity of the SSL0 input signal depends on the setting of the SSL0P bit in the RSPI slave select polarity register (SSLP).

(b) Terminating a Serial Transfer

Irrespective of the CPHA bit in RSPI command register 0 (SPCMD0), the RSPI terminates the serial transfer after detecting an RSPCK edge corresponding to the final sampling timing. When the SPRF bit in the RSPI status register (SPSR) is 0 and free space is available in the receive buffer, upon termination of serial transfer the RSPI copies received data from the shift register to the receive buffer of the RSPI data register (SPDR). Irrespective of the value of the SPRF bit, upon termination of a serial transfer the RSPI changes the status of the shift register to "empty". A mode fault error occurs if the RSPI detects an SSL0 input signal negation from the beginning of serial transfer to the end of serial transfer (see section 15.4.8, Error Detection).

The final sampling timing changes depending on the bit length of the transfer data. In slave mode, the RSPI data length depends on the settings in bits SPB3 to SPB0 bits in SPCMD0. The polarity of the SSL0 input signal depends on the setting in the SSL0P bit in the RSPI slave select polarity register (SSLP). For details on the RSPI transfer format, see section 15.4.4, Transfer Format.

(c) Notes on Single-Slave Operations

If the CPHA bit in RSPI command register 0 (SPCMD0) is 0, the RSPI starts serial transfers when it detects the assertion edge for an SSL0 input signal. In the type of configuration shown in figure 15.4 as an example, if the RSPI is used in single-slave mode, the SSL0 signal is always fixed at active state. Therefore, when the CPHA bit is set to 0, the RSPI cannot correctly start a serial transfer. To correctly execute send/receive operation by the RSPI in a configuration in which the SSL0 input signal is fixed at active state, the CPHA bit should be set to 1. If there is a need for setting the CPHA bit to 0, the SSL0 input signal should not be fixed.

(d) Burst Transfer

If the CPHA bit in RSPI command register 0 (SPCMD0) is 1, continuous serial transfer (burst transfer) can be executed while retaining the assertion state for the SSL0 input signal. If the CPHA bit is 1, the period from the first RSPCK edge to the sampling timing for the reception of the final bit in an SSL0 signal active state corresponds to a serial transfer period. Even when the SSL0 input signal remains at the active level, the RSPI can accommodate burst transfers because it can detect the start of access.

If the CPHA bit is 0, for the reason given in section 15.4.10 (2) (c), Notes on Single-Slave Operations, second and subsequent serial transfers during the burst transfer cannot be executed correctly.

(e) Initialization Flowchart

Figure 15.30 is a flowchart illustrating an example of initialization in SPI operation when the RSPI is used in slave mode. For a description of how to set up an interrupt controller, the DMAC, and input/output ports, see the descriptions given in the individual blocks.

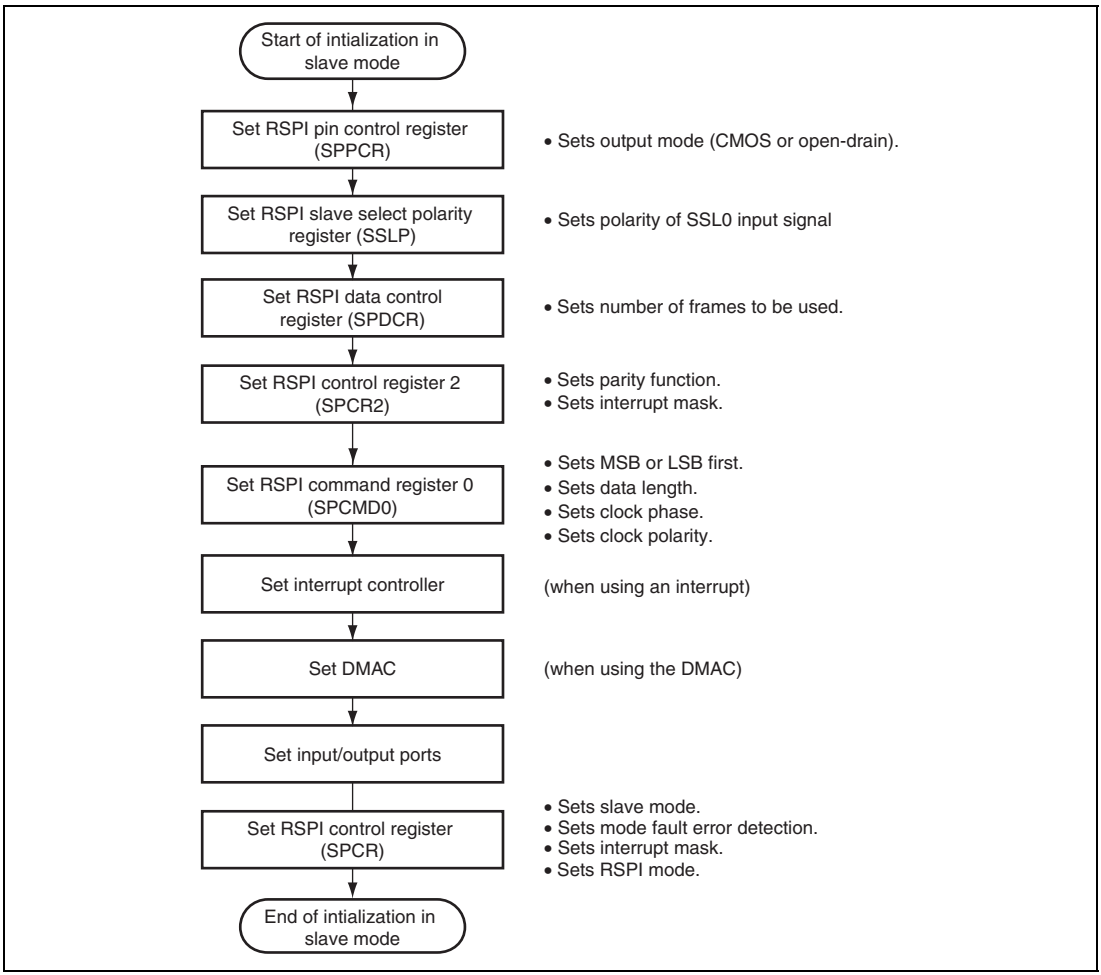


Figure 15.30 Example of Initialization Flowchart in Slave Mode

(f) Transfer Operation Flowchart (CPHA = 0)

Figure 15.31 is a flowchart illustrating a transfer in SPI operation when the RSPI is used in slave mode with the CPHA bit in the RSPI command register 0 (SPCMD0) set to 0.

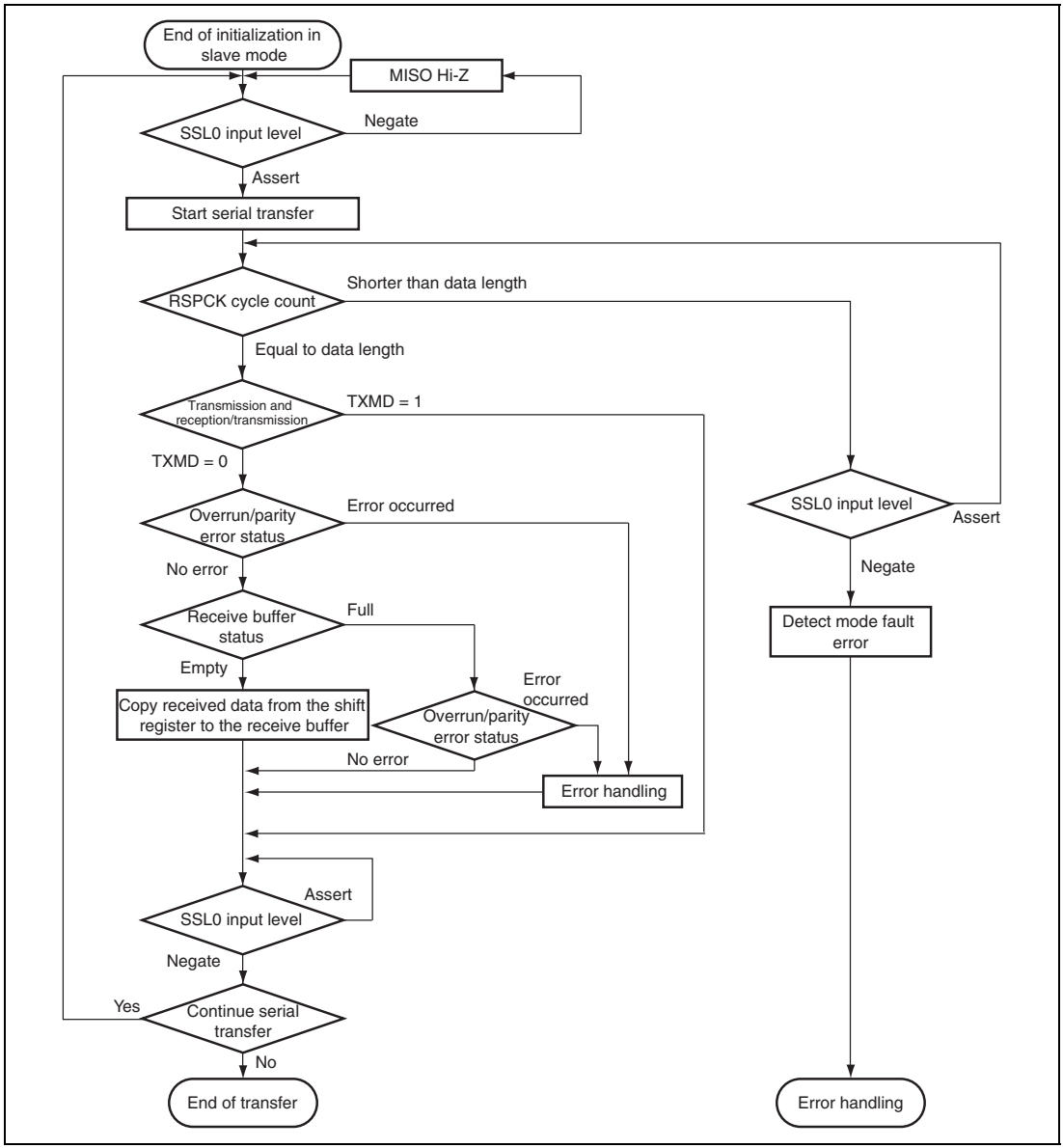


Figure 15.31 Transfer Operation Flowchart in Slave Mode (CPHA = 0)

(g) Transfer Operation Flowchart (CPHA = 1)

Figure 15.32 is a flowchart illustrating a transfer in SPI operation when the RSPI is used in slave mode with the CPHA bit in the RSPI command register 0 (SPCMD0) and the MODFEN bit in the RSPI control register (SPCR) set to 1, respectively. The subsequent operation is not guaranteed when the serial transfer is started with the MODFEN bit set to 0 and the SSL0 input level is negated with the number of RSPCK cycles shorter than the data length.

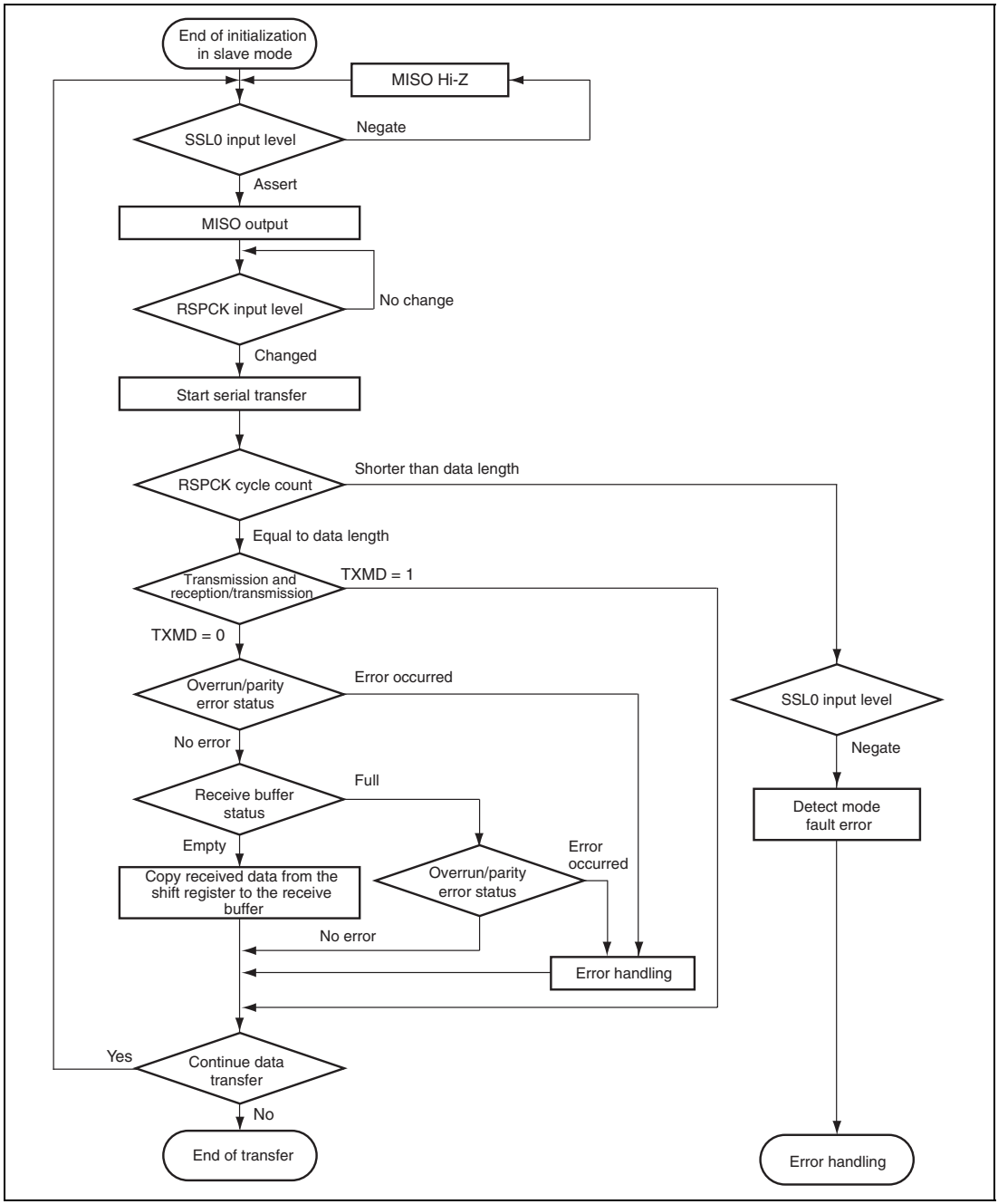


Figure 15.32 Transfer Operation Flowchart in Slave Mode (CPHA = 1)

15.4.11 Clock Synchronous Operation

Setting the SPMS bit in the RSPI control register (SPCR) to 1 selects clock synchronous operation of the RSPI. In clock synchronous operation, the SSL pins are not used, and the three pins of RSPCK, MOSI, and MISO handle communications. The SSL pins are available as I/O port pins.

Although clock synchronous operation does not require use of the SSL pins, operation of the module is the same as in SPI operation. That is, in both master and slave operations, communications can be performed with the same flow as in SPI operation. However, mode fault errors are not detected because the SSL pins are not used.

Furthermore, operation is not guaranteed if clock synchronous operation proceeds when the CPHA bit in the RSPI command register (SPCMD) is set to 0 in slave mode (MSTR = 0).

(1) Master Mode Operation

(a) Starting Serial Transfer

The RSPI updates the data in the transmit buffer when data is written to the RSPI data register (SPDR) with the SPTEF bit in the RSPI status register (SPSR) set to 1. If the shift register is empty in a condition where the SPTEF bit has been cleared to 0 due to the writing of 0 either after the writing to SPDR or by the writing of 0 after the value 1 is read from the SPTEF bit by the CPU, the RSPI copies the data in the transmit buffer to the shift register and starts a serial transfer. Upon copying transmit data to the shift register, the RSPI changes the status of the shift register to "full", and upon termination of serial transfer, it changes the status of the shift register to "empty". The status of the shift register cannot be referenced from the CPU.

For details on the RSPI transfer format, see section 15.4.4, Transfer Format.

(b) Terminating a Serial Transfer

The RSPI terminates the serial transfer after transmitting an RSPCK edge corresponding to the sampling timing. If the SPRF bit in the RSPI status register (SPSR) is 0 and free space is available in the receive buffer, upon termination of serial transfer, the RSPI copies data from the shift register to the receive buffer of the RSPI data register (SPDR).

It should be noted that the final sampling timing varies depending on the bit length of transfer data. In master mode, the RSPI data length depends on the settings in bits SPB2 to SPB0 in the RSPI command register (SPCMD). For details on the RSPI transfer format, see section 15.4.4, Transfer Format.

(c) Sequence Control

The transfer format employed in master mode is determined by the RSPI sequence control register (SPSCR), RSPI command registers 0 to 7 (SPCMD0 to SPCMD7), the RSPI bit rate register (SPBR), the RSPI clock delay register (SPCKD), the RSPI slave select negation delay register (SSLND), and the RSPI next-access delay register (SPND). Although the SSL signals are not output in clock synchronous operation, these settings are valid.

SPSCR is a register used to determine the sequence configuration for serial transfers that are executed by a master mode RSPI. The following items are set in RSPI command registers SPCMD0 to SPCMD7: SSL output signal value, MSB/LSB first, data length, some of the bit rate settings, RSPCK polarity/phase, whether SPCKD is to be referenced, whether SSLND is to be referenced, and whether SPND is to be referenced. SPBR holds some of the bit rate settings; SPCKD, an RSPI clock delay value; SSLND, an SSL negation delay; and SPND, a next-access delay value.

According to the sequence length that is assigned to SPSCR, the RSPI makes up a sequence comprised of a part or all of SPCMD0 to SPCMD7. The RSPI contains a pointer to the SPCMD that makes up the sequence. The CPU can check the value of this pointer by reading bits SPCP2 to SPCP0 in the RSPI sequence status register (SPSSR). When the SPE bit in the RSPI control register (SPCR) is set to 1 and the RSPI function is enabled, the RSPI loads the pointer to the commands in SPCMD0, and incorporates the SPCMD0 settings into the transfer format at the beginning of serial transfer. The RSPI increments the pointer each time the next-access delay period for a data transfer ends. Upon completion of the serial transfer that corresponds to the final command comprising the sequence, the RSPI sets the pointer in SPCMD0, and in this manner the sequence is executed repeatedly.

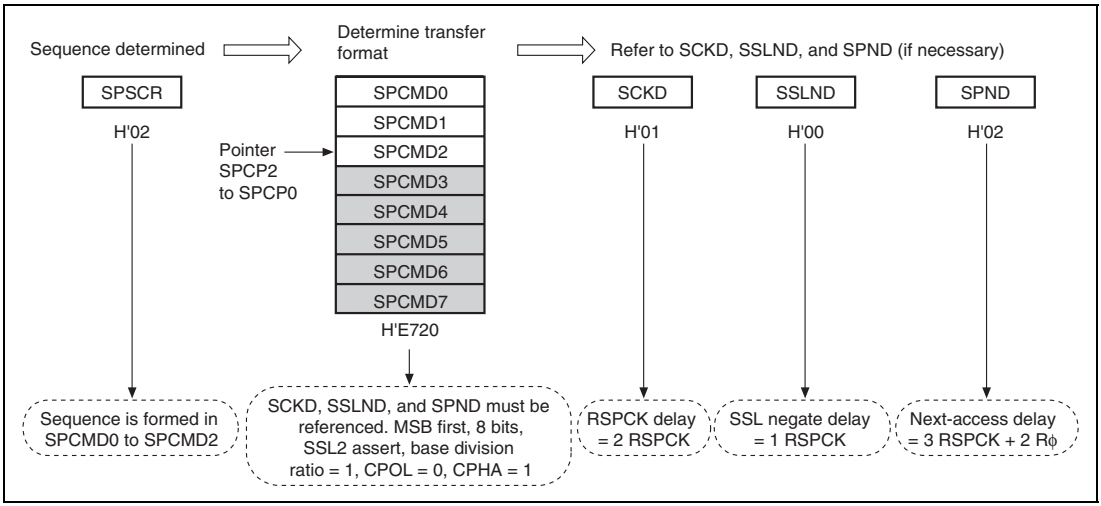


Figure 15.33 Determination Procedure of Serial Transfer Mode in Master Mode

(d) Initialization Flowchart

Figure 15.34 is a flowchart illustrating an example of initialization in clock synchronous operation when the RSPI is used in master mode. For a description of how to set up an interrupt controller, the DMAC, and input/output ports, see the descriptions given in the individual blocks.

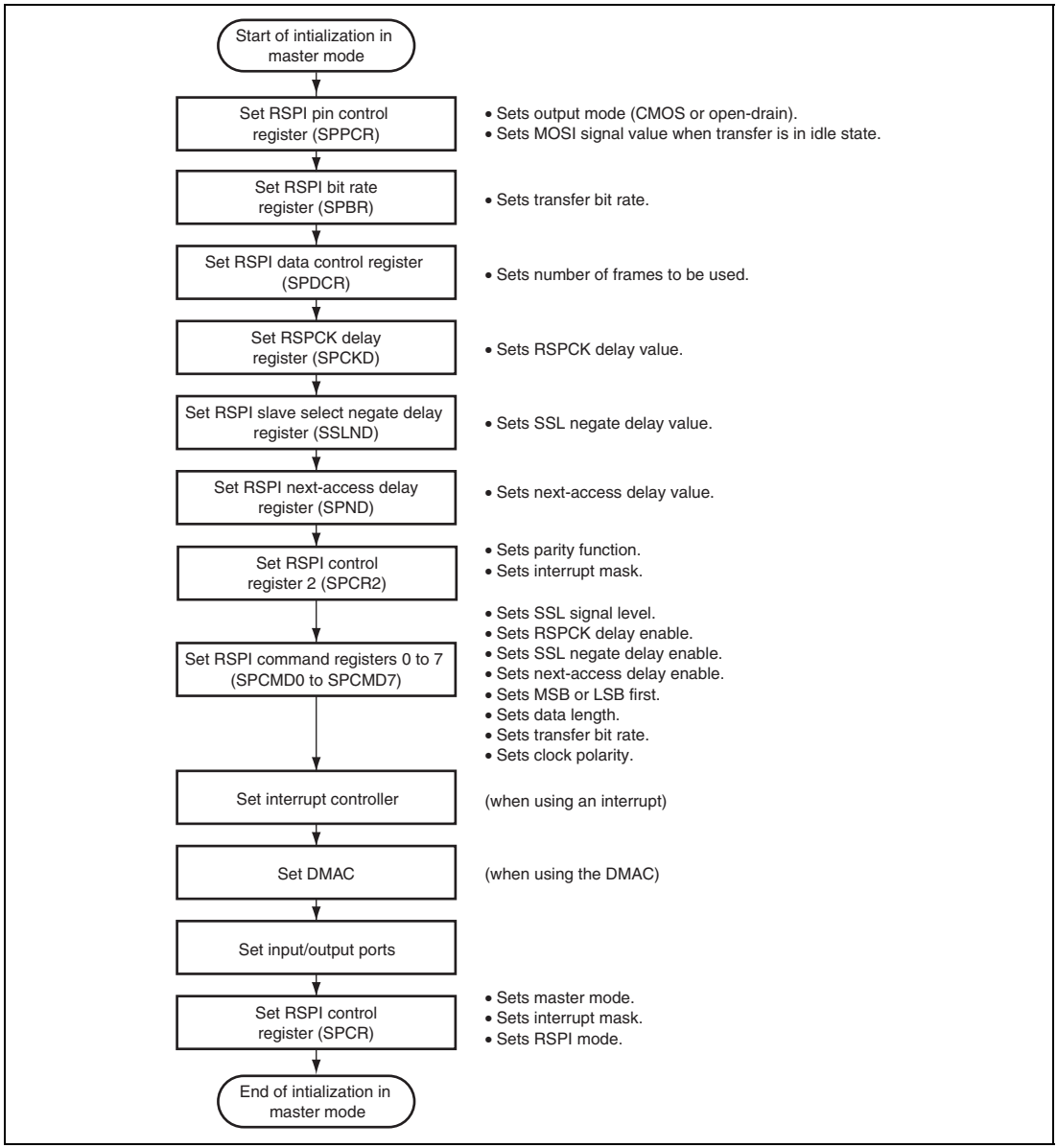


Figure 15.34 Example of Initialization Flowchart in Master Mode

(e) Transfer Operation Flowchart

Figure 15.35 is a flowchart illustrating a transfer in clock synchronous operation when the RSPI is used in master mode.

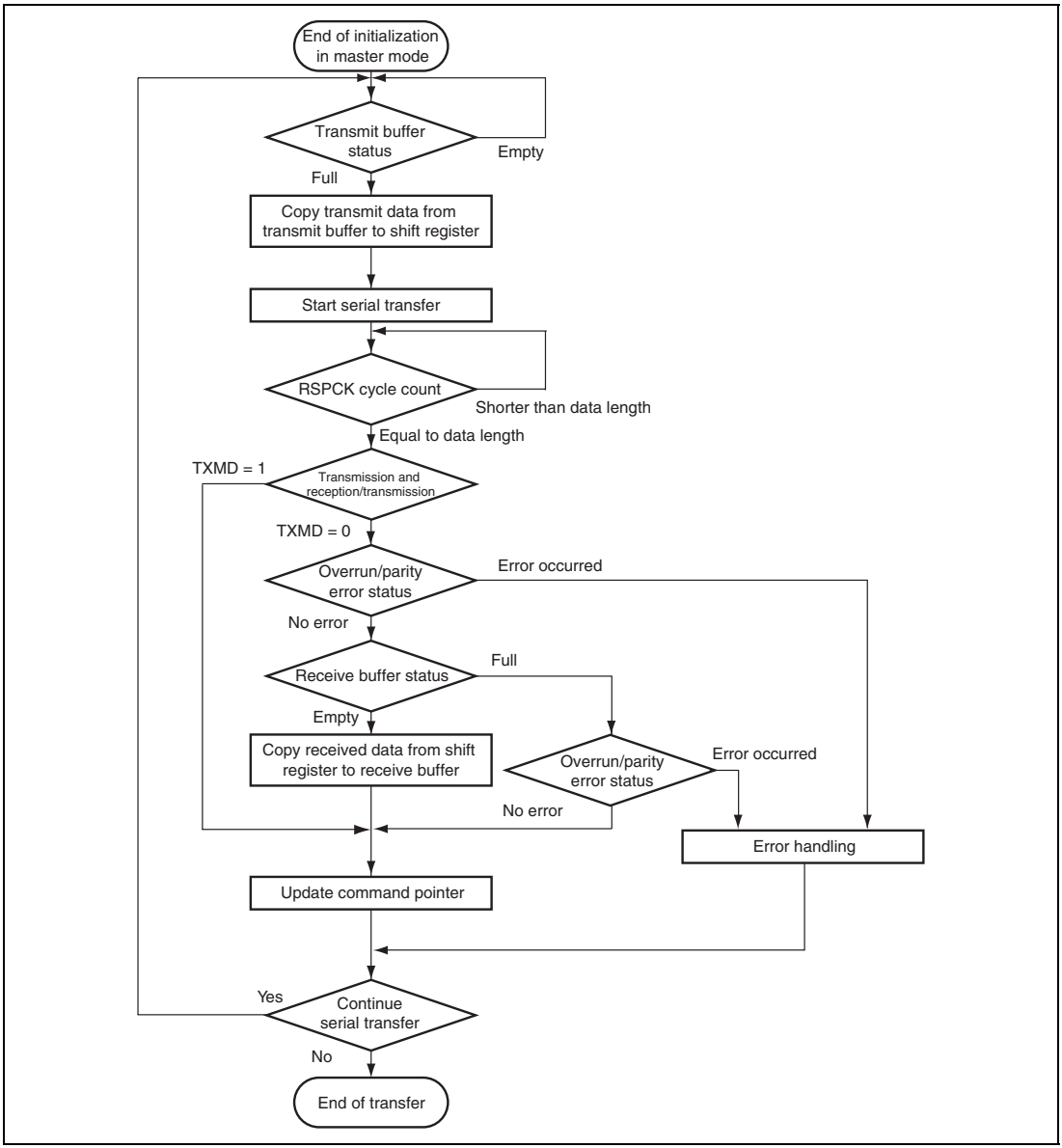


Figure 15.35 Transfer Operation Flowchart in Master Mode

(2) Slave Mode Operation

(a) Starting a Serial Transfer

When the SPMS bit in the RSPI control register (SPCR) is 1, the first RSPCK edge triggers the start of a serial transfer in the RSPI.

When detecting the start of a serial transfer in a condition in which the shift register is empty, the RSPI changes the status of the shift register to "full", so that data cannot be copied from the transmit buffer to the shift register when serial transfer is in progress. If the shift register was full before the serial transfer started, the RSPI keeps the status of the shift register unchanged, in the full state.

When the SPMS bit is 1, the RSPI always drives the MISO output signal. For details on the RSPI transfer format, see section 15.4.4, Transfer Format.

(b) Terminating a Serial Transfer

The RSPI terminates the serial transfer after detecting an RSPCK edge corresponding to the final sampling timing. When the SPRF bit in the RSPI status register (SPSR) is 0 and free space is available in the receive buffer, upon termination of serial transfer the RSPI copies received data from the shift register to the receive buffer of the RSPI data register (SPDR). Irrespective of the value of the SPRF bit, upon termination of a serial transfer the RSPI changes the status of the shift register to "empty". The final sampling timing changes depending on the bit length of the transfer data. In slave mode, the RSPI data length depends on the settings in bits SPB3 to SPB0 in SPCMD0. For details on the RSPI transfer format, see section 15.4.4, Transfer Format.

(c) Initialization Flowchart

Figure 15.36 is a flowchart illustrating an example of initialization in clock synchronous operation when the RSPI is used in slave mode. For a description of how to set up an interrupt controller, the DMAC, and input/output ports, see the descriptions given in the individual blocks.

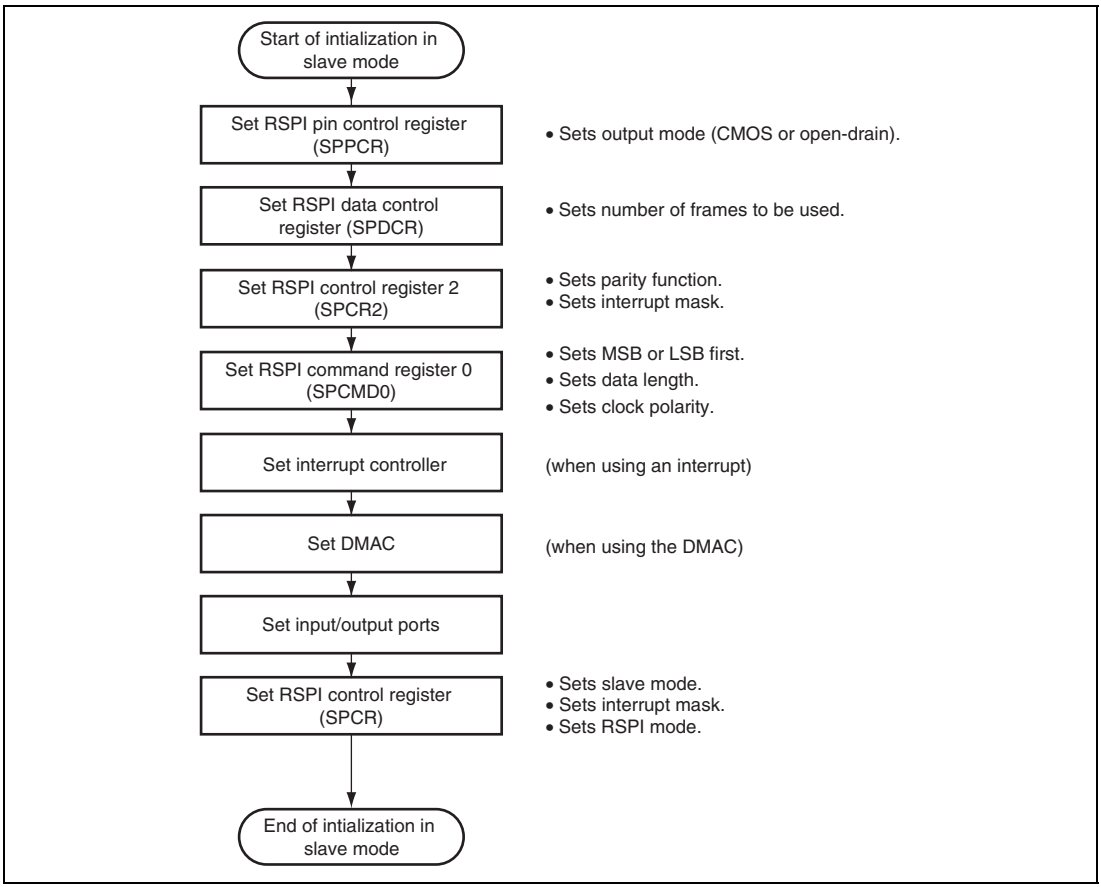


Figure 15.36 Example of Initialization Flowchart in Slave Mode

(d) Transfer Operation Flowchart

Figure 15.37 is a flowchart illustrating a transfer when the RSPI is in clock synchronous operation.

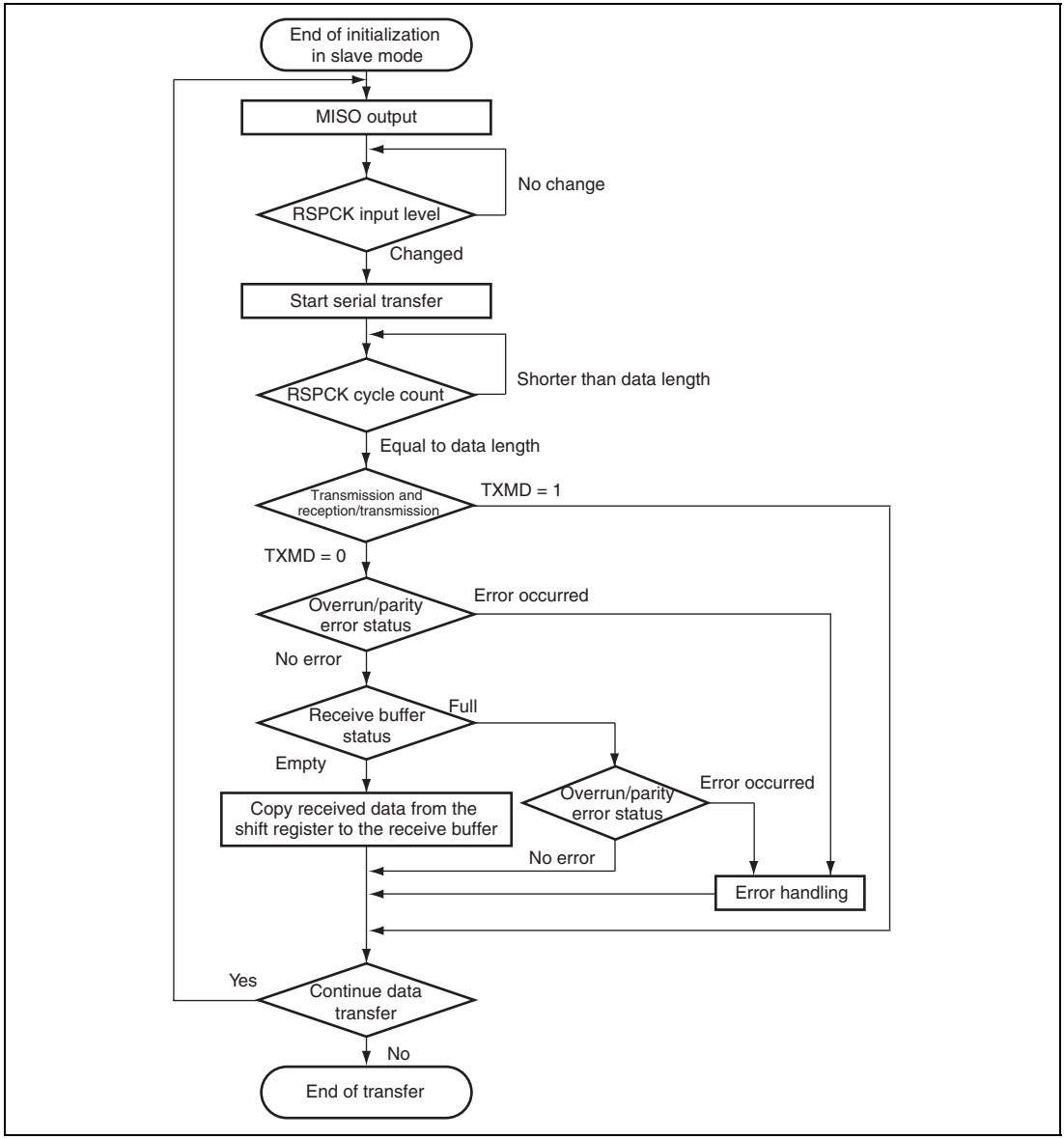


Figure 15.37 Transfer Operation Flowchart in Slave Mode (CPHA = 1)

15.4.12 Error Handling

Figures 15.38 to 15.40 show the error handling for the RSPI. The following error handling is used to return from the error state after an error occurred in master or slave mode.

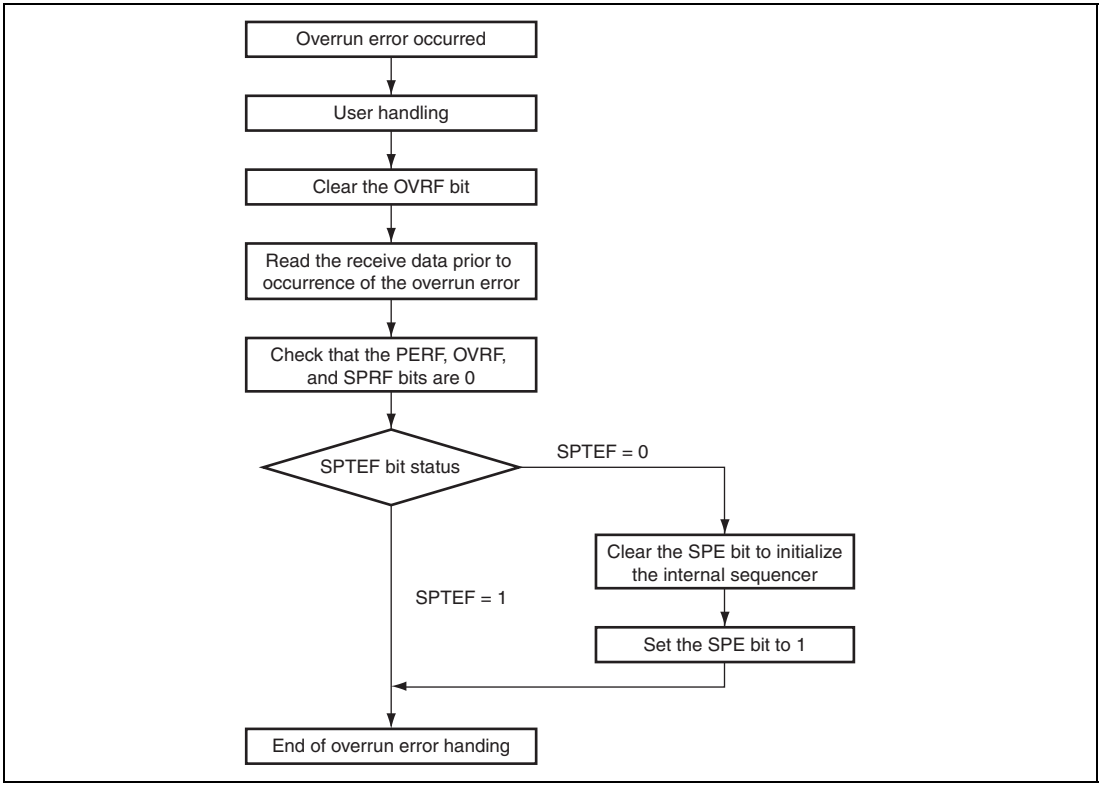


Figure 15.38 Error Handling (Overrun Error)

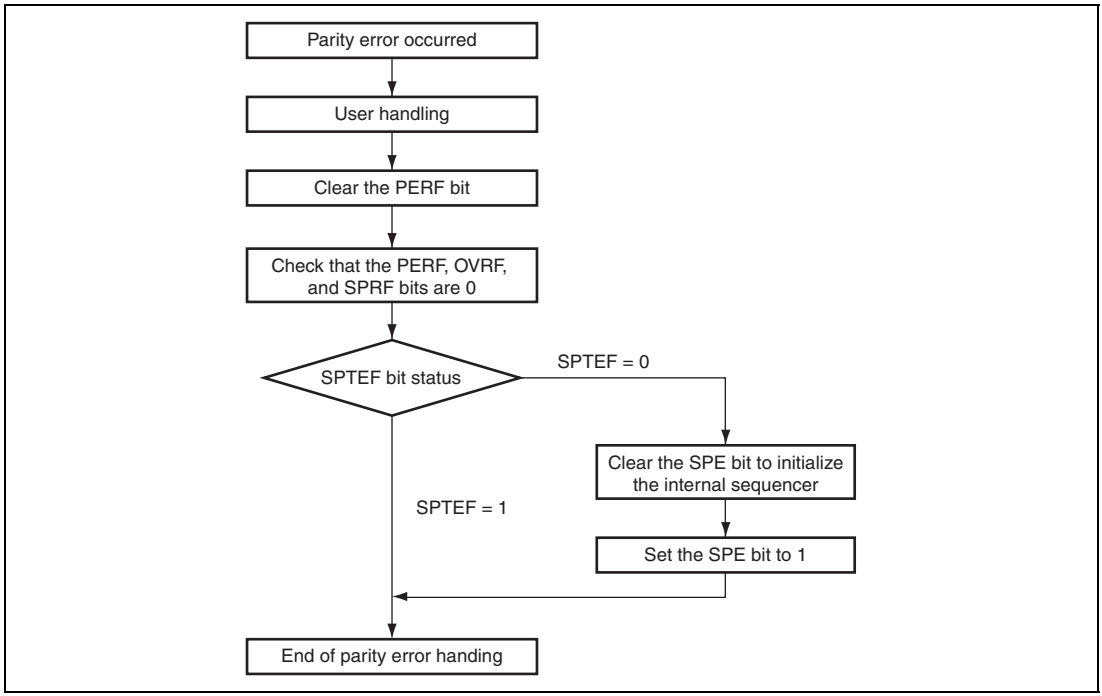


Figure 15.39 Error Handling (Parity Error)

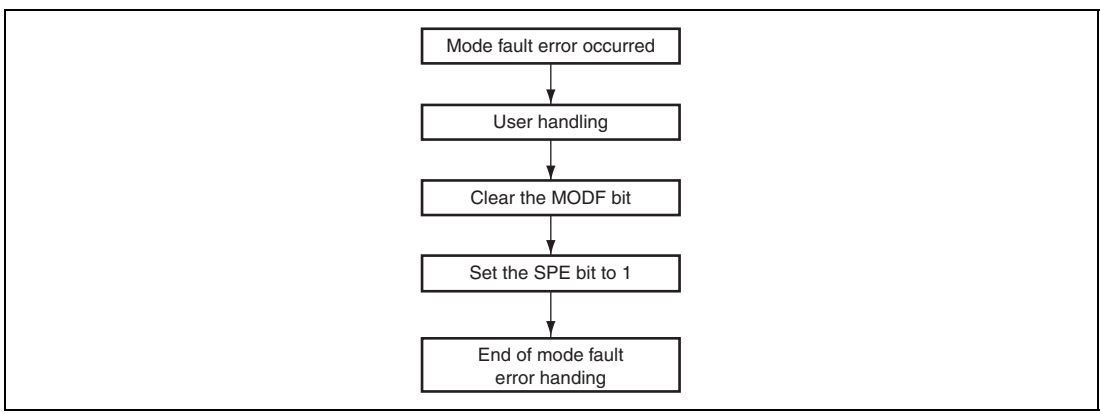


Figure 15.40 Error Handling (Mode Fault Error)

15.4.13 Loopback Mode

When the CPU writes 1 to the SPLP2 bit or SPLP bit in the RSPI pin control register (SPPCR), the RSPI shuts off the path between the MISO pin and the shift register, and between the MOSI pin and the shift register, and connects the input path and the output path of the shift register. This is called loopback mode. When a serial transfer is executed in loopback mode, the transmit data for the RSPI or the reversed transmit data for the RSPI becomes the received data for the RSPI.

The relationship between the settings of bits SPLP2 and SPLP and the received data is shown in table 15.12.

Table 15.12 SPLP2 and SPLP Bit Settings and Received Data

SPLP2	SPLP	Received Data
0	0	Input data from MOSI or MISO pin
0	1	Reversed transmit data
1	0	Transmit data
1	1	Transmit data

Figure 15.41 shows the configuration of the shift register input/output paths for the case where the RSPI in master mode is set in loopback mode (SPLP2 = 0, SPLP = 1).

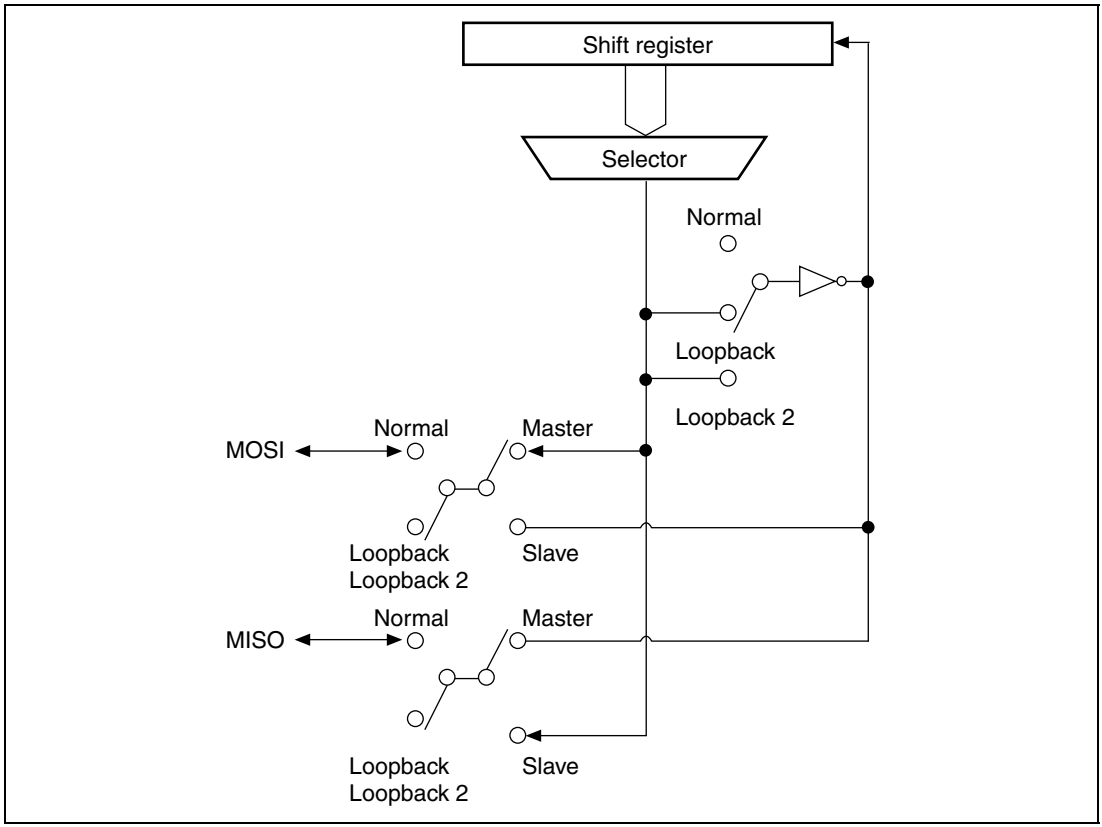


Figure 15.41 Configuration of Shift Register Input/Output Paths in Loopback Mode (Master Mode)

15.4.14 Self Diagnosis of Parity Function

The parity circuit consists of a parity adding unit for the transmit data and an error detecting unit for the receive data. Self diagnosis of the parity circuit is performed according to the flowchart in figure 15.42 in order to detect defects in the parity adding unit and error detecting unit of the parity circuit.

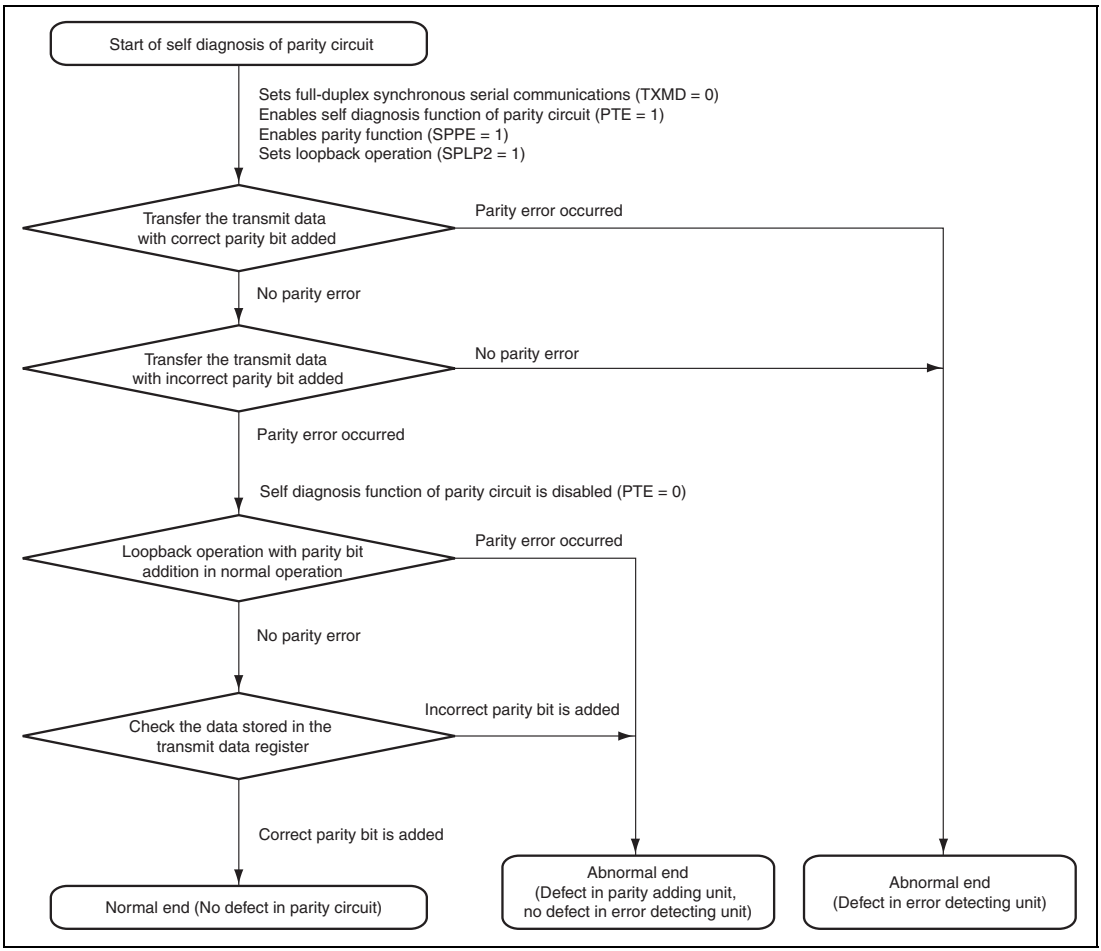


Figure 15.42 Parity Circuit Self Diagnosis Flowchart

15.4.15 Interrupt Sources

The RSPI has interrupt sources of receive buffer full, transmit buffer empty, mode fault, and overrun. In addition, the DMAC can be activated by the receive buffer full or transmit buffer empty interrupt for data transfer.

The receive buffer full interrupt request is assigned to vector address sp_rxint, the transmit buffer empty interrupt request is assigned to vector address sp_txint, and the mode fault and overrun interrupt requests are assigned to vector address sp_errint. Therefore, flag-testing must be used to determine the source of the request. Table 15.13 shows the RSPI interrupt sources.

When any of the interrupt conditions in table 15.13 is met, an interrupt is generated. Clear the interrupt sources with data transfer by the CPU or DMAC.

Table 15.13 Interrupt Sources of RSPI

Name	Interrupt Source	Abbreviation	Interrupt Condition	DMAC Activation
sp_rxint	Receive buffer full	RXI	(sprie = 1) • (sprf = 1)	Possible
sp_txint	Transmit buffer empty	TXI	(sptie = 1) • (sptef = 1)	Possible
sp_errint	Mode fault	MOI	(speie = 1) • (modf = 1)	—
	Overrun	OVI	(speie = 1) • (ovrf = 1)	—
	Parity error	PRI	(speie = 1) • (perf = 1)	—
sp_idint	RSPI idle	IDI	(spiie = 1) • (idlnf = 1)	—

Section 16 Hardware LIN (HWLIN)

Along with the SCI, the hardware LIN handles LIN communications.

16.1 Features

The hardware LIN has the following features. Figure 16.1 is a block diagram of the hardware LIN.

Master mode:

- Generation of sync breaks
- Detection of bus conflicts
- Detection of timeouts

Slave mode:

- Detection of sync breaks
- Measurement of sync fields
- SCI input control function for sync break and sync field signals
- Detection of bus conflicts
- Detection of timeouts

Note: The SCI is used for wake-up transmission. Wake-up reception is detected by an IRQ.

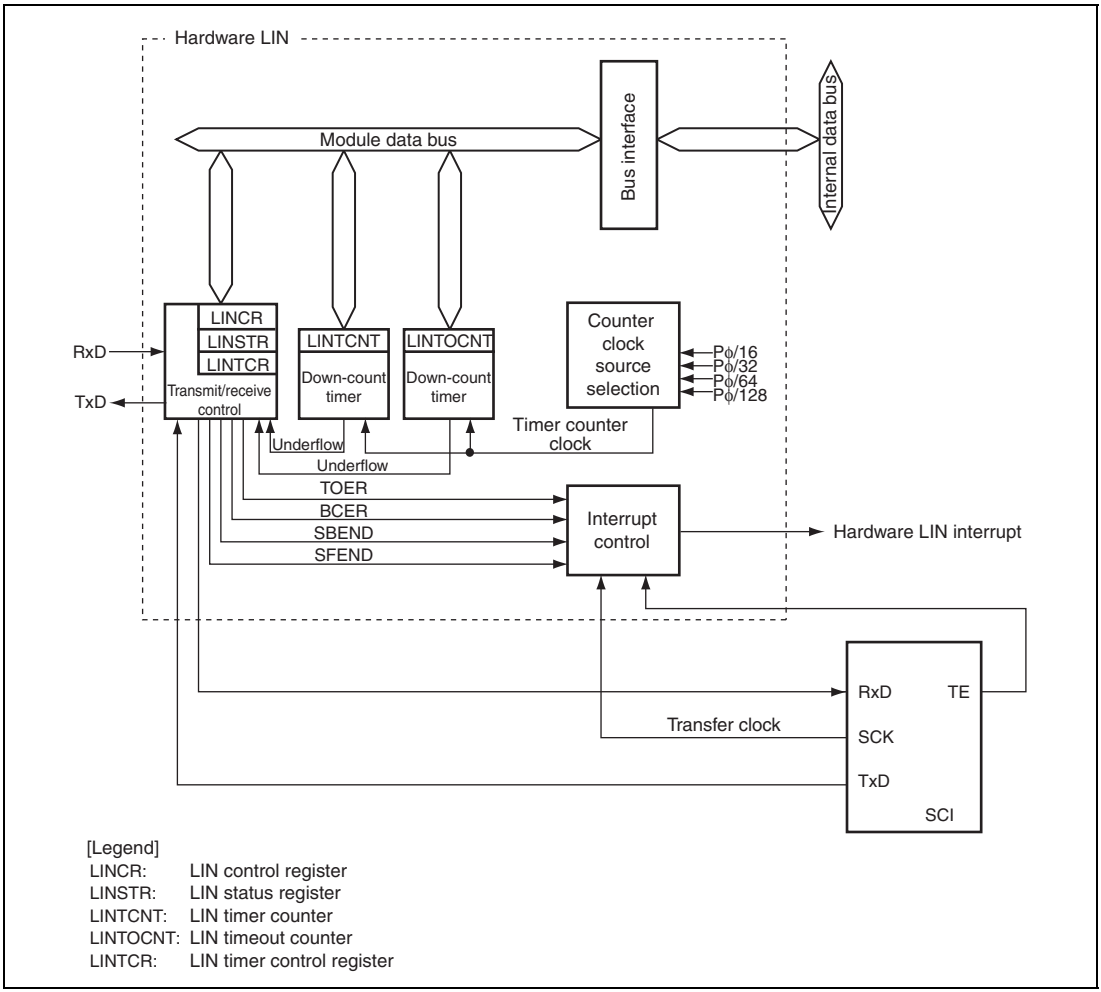


Figure 16.1 Block Diagram of Hardware LIN

16.2 Input/Output Pins

Table 16.1 shows the pin configuration of the hardware LIN.

Table 16.1 Pin Configuration

Name	Symbol	I/O	Function
Receive data input	RxD	Input	Receive data input pin for hardware LIN
Transmit data output	TxD	Output	Transmit data output pin for hardware LIN

16.3 Register Descriptions

The hardware LIN module has the following registers.

- LIN control register (LINCR)
- LIN status register (LINST)
- LIN timer control register (LINTCR)
- LIN timer counter (LINTCNT)
- LIN timeout counter (LINTOCNT)

16.3.1 LIN Control Register (LINCR)

LINCR controls the operation of the hardware LIN.

Bit	7	6	5	4	3	2	1	0
Bit Name	LINE	MSS	RXDMRS	SBSTR	RXDSF	BCIE	SBIE	SFIE
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	LINE	0	R/W	LIN Enable 0: Stops LIN operation 1: Starts LIN operation (Input to the SCI immediately after this bit has been set to 1 is prohibited.) Writing 0 initializes LINCR, LINSTR, LINTCR, LINTCNT, LINTOCNT, and the internal circuit of the hardware LIN.
6	MSS	0	R/W	Master/Slave Selection 0: Slave mode (sync break detection circuit in operation, sync field measurement circuit in operation) 1: Master mode (sync break generation circuit in operation) Stop LIN operation before switching the LIN operating mode (LINE = 0).
5	RXDMRS	0	R/W	RXD Mask Release Timing Selection (valid in slave mode) Selects the timing with which the mask control specified for RxD is released. 0: Released after detection of sync break 1: Released after detection of sync field
4	SBSTR	0	R/W	Sync Break Start (valid in slave mode) This bit is always read as 0. After setting the SBSTR bit, confirm that the RXDSF flag has been set to 1 before starting the input of a sync break. 0: Don’t care 1: Starts masking of the SCI’s RxD input
3	RXDSF	0	R	RxD Input Status Flag 0: Masking of the RxD input is released 1: RxD input is masked

Bit	Bit Name	Initial Value	R/W	Description
2	BCIE	0	R/W	Bus Conflict Interrupt Enable 0: Disables the bus conflict detection interrupt 1: Enables the bus conflict detection interrupt
1	SBIE	0	R/W	Sync Break Interrupt Enable 0: Disables the sync break completion interrupt and the sync break detection interrupt 1: Enables the sync break completion interrupt and the sync break detection interrupt
0	SFIE	0	R/W	Sync Field Interrupt Enable 0: Disables the sync field measurement completion interrupt 1: Enables the sync field measurement completion interrupt

16.3.2 LIN Status Register (LINSTR)

LINSTR controls the operation of the hardware LIN and consists of status flags.

Bit	7	6	5	4	3	2	1	0
Bit Name	BCE	—	TOIE	RXDS	TOER	BCER	SBEND	SFEND
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/(W)*	R/(W)*	R/(W)*	R/(W)*

Note: * Only 1 can be written to clear the flag.

Bit	Bit Name	Initial Value	R/W	Description
7	BCE	0	R/W	Bus Conflict Detection Enable 0: Stops the bus conflict detection circuit 1: Starts the bus conflict detection circuit
6	—	0	R/W	Reserved This bit is always read as 0. The write value should always be 0.

Bit	Bit Name	Initial Value	R/W	Description
5	TOIE	0	R/W	Timeout Interrupt Enable 0: Disables the timeout detection interrupt 1: Enables the timeout detection interrupt
4	RXDS	0	R/W	RxD Selector (valid in slave mode) Selects whether masking control is applied to RxD or not. 0: Masking control is not applied to the RxD signal. 1: Masking control is applied to the RxD signal.
3	TOER	0	R/(W)*	Timeout Error [Setting condition] When LINTOCNT underflows [Clearing condition] When writing 0 after reading TOER = 1
2	BCER	0	R/(W)*	Bus Conflict Error [Setting condition] When the TE bit of the SCI is 1 and a bus conflict is detected by the bus conflict detection function while LINE = 1 and BCE = 1 [Clearing condition] When writing 0 after reading BCER = 1
1	SBEND	0	R/(W)*	Sync Break End [Setting condition] When LINTCNT underflows [Clearing condition] When writing 0 after reading SBEND = 1
0	SFEND	0	R/(W)*	Sync Field End [Setting condition] The period for reception of the sync field start bit and bits 0 to 6 has elapsed in slave mode (MSS = 0) [Clearing condition] When writing 0 after reading SFEND = 1

Note: * Only 1 can be written to clear the flag.

16.3.3 LIN Timer Control Register (LINTCR)

LINTCR controls the timer function of the hardware LIN.

Bit	7	6	5	4	3	2	1	0
Bit Name	TOCSTR	TCK1	TCK0	—	—	—	TCSTF	TCSTR
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	TOCSTR	0	R/W	Timeout Counter Start 0: Stops the counter 1: Starts the counter
6	TCK1	0	R/W	Timer Count Source Selection
5	TCK0	0	R/W	Do not switch the counter clock source at the same time as writing 1 to TOCSTR or TCSTR. Do not switch the counter clock source while counting is in progress. To switch the count source, stop counting by the timer. 00: Pϕ/16 01: Pϕ/32 10: Pϕ/64 11: Pϕ/128
4 to 2	—	All 0	R/W	Reserved These bits are always read as 0. The write value should always be 0.
1	TCSTF	0	R	Timer Count Status Flag After 1 has been written to TCSTR to start counting, TCSTF is set to 1. After 0 has been written to TCSTR to stop counting, TCSTF is cleared to 0. Do not make any change until TCSTF reads 1 after the counter has been started or until TCSTF reads 0 after the counter has been stopped. 0: Counting has stopped 1: Counting is in progress

Bit	Bit Name	Initial Value	R/W	Description
0	TCSTR	0	R/W	Timer Count Start Selects whether the LIN timer counter stops or starts counting. When this bit is cleared to 0, LINTCNT is reloaded. 0: Stops counting 1: Starts counting

16.3.4 LIN Timer Counter (LINTCNT)

LINTCNT is a 16-bit readable/writable down counter. LINTCNT must not be accessed in units of eight bits and must be accessed in units of 16 bits.

Bit	15	14	13	12	11	10	9	8
Bit Name								
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

16.3.5 LIN Timeout Counter (LINTOCNT)

LINTOCNT is a 16-bit readable/writable down-counter. LINTOCNT must not be accessed in units of eight bits and must be accessed in units of 16 bits.

Bit	15	14	13	12	11	10	9	8
Bit Name								
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

16.3.6 Setting of LIN Timer Counter and LIN Timeout Counter

Table 16.2 shows the relation that the setting N in LINTCNT and LINTOCNT for the transfer of one bit of data and the bit rate B must satisfy.

Table 16.2 Relation between Setting N in LINTCNT and LINTOCNT for One-Bit Transfer and Bit Rate B

Setting	Error
$N = \frac{P\phi \times 10^6}{n \times B}$	$\text{Error}\ (\%) = \{\frac{P\phi \times 10^6}{B \times n \times N} - 1\} \times 100$

[Legend]
B: Bit rate (bit/s)
N: Required LINTCNT/LINTOCNT value for the transmission or reception of one bit of data
Pϕ: Operating frequency (MHz)
n: The value of n depends on the setting of LINTCR as shown below.

Setting of LINTCR

TCK1	TCK0	n
0	0	16
0	1	32
1	0	64
1	1	128

Since the values in tables 16.3 and 16.4 are absolute results obtained from the formulae in table 16.2, sufficient margins must be allowed in the design of actual systems.

Tables 16.3 to 16.6 list required values for the transmission or reception of one bit in LINTCNT/LINTOCNT when the bit rate is 2400, 9600, 10417, or 19200 bps.

Table 16.3 Settings for One-Bit Transfer in LINTCNT/LINTOCNT When the Bit Rate is 2400 bps

Bit Rate (bit/s)	Operating Frequency Pϕ (MHz)	LINTCNT n	LINTOCNT	Error
2400	8	16	208	0.16 %
		32	104	0.16 %
		64	52	0.16 %
		128	26	0.16 %
	10	16	260	0.16 %
		32	130	0.16 %
		64	65	0.16 %
		128	33	–1.36 %
	12	16	313	–0.16 %
		32	156	0.16 %
		64	78	0.16 %
		128	39	0.16 %
	14	16	365	–0.11 %
		32	182	0.16 %
		64	91	0.16 %
		128	46	–0.93 %
	16	16	417	–0.08 %
		32	208	0.16 %
		64	104	0.16 %
		128	52	0.16 %
	18	16	469	–0.05 %
		32	234	0.16 %
		64	117	0.16 %
		128	59	–0.69 %
2400	20	16	521	–0.03 %
		32	260	0.16 %
		64	130	0.16 %
		128	65	0.16 %
	25	16	651	0.01 %
		32	326	–0.15 %
		64	163	–0.15 %
		128	81	0.47 %
	30	16	781	0.03 %
		32	391	–0.10 %
		64	195	0.16 %
		128	98	–0.35 %
	35	16	911	0.05 %
		32	456	–0.06 %
		64	228	–0.06 %
		128	114	–0.06 %
	40	16	1042	–0.03 %
		32	521	–0.03 %
		64	260	0.16 %
		128	130	0.16 %

Table 16.4 Settings for One-Bit Transfer in LINTCNT/LINTOCNT When the Bit Rate is 9600 bps

Bit Rate (bit/s)	Operating Frequency Pϕ (MHz)	n	LINTCNT LINTOCNT	Error
9600	8	16	52	0.16 %
		32	26	0.16 %
		64	13	0.16 %
		128	7	–6.99 %
	10	16	65	0.16 %
		32	33	–1.36 %
		64	16	1.73 %
		128	8	1.73 %
	12	16	78	0.16 %
		32	39	0.16 %
		64	20	–2.34 %
		128	10	–2.34 %
	14	16	91	0.16 %
		32	46	–0.93 %
		64	23	–0.93 %
		128	11	3.57 %
	16	16	104	0.16 %
		32	52	0.16 %
		64	26	0.16 %
		128	13	0.16 %
	18	16	117	0.16 %
		32	59	–0.69 %
		64	29	1.02 %
		128	15	–2.34 %
9600	20	16	130	0.16 %
		32	65	0.16 %
		64	33	–1.36 %
		128	16	1.73 %
	25	16	163	–0.15 %
		32	81	0.47 %
		64	41	–0.76 %
		128	20	1.73 %
	30	16	195	0.16 %
		32	98	–0.35 %
		64	49	–0.35 %
		128	24	1.73 %
	35	16	228	–0.06 %
		32	114	–0.06 %
		64	57	–0.06 %
		128	28	1.73 %
	40	16	260	0.16 %
		32	130	0.16 %
		64	65	0.16 %
		128	33	–1.36 %

Table 16.5 Settings for One-Bit Transfer in LINTCNT/LINTOCNT When the Bit Rate is 10417 bps

Bit Rate (bit/s)	Operating Frequency Pϕ (MHz)	LINTCNT n	LINTOCNT	Error
10417	8	16	48	0.00 %
		32	24	0.00 %
		64	12	0.00 %
		128	6	0.00 %
	10	16	60	0.00 %
		32	30	0.00 %
		64	15	0.00 %
		128	7	7.14 %
	12	16	72	0.00 %
		32	36	0.00 %
		64	18	0.00 %
		128	9	0.00 %
	14	16	84	0.00 %
		32	42	0.00 %
		64	21	0.00 %
		128	10	5.00 %
	16	16	96	0.00 %
		32	48	0.00%
		64	24	0.00%
		128	12	0.00%
	18	16	108	0.00%
		32	54	0.00%
		64	27	0.00%
		128	13	3.84%
10417	20	16	120	0.00%
		32	60	0.00%
		64	30	0.00%
		128	15	0.00%
	25	16	150	0.00%
		32	75	0.00%
		64	37	1.35%
		128	19	–1.32%
	30	16	180	0.00%
		32	90	0.00%
		64	45	0.00%
		128	22	2.27%
	35	16	210	0.00%
		32	105	0.00%
		64	52	0.96%
		128	26	0.96%
	40	16	240	0.00%
		32	120	0.00%
		64	60	0.00%
		128	30	0.00%

Table 16.6 Settings for One-Bit Transfer in LINTCNT/LINTOCNT When the Bit Rate is 19200 bps

Bit Rate (bit/s)	Operating Frequency Pϕ (MHz)	n	LINTCNT LINTOCNT	Error
19200	8	16	26	0.16 %
		32	13	0.16 %
		64	7	–6.99 %
		128	3	8.51 %
	10	16	33	–1.36 %
		32	16	1.73 %
		64	8	1.73 %
		128	4	1.73 %
	12	16	39	0.16 %
		32	20	–2.34 %
		64	10	–2.34 %
		128	5	–2.34 %
	14	16	46	–0.93 %
		32	23	–0.93 %
		64	11	3.57 %
		128	6	–5.06 %
	16	16	52	0.16 %
		32	26	0.16 %
		64	13	0.16 %
		128	7	–6.99 %
	18	16	59	–0.69 %
		32	29	1.02 %
		64	15	–2.34 %
		128	7	4.63 %
19200	20	16	65	0.16 %
		32	33	–1.36 %
		64	16	1.73 %
		128	8	1.73 %
	25	16	81	0.47 %
		32	41	–0.76 %
		64	20	1.73 %
		128	10	1.73 %
	30	16	98	–0.35 %
		32	49	–0.35 %
		64	24	1.73 %
		128	12	1.73 %
	35	16	114	–0.06 %
		32	57	–0.06 %
		64	28	1.73 %
		128	14	1.73 %
	40	16	130	0.16 %
		32	65	0.16 %
		64	33	–1.36 %
		128	16	1.73 %

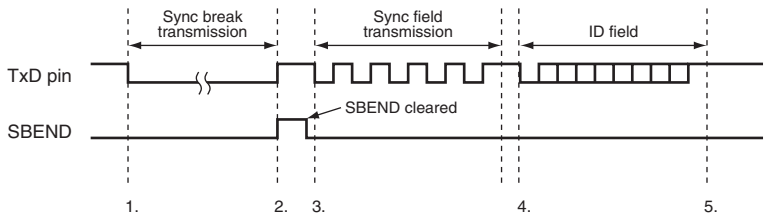
16.4 Operation

16.4.1 Master Mode

Figure 16.2 shows an example of operation when a header field is transmitted in master mode. In addition, figures 16.3 to 16.6 are flowcharts for the transmission of header fields.

In transmission of a header field, the hardware LIN operates as follows.

1. When 1 is written to the TCSTR bit in LINTCR, the low level is output from the TxD pin over the period specified in LINTCNT.
2. When LINTCNT underflows, the output on the TxD pin is inverted and the SBEND flag in LINSTR is set to 1. An interrupt is also generated if the SBIE bit in LINCR is set to 1.
3. H'55 is transmitted via the SCI.
4. After the transmission of H'55 via the SCI is completed, an ID field is transmitted.
5. Once transmission of the ID field has been completed, communication of the response field proceeds.



Note: The following conditions are assumed.
LINE = 1
MSS = 1

Figure 16.2 Example of Operation in the Transmission of a Header Field

(1) Initial Settings in Master Mode

Figure 16.3 shows an example of initial settings in master mode.

Clear the LINE bit in LINCR before initiating LIN communications, and then make initial settings according to the flowchart in figure 16.3. Also be sure to clear the LINE bit to 0 before changing the operating mode.

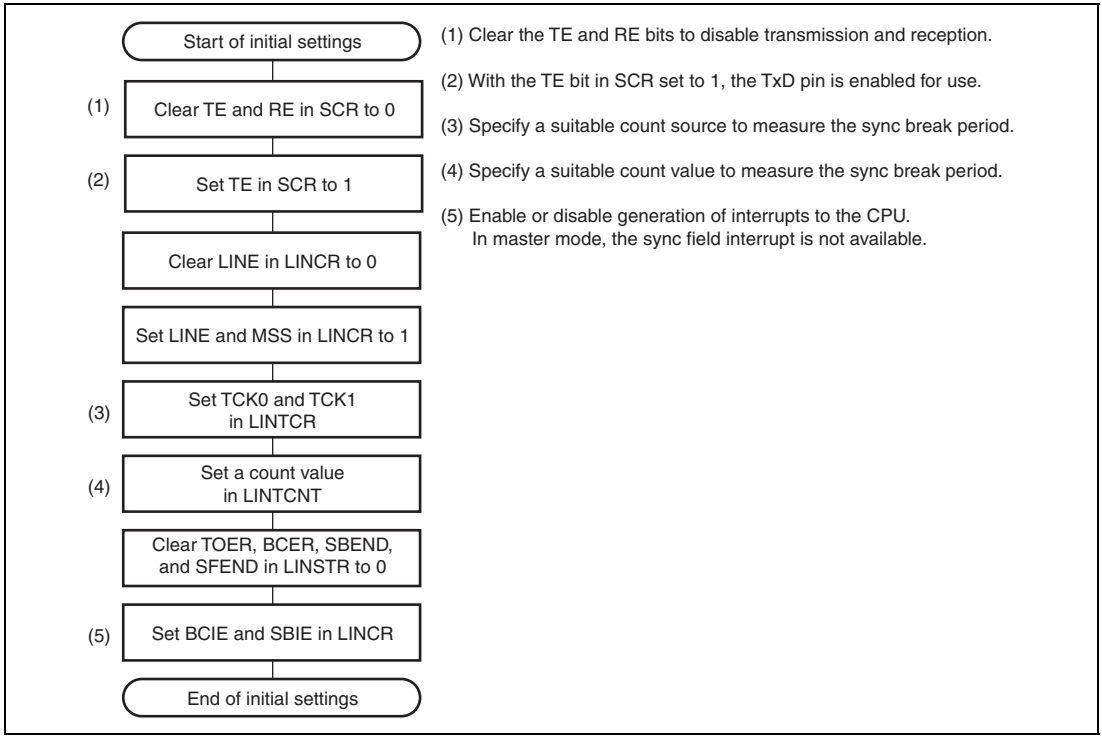


Figure 16.3 Example of Initial Settings in Master Mode

(2) Sync Break Transmission in Master Mode

Figure 16.4 shows an example of settings for sync break transmission in master mode.

In transmission of a sync break, the hardware LIN operates as follows.

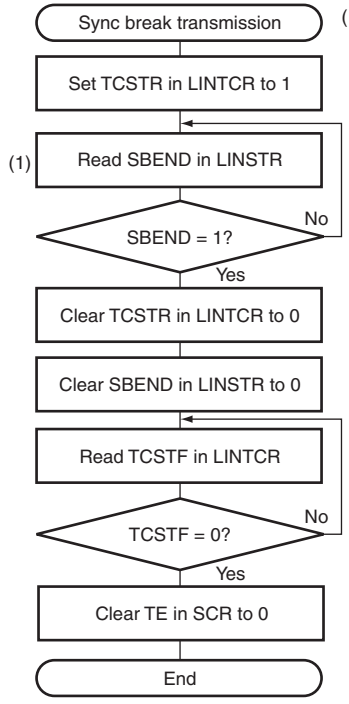
Setting TCSTR in LINTCR to 1 makes the hardware LIN transmit a sync break in synchronization with the timer count source as selected by TCK1 and TCK0 in LINTCR.

Once the transmission of a sync break starts, TCSTF in LINTCR is automatically set to 1 and the low level is output from the TxD pin over the period specified in LINTCNT.

LINTCNT counts down in synchronization with the timer count source as selected by TCK1 and TCK0 in LINTCR. When LINTCNT underflows, the output on the TxD pin is inverted and the SBEND bit in LINSTR is set to 1. If SBIE in LINCR is set to 1, an interrupt is also generated on completion of the sync break.

When LINTCNT underflows, LINTCNT is reloaded with the set value. If TCSTR in LINTCR remains set to 1 after reloading, LINTCNT will continue to count down.

After confirming that SBEND in LINST has been set to 1, successively clear TCSTR in LINTCR and SBEND in LINSTR to 0. When TCSTR in LINTCR is cleared to 0, LINTCNT stops counting down and reloads.



(1) Procedure for completion of sync break transmission:
SBEND = 1 in LINSTR means that sync break transmission has been completed. After the completion of sync break transmission, TCSTR in LINTCR and SBEND in LINSTR should be cleared to 0 successively.

Figure 16.4 Example of Settings for Sync Break Transmission in Master Mode

(3) Sync Field Transmission in Master Mode

Figure 16.5 shows an example of settings for sync field transmission in master mode.

In transmission of a sync field, the hardware LIN operates as follows.

When BCE in LINSTR is set to 1, the hardware LIN starts operation of the bus conflict detection circuit. For details on the bus conflict detection function, see section 16.5, Bus Conflict Detection Function.

In sync field transmission, H'55 is transmitted via the SCI.

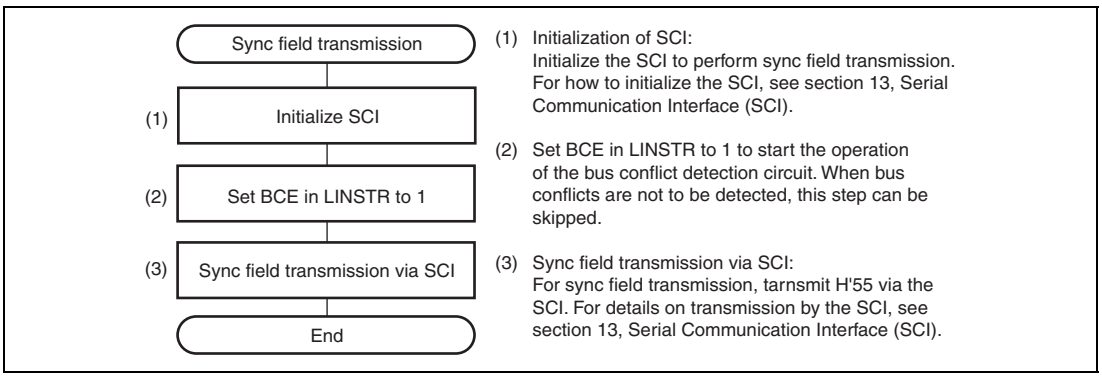


Figure 16.5 Example of Settings for Sync Field Transmission in Master Mode

(4) ID Field Transmission in Master Mode

Figure 16.6 shows an example of settings for ID field transmission in master mode. The SCI is used to transmit the ID field.

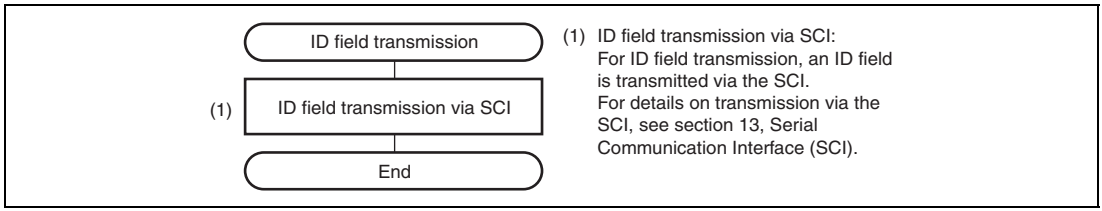


Figure 16.6 Example of Settings for ID Field Transmission in Master Mode

(5) Response Field Communication in Master Mode

Figure 16.7 shows an example of settings for response field communication in master mode. The SCI is used for response field communication.

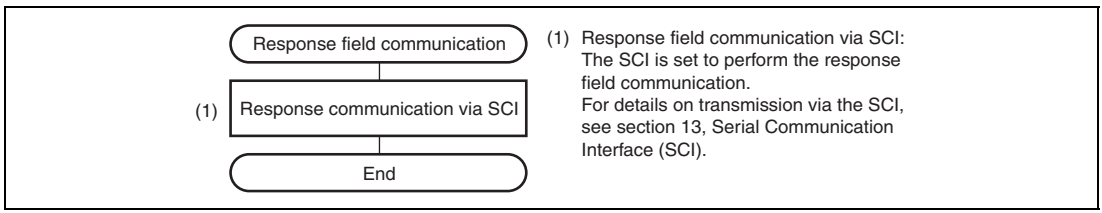


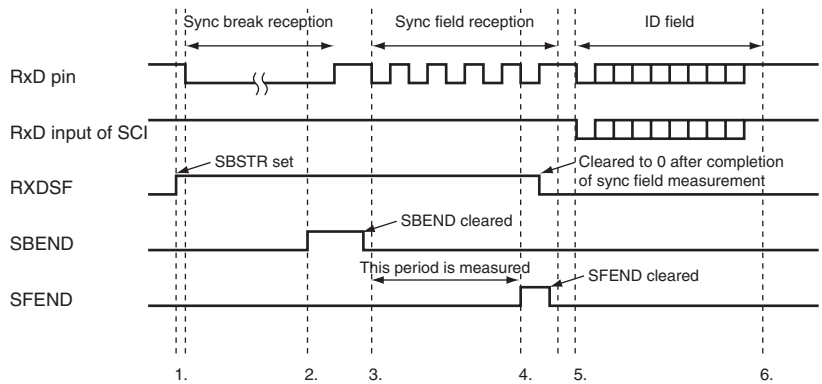
Figure 16.7 Example of Settings for Response Communication in Master Mode

16.4.2 Slave Mode

Figure 16.8 shows an example of operation when a header field is received in slave mode. In addition, figures 16.9 to 16.12 are flowcharts for the reception of header fields.

In reception of a header field, the hardware LIN operates as follows.

1. When 1 is written to the SBSTR bit in LINCR, masking of the RxD input starts.
2. A low level input over the period specified in LINTCNT is detected as a sync break. Then, the SBEND flag in LINSTR is set to 1. An interrupt is also generated on detection of a sync break if the SBIE bit in LINCR is set to 1.
3. The sync field (H'55) is received. At this time, LINTCNT is used to measure the period of the start bit and bits 0 to 6. Also, the RXDMRS bit in LINCR can be used to select whether or not the input of the sync field signal to RxD of the SCI is disabled.
4. After the measurement of the sync field is completed, the SFEND flag in LINSTR is set to 1. An interrupt is also generated on completion of the sync field measurement if the SFIE bit in LINCR is set to 1.
5. On completion of measurement of the sync field, the transfer speed is calculated from the count value of LINTCNT, and the result is set in the SCI. Then, an ID field is received via the SCI.
6. Once reception of the ID field has been completed, communication of the response field starts.



Note: The following conditions are assumed.
LINE = 1
MSS = 0
RXDMRS = 1

Figure 16.8 Example of Operation in the Reception of a Header Field

(1) Initial Settings in Slave Mode

Figure 16.9 shows an example of initial settings in slave mode.

Clear the LINE bit in LINCR before initiating LIN communications, and then make initial settings according to the flowchart in figure 16.9. Also be sure to clear the LINE bit to 0 before changing the operating mode.

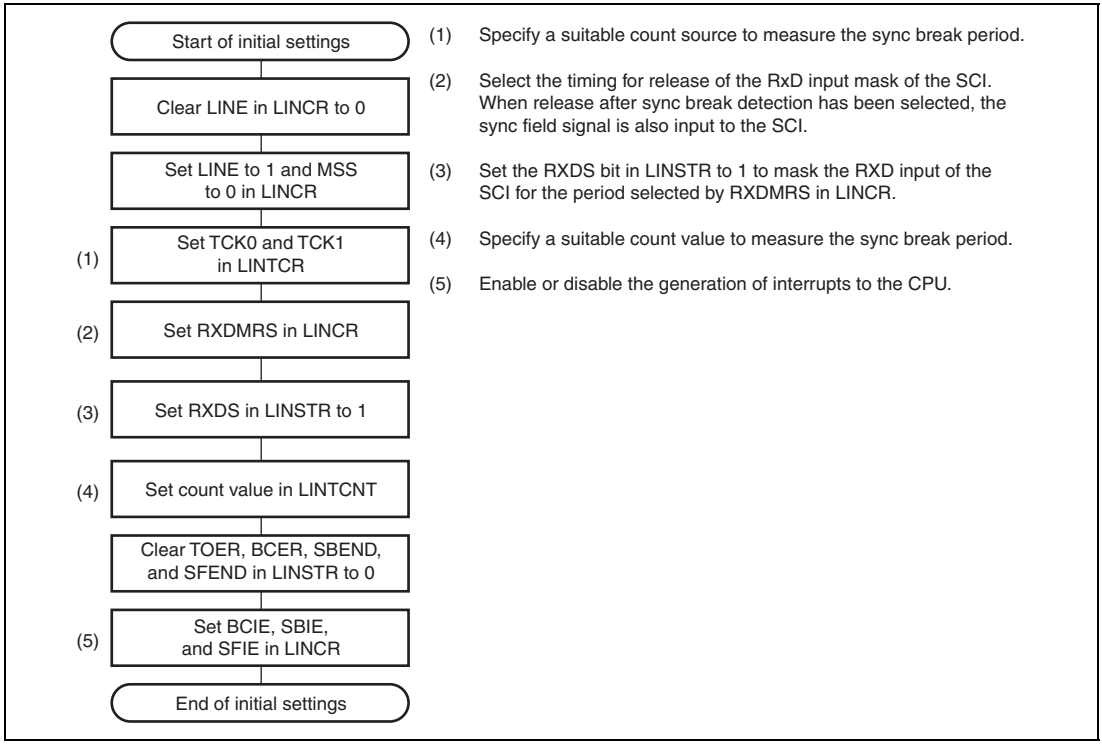


Figure 16.9 Example of Initial Settings in Slave Mode

(2) Sync Break Reception in Slave Mode

Figure 16.10 shows an example of settings for sync break reception in slave mode.

In reception of a sync break, the hardware LIN operates as follows.

Setting TCSTR in LINTCR to 1 places the hardware LIN in the state in which a sync break can be received in synchronization with the timer count source as selected by TCK1 and TCK0 in LINTCR.

Once the down counter is in the operational state, TCSTF in LINTCR is automatically set to 1.

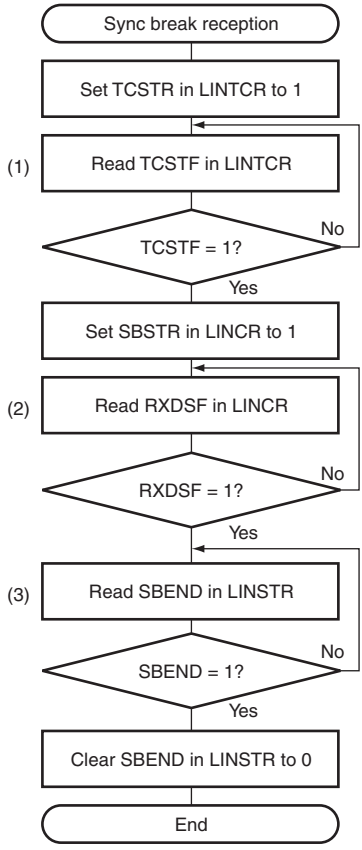
When SBSTR in LINCR is set to 1, the hardware LIN automatically masks the RxD input of the SCI. Do not input the low level to the RxD pin before confirming that RXDSF is 1. A low level input before this will be input to the SCI.

When the low level corresponding to a sync break is input to the RxD pin, LINTCNT starts counting down in synchronization with the timer count source as selected by TCK1 and TCK0 in LINTCR. When LINTCNT underflows, SBEND in LINSTR is set to 1. In addition, if SBIE in LINCR is set to 1, an interrupt is generated on detection of a sync break.

Confirm that SBEND in LINSTR has been set to 1 and then clear SBEND in LINSTR to 0.

When LINTCNT underflows, LINTCNT is reloaded with the set value. If the low level continues to be input to the RxD pin after reloading, down-counting will continue. When a high level is input to the RxD pin on completion of reception of the sync break, LINTCNT is reloaded with the set value and stops counting down. The hardware LIN is placed in the state in which a sync field can be received.

When the period of low-level input to the RxD pin for a sync break is shorter than the period specified in LINTCNT, LINTCNT is reloaded with the initially set value and waits until the next time a low level is input. In this case, SBEND in LINSTR is not set to 1.



(1) Confirm that the count-down timer is operational:
Confirm this by checking that TCSTF in LINTCR is 1.

(2) Confirm that the RxD input of the SCI is masked:
Confirm this by checking that RXDSF in LINCR is 1.
Do not input the low level to the RxD pin before confirming that RXDSF is 1. A low level input before this will be input to the SCI.

(3) Procedure for completion of sync break reception:
Confirm the completion of sync break reception by checking that SBEND in LINSTR is 1. When a sync break is detected, LINTCNT is reloaded with the initially set value.
When the period of low-level input is shorter than the period specified in LINTCNT, LINTCNT is reloaded with the initially set value and counting is suspended until the next low-level input.

Figure 16.10 Example of Settings for Sync Break Reception in Slave Mode

(3) Sync Field Reception in Slave Mode

Figure 16.11 shows an example of settings for sync field reception in slave mode.

In reception of a sync field, the hardware LIN operates as follows.

When receiving a sync field after reception of a sync break is completed, if RXDMRS in LINCR is cleared to 0, data received as the sync field is input to the SCI. In the hardware LIN, LINTCNT counts down for the period of the start bit and bits 0 to 6 of the sync field in synchronization with the timer count source as selected by TCK1 and TCK0 in LINCR.

On completion of reception over the period of the start bit and bits 0 to 6 of the sync field, the hardware LIN automatically sets SFEND in LINSTR to 1. An interrupt is also generated on completion of the sync field measurement if the SFIE bit in LINCR is set to 1. LINTCNT stops counting down on completion of the sync field measurement.

If reception over the period of the start bit and bits 0 to 6 of the sync field takes longer than the period specified in LINTCNT, LINTCNT underflows, and SBEND in LINSTR is set to 1. An interrupt is also generated on detection of a sync break if SBIE in LINCR is set to 1.

The transfer rate is calculated from the count values of LINTCNT before and after sync field measurement, and is set in the SCI. When reading the count value of LINTCNT, read the value before clearing TCSTR in LINTCR to 0. When TCSTR is cleared to 0, LINTCNT is reloaded.

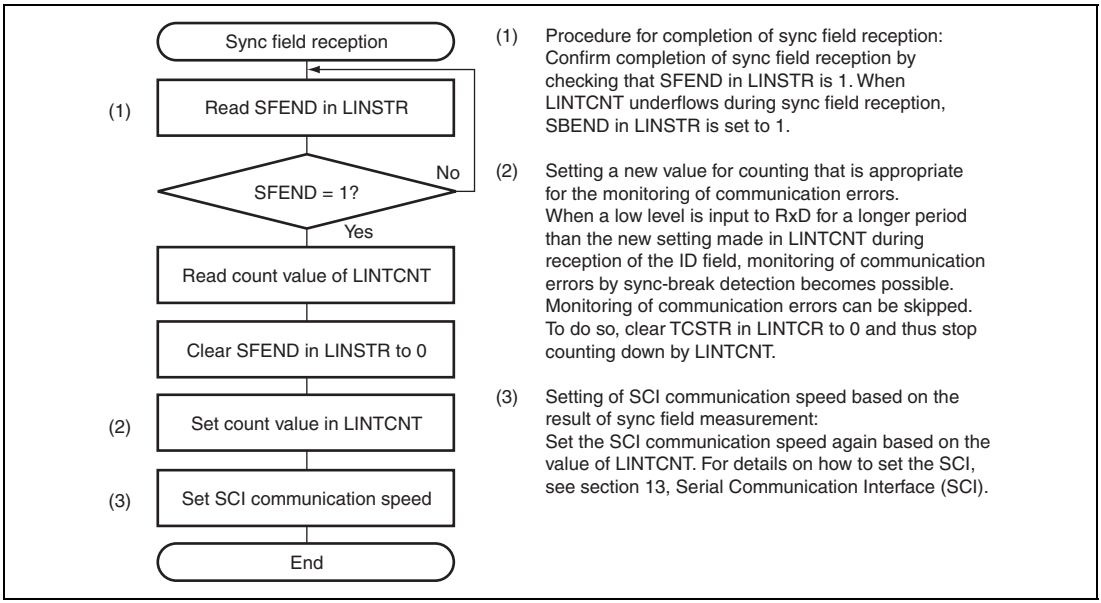


Figure 16.11 Example of Settings for Sync Field Reception in Slave Mode

(4) ID Field Reception in Slave Mode

Figure 16.12 shows an example of settings for ID field reception in slave mode. The SCI is used to receive the ID field.

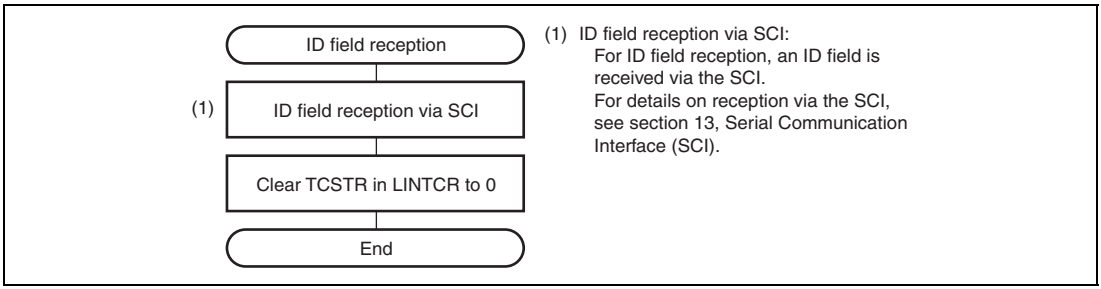


Figure 16.12 Example of Settings for ID Field Reception in Slave Mode

(5) Response Field Communication in Slave Mode

Figure 16.13 shows an example of settings for response field communication in slave mode. The SCI is used for the response field communication.

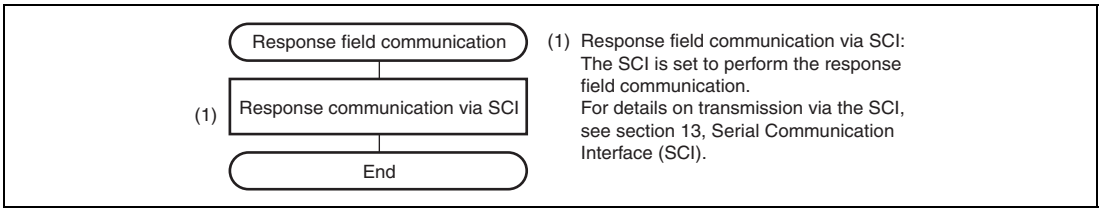


Figure 16.13 Example of Settings for Response Communication in Slave Mode

16.5 Bus Conflict Detection Function

When transmission by the SCI is enabled (the TE bit in SCR is 1), the bus conflict detection function is available.

To use the bus conflict detection function, set the BCE bit in LINSTR to 1 only during transmissions of the sync, ID, and response fields, which take place as data transmission by the SCI. The bus conflict detection function is not available in other operations, that is, sync break transmission, receptions of the sync break, sync field, ID field, and response field. In these operations, clear the BCE bit in LINSTR to 0 to disable the bus conflict detection function.

Figure 16.14 shows an example of operation when a bus conflict is detected.

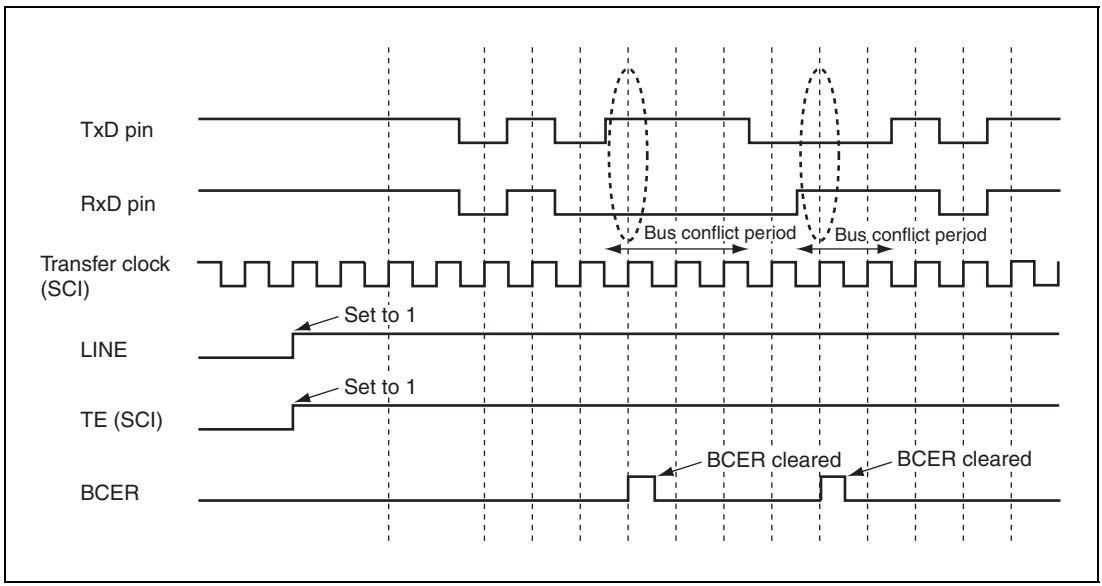


Figure 16.14 Example of Operation When a Bus Conflict is Detected

Bus conflicts are detected on the rising edge of the first SCI transfer clock cycle of the bus conflict period.

When the hardware LIN detects a bus conflict, BCER in LINSTR is set to 1. If the value of BCIE in LINCR is 1 at this time, an interrupt is also generated on detection of a bus conflict.

After confirming that BCER in LINSTR has been set to 1, clear BCER to 0.

16.6 Timeout Detection Function

The timeout detection function refers to determining that an error has occurred when a frame is not completed within the timeout measurement period. In the timeout detection function, software initiates measurement by LINTOCNT when a sync break completion or the sync break detection interrupt is generated. This is the case for both master and slave modes. Measurement by LINTOCNT continues until frame transfer is completed and stops at that point. When LINTOCNT underflows, TOER in LINSTR is set to 1. An interrupt is also generated on detection of a timeout if TOIE in LINSTR has been set to 1.

16.6.1 Timeout Detection in Master Mode

Figure 16.15 shows the frame area over which timeout measurement is performed by the hardware LIN in master mode.

Setting TOCSTR in LINTCR to 1 makes LINTOCNT count down in synchronization with the timer count source as selected by TCK1 and TCK0 in LINTCR. Set the count value required to cover the period from the sync break completion to frame completion in LINTOCNT.

For timeout detection in master mode, set TOCSTR in LINTCR to 1 to start measurement by LINTOCNT after detecting the sync break completion interrupt. On completion of the frame, clear TOCSTR in LINTCR to 0 to stop measurement by LINTOCNT. Note that the LINTOCONT is not reloaded even when the TOCSTR is cleared to 0. When LINTOCNT underflows, TOER in LINSTR is set to 1. When TOIE in LINSTR is set to 1, an interrupt is generated on detection of a timeout. When LINTOCNT underflows, LINTOCNT is reloaded with the set value. Counting down from the reloaded value then continues if TOCSTR is set to 1 after reloading.

Timeout errors from the start of sync break transmission to completion of the frame can be detected in master mode by setting a count value sufficient to cover the period from the start of a sync break to completion of the frame in LINTOCNT and then setting TOCSTR in LINTCR to 1 immediately after the start of the sync break transmission.

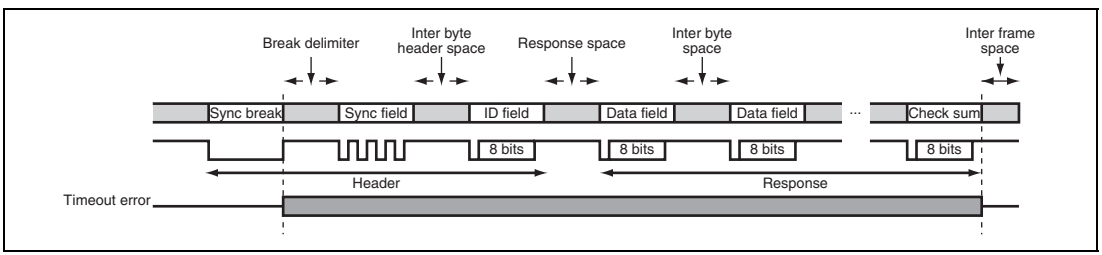


Figure 16.15 Timeout Error Detection Target in Master Mode

16.6.2 Timeout Detection in Slave Mode

Figure 16.16 shows the frame area over which timeout measurement is performed by the hardware LIN in slave mode.

Setting TOCSTR in LINTCR to 1 makes LINTOCNT count down in synchronization with the timer count source as selected by TCK1 and TCK0 in LINTCR. Set the count value required to cover the period from the sync break detection to frame completion in LINTOCNT.

For timeout detection in slave mode, set TOCSTR in LINTCR to 1 to start measurement by LINTOCNT after detecting the sync break detection interrupt. On completion of the frame, clear TOCSTR in LINTCR to 0 to stop measurement by LINTOCNT. Note that LINTOCNT is not reloaded even if TOCSTR is cleared to 0. When LINTOCNT underflows, TOER in LINSTR is set to 1. When TOIE in LINSTR is set to 1, an interrupt is generated on detection of a timeout. When LINTOCNT underflows, LINTOCNT is reloaded with the set value. Counting down from the reloaded value then continues if TOCSTR is set to 1 after reloading.

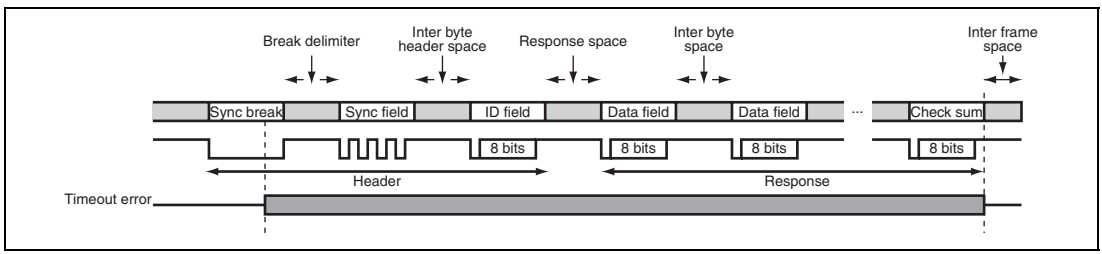


Figure 16.16 Timeout Error Detection Target in Slave Mode

16.7 Wake-Up Detection Method

The RxD pin of the hardware LIN also functions as an IRQn input. Wake-up should be detected as a falling edge on the RxD input pin, with the pin used as an IRQn interrupt. Also, the SCI should be used for transmission of the wake-up.

For details on IRQn interrupts, see section 5, Interrupt Controller.

16.8 Interrupt Request

The hardware LIN generates three types of interrupts in master mode and generates four types of interrupts in slave mode. Table 16.7 gives descriptions of the interrupt sources in master mode. Table 16.8 gives descriptions of the interrupt sources in slave mode. When SBIE, SFIE, BCIE, and TOIE are set to 1, an interrupt is generated when any bit from among SBEND, SFEND, BCER, and TOER is set to 1.

Table 16.7 Interrupt Sources in Master Mode

Interrupt Source	Status Flag	Condition for Interrupt Generation
Sync break completion	SBEND	Underflow of LINTCNT after the period of sync break output has elapsed
Bus conflict detection	BCER	When transmission by the SCI is enabled, values for RxD input and TxD output differ at the time of data latching.
Timeout detection	TOER	Underflow of LINTOCNT

Table 16.8 Interrupt Sources in Slave Mode

Interrupt Source	Status Flag	Condition for Interrupt Generation
Sync break detection	SBEND	Underflow of LINTCNT during measurement of the period of low-level input on RxD
Sync field measurement completion	SFEND	Completion of measurement of the 8 bits of the sync field
Bus conflict detection	BCER	When transmission by the SCI is enabled, values for RxD input and TxD output differ at the time of data latching.
Timeout detection	TOER	Underflow of LINTOCNT

16.9 Usage Notes

16.9.1 Module Stop Mode Setting

Operation of the HWLIN can be enabled or disabled by setting the module stop control register. With the initial value, the HWLIN is stopped. The HWLIN registers become accessible after the HWLIN is released from the module stop mode. For details, see section 23, Power-Down Modes.

16.9.2 Conflict between Writing to the LIN Timer Counter (LINTCNT) and Counting Down

If a counting down condition occurs in the T2 state of a LINTCNT write cycle, writing to LINTCNT takes priority and LINTCNT is not decremented. Figure 16.17 shows this operation.

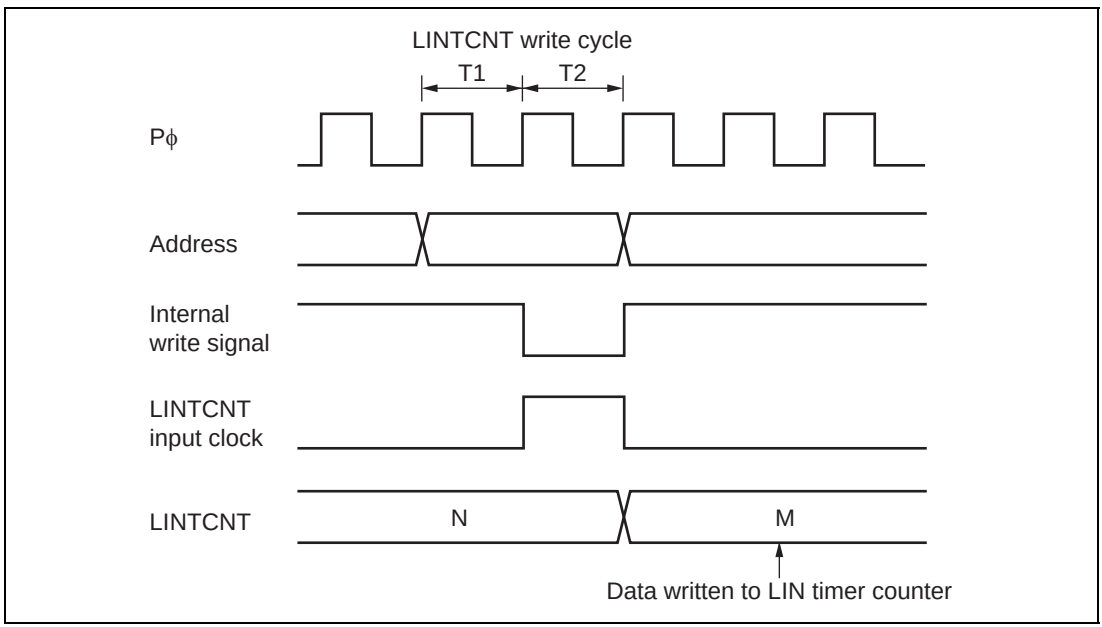


Figure 16.17 Conflict between Writing to LINTCNT and Counting Down

16.9.3 Conflict between Writing to the LIN Timeout Counter (LINTOCNT) and Counting Down

If a counting down condition occurs in the T2 state of a LINTOCNT write cycle, writing to LINTOCNT takes priority and LINTOCNT is not decremented. Figure 16.18 shows this operation.

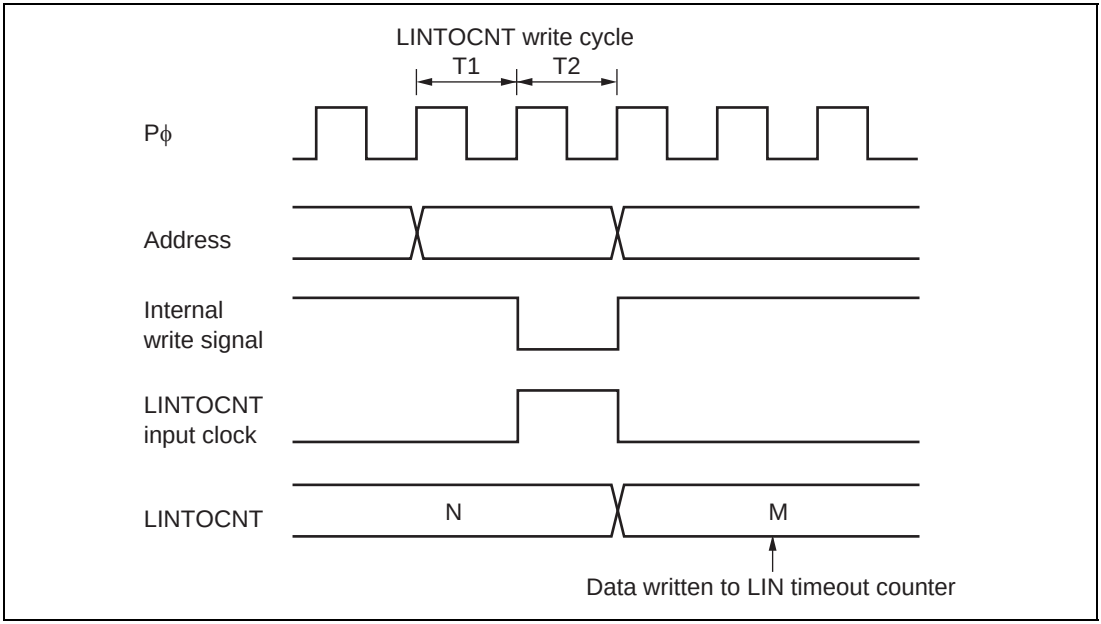


Figure 16.18 Conflict between Writing to LINTOCNT and Counting Down

16.9.4 Conflict between Writing to the LIN Timer Counter (LINTCNT) and Underflow

If LINTCNT is decremented and underflows in the T2 state of a LINTCNT write cycle, writing to LINTCNT takes priority and LINTCNT is not reloaded. The SBEND flag in LINSTR is set. Figure 16.19 shows this operation.

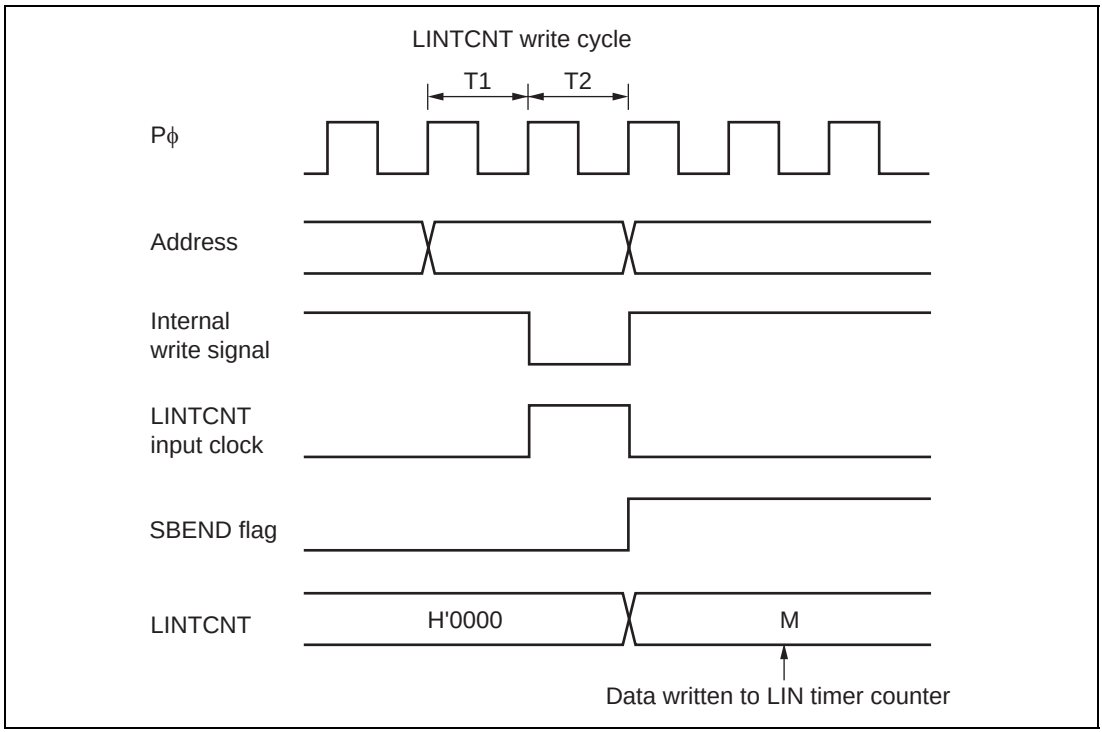


Figure 16.19 Conflict between Writing to LINTCNT and Underflow

16.9.5 Conflict between Writing to the LIN Timeout Counter (LINTOCNT) and Underflow

If LINTOCNT is decremented and underflows in the T2 state of a LINTOCNT write cycle, writing to LINTOCNT takes priority and LINTOCNT is not reloaded. The TOER flag in LINSTR is set. Figure 16.20 shows this operation.

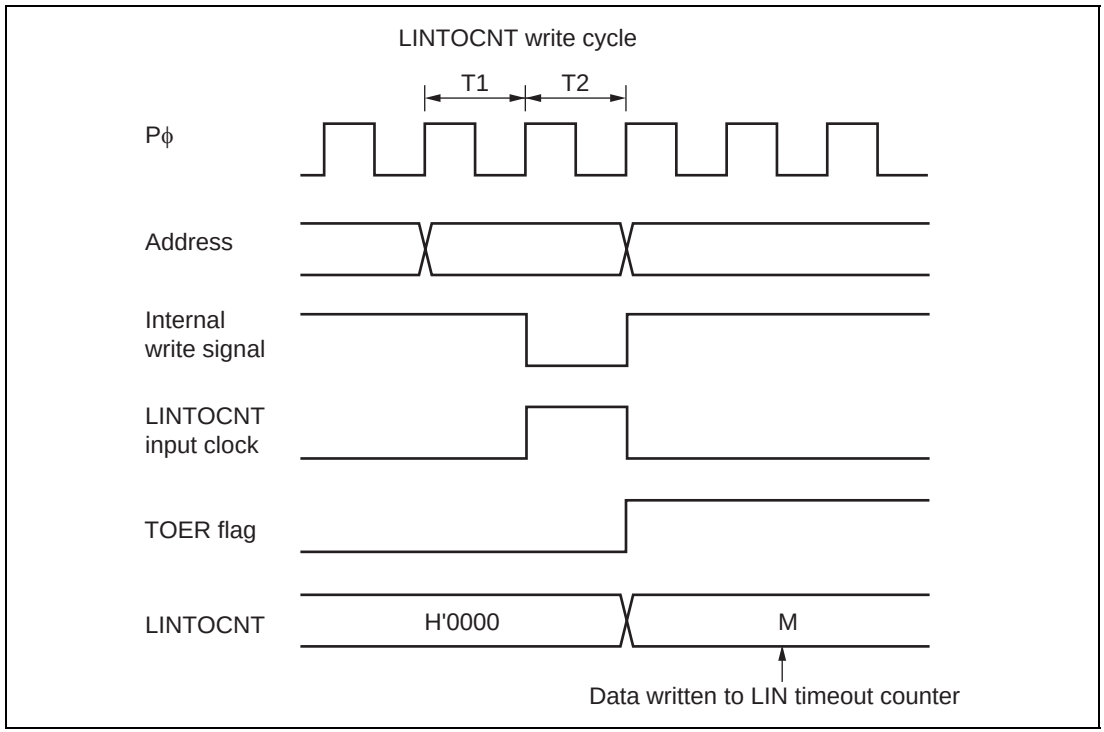


Figure 16.20 Conflict between Writing to LINTOCNT and Underflow

Section 17 CRC Operation Circuit

The cyclic redundancy check (CRC) operation circuit detects errors in data blocks.

17.1 Features

- CRC code generated for any desired data length in 8-bit units
- CRC operation executed on eight bits in parallel
- One of three generating polynomials selectable
- CRC code generation for LSB-first or MSB-first communication selectable

Figure 17.1 shows a block diagram of the CRC operation circuit.

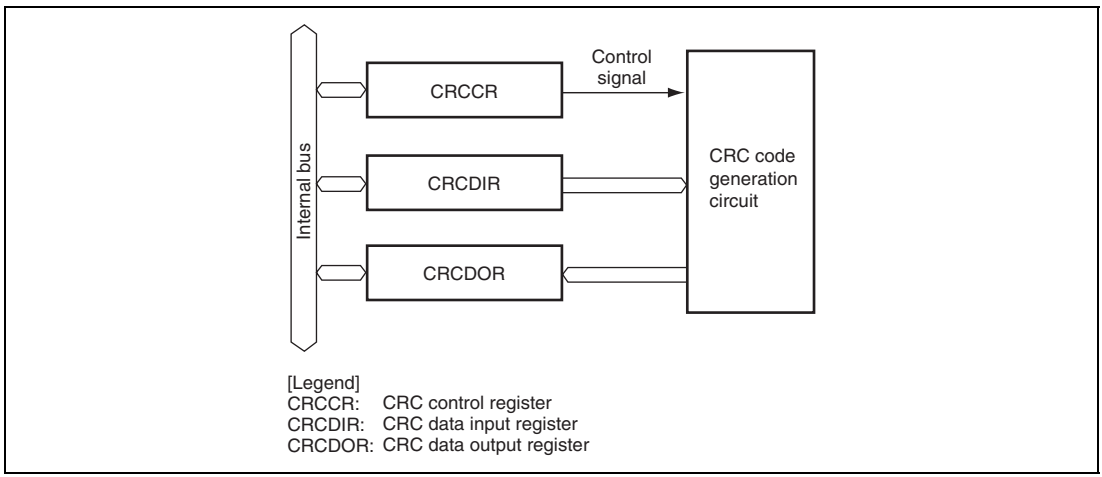


Figure 17.1 Block Diagram of CRC Operation Circuit

17.2 Register Descriptions

The CRC operation circuit has the following registers.

- CRC control register (CRCCR)
- CRC data input register (CRCDIR)
- CRC data output register (CRCDOR)

17.2.1 CRC Control Register (CRCCR)

CRCCR initializes the CRC operation circuit, switches the operation mode, and selects the generating polynomial.

Bit	7	6	5	4	3	2	1	0
Bit Name	DORCLR	—	—	—	—	LMS	G1	G0
Initial Value	0	0	0	0	0	0	0	0
R/W	W	R	R	R	R	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	DORCLR	0	W	CRCDOR Clear Setting this bit to 1 clears CRCDOR to H'0000.
6 to 3	—	All 0	R	Reserved The initial value should not be changed.
2	LMS	0	R/W	CRC Operation Switch Selects CRC code generation for LSB-first or MSB-first communication. 0: Performs CRC operation for LSB-first communication. The lower byte (bits 7 to 0) is first transmitted when CRCDOR contents (CRC code) are divided into two bytes to be transmitted in two parts. 1: Performs CRC operation for MSB-first communication. The upper byte (bits 15 to 8) is first transmitted when CRCDOR contents (CRC code) are divided into two bytes to be transmitted in two parts.
1	G1	0	R/W	CRC Generating Polynomial Select:
0	G0	0	R/W	Selects the polynomial. 00: Reserved 01: $X^8 + X^2 + X + 1$ 10: $X^{16} + X^{15} + X^2 + 1$ 11: $X^{16} + X^{12} + X^5 + 1$

17.2.2 CRC Data Input Register (CRCDIR)

CRCDIR is an 8-bit readable/writable register, to which the bytes to be CRC-operated are written. The result is obtained in CRCDOR.

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

17.2.3 CRC Data Output Register (CRCDOR)

CRCDOR is a 16-bit readable/writable register that contains the result of CRC operation when the bytes to be CRC-operated are written to CRCDIR after CRCDOR is cleared. When the CRC operation result is additionally written to the bytes to which CRC operation is to be performed, the CRC operation result will be H'0000 if the data contains no CRC error. When bits 1 and 0 in CRCCR (G1 and G0 bits) are set to 0 and 1, respectively, the lower byte of this register contains the result.

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name								
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

17.3 CRC Operation Circuit Operation

The CRC operation circuit generates a CRC code for LSB-first/MSB-first communications. An example in which a CRC code for hexadecimal data H'F0 is generated using the $X^{16} + X^{12} + X^5 + 1$ polynomial with the G1 and G0 bits in CRCCR set to B'11 is shown below.

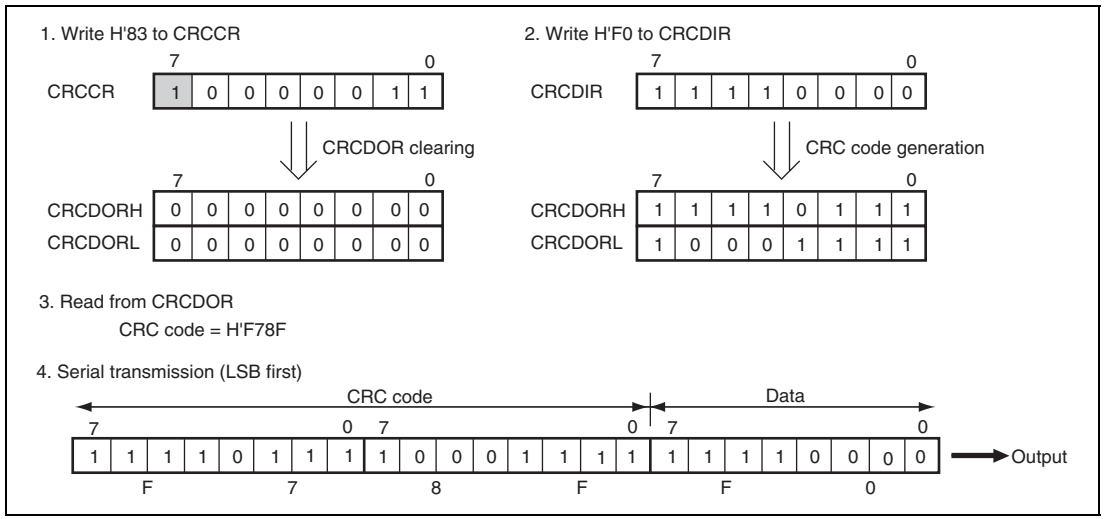


Figure 17.2 LSB-First Data Transmission

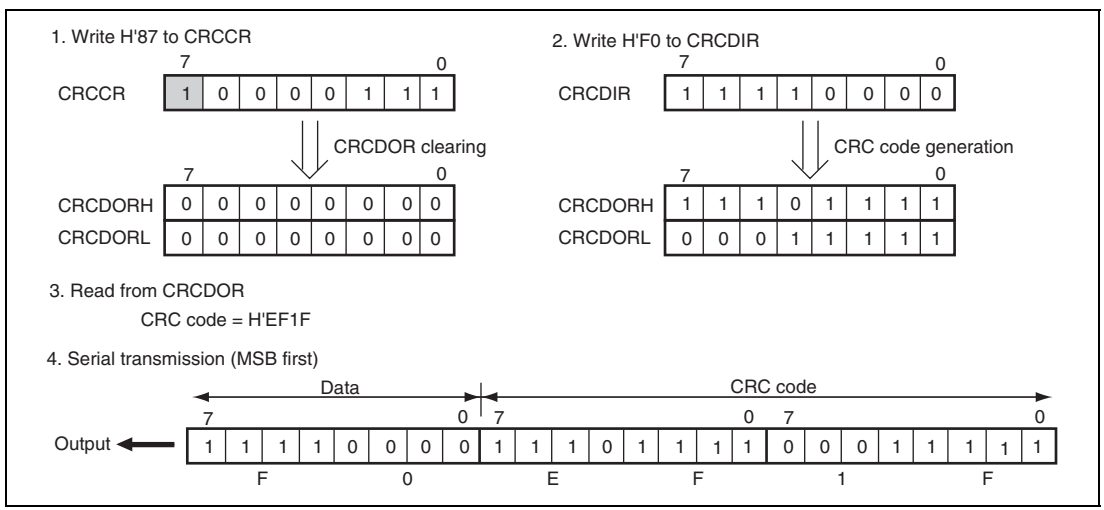


Figure 17.3 MSB-First Data Transmission

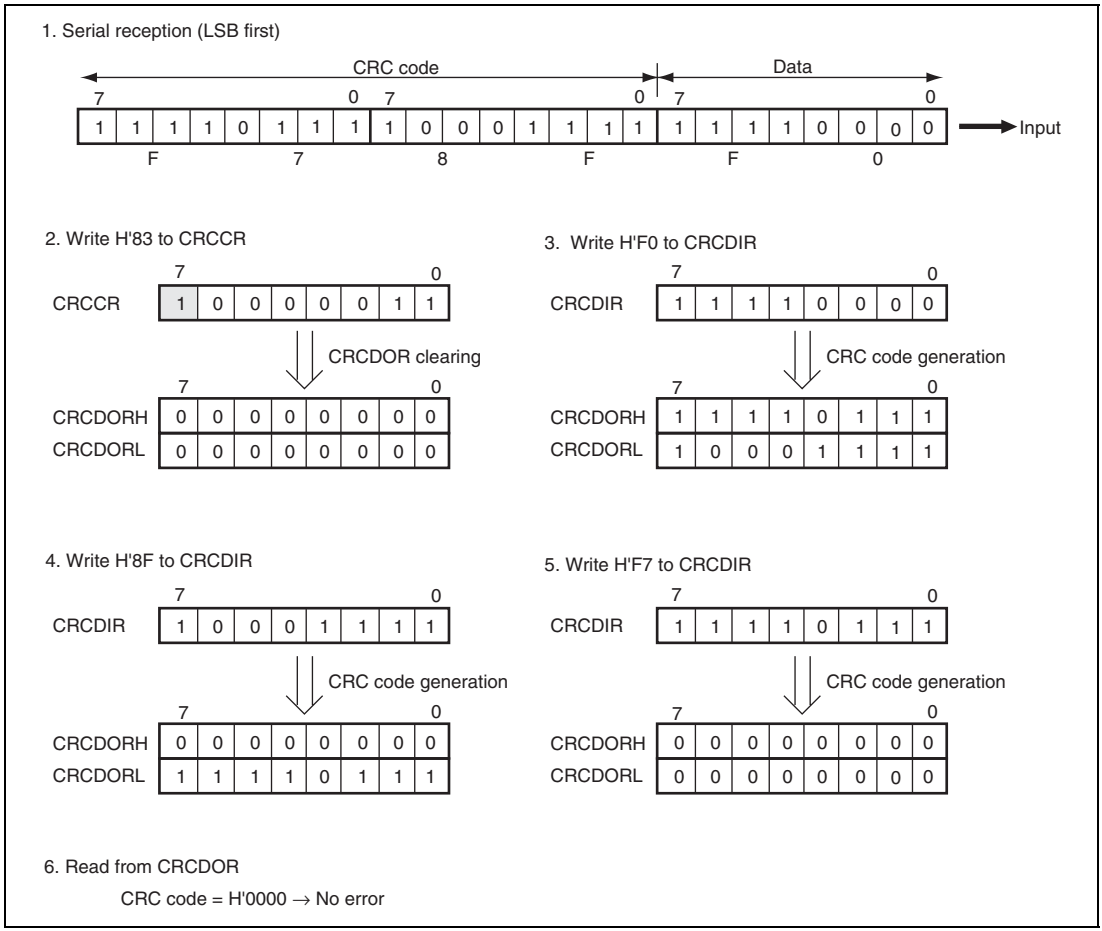


Figure 17.4 LSB-First Data Reception

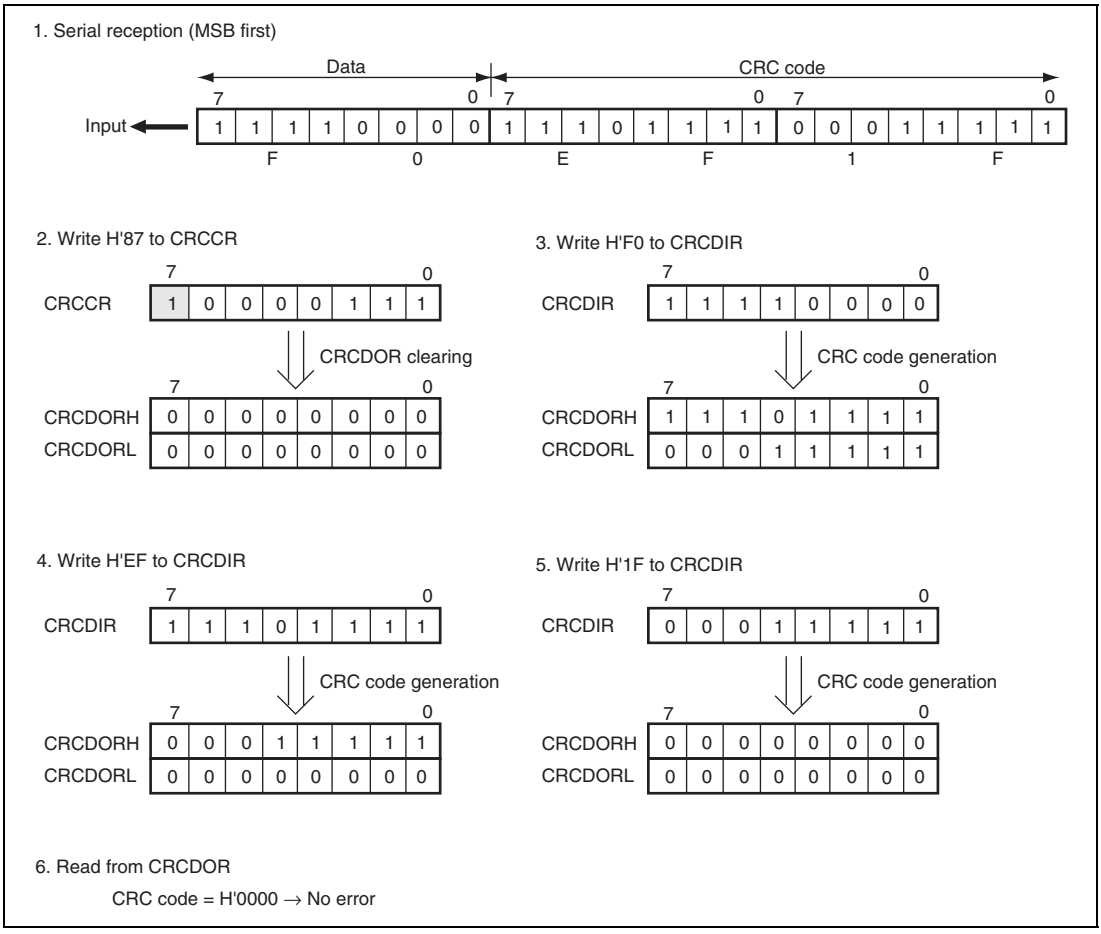


Figure 17.5 MSB-First Data Reception

17.4 Note on CRC Operation Circuit

Note that the sequence to transmit the CRC code differs between LSB-first transmission and MSB-first transmission.

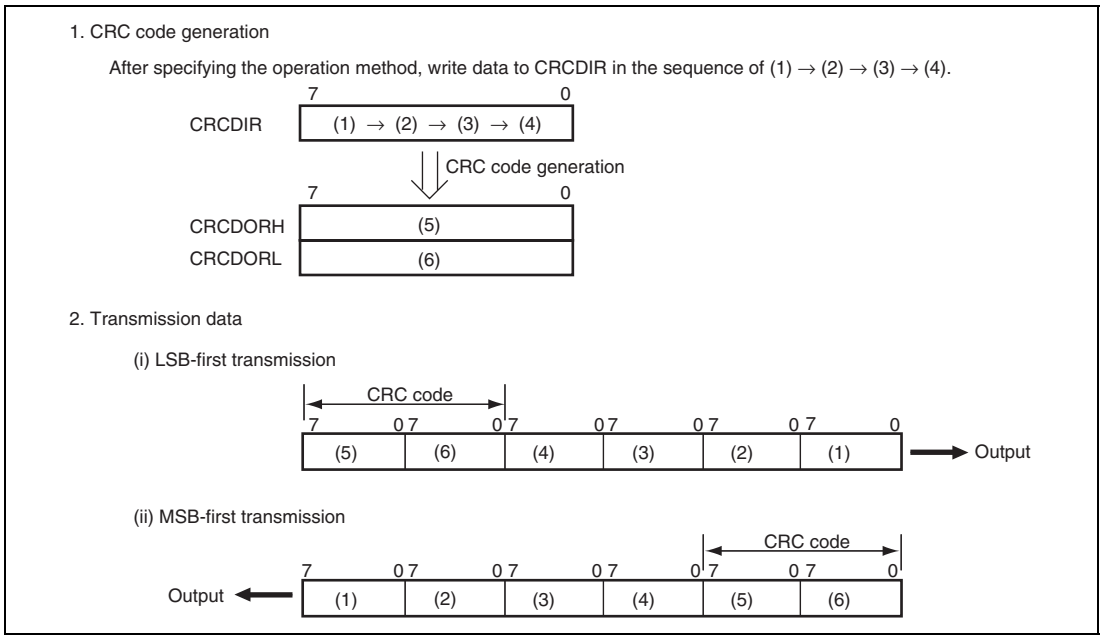


Figure 17.6 LSB-First and MSB-First Transmit Data

Section 18 A/D Converter

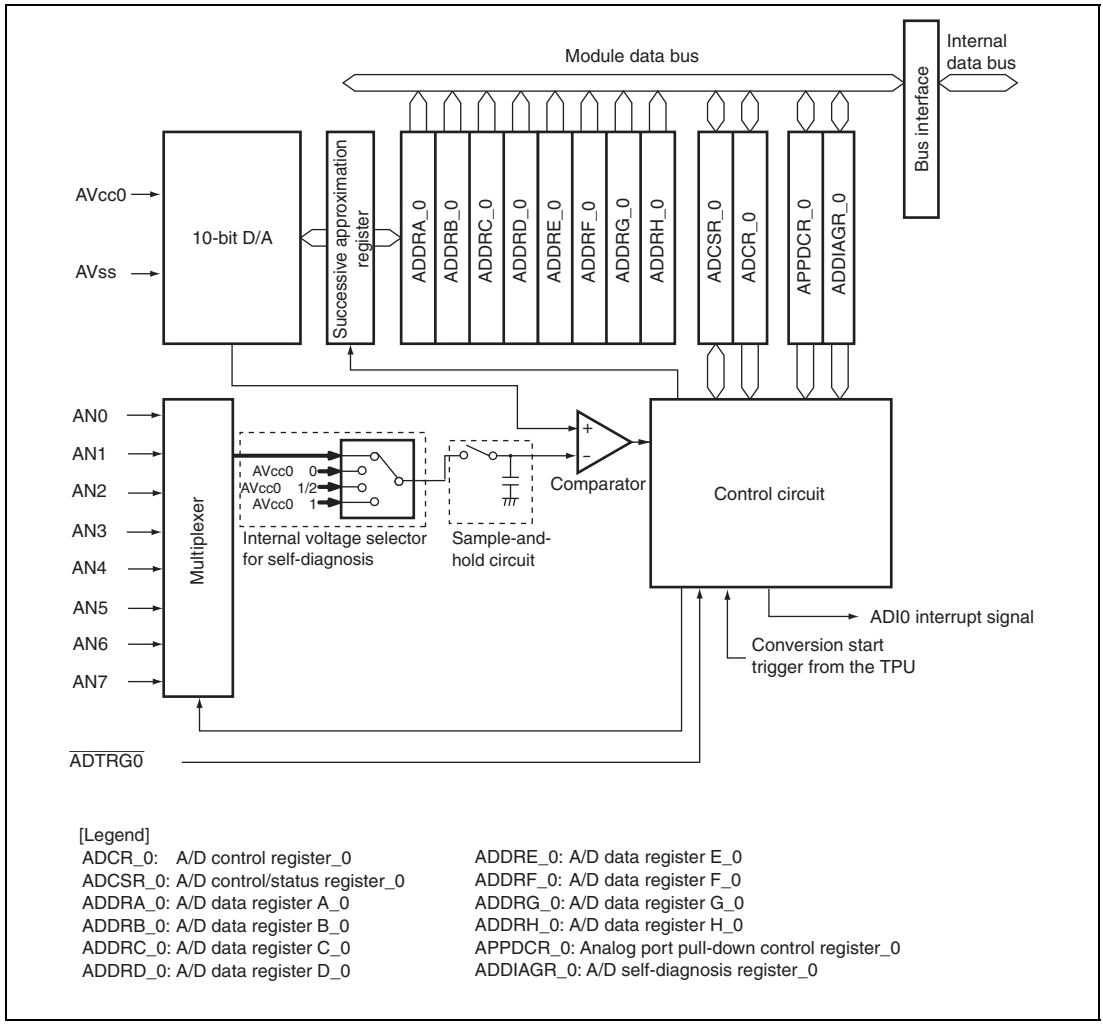
This LSI includes two units (unit 0 and unit 1) of a successive approximation type 10-bit A/D converters that allow up to 8 analog input channels to be selected.

Figures 18.1 and 18.2 are block diagrams for unit 0 and unit 1, respectively.

This section describes unit 0, which has the same functions as the other unit.

18.1 Features

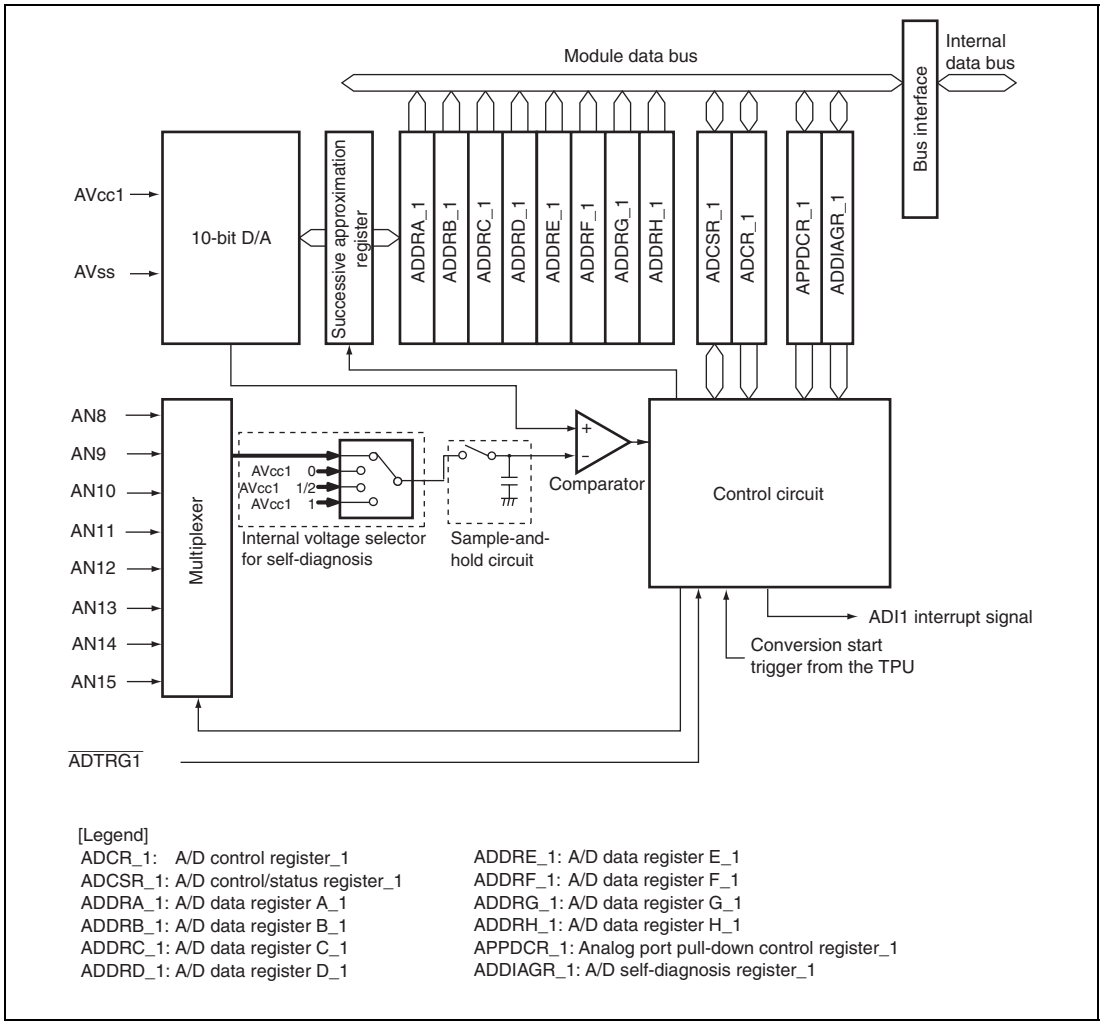
- 10-bit resolution
- 8 input channels (total of 16 channels for two units)
- Conversion cycle: 40 cycles (A/D conversion clock)
- Two kinds of operating modes
  - Single mode: Single-channel A/D conversion
  - Scan mode: Continuous A/D conversion on 1 to 4 channels, or 1 to 8 channels
- Clock for A/D conversion can be selected for each unit (Aϕ, Aϕ/2, Aϕ/4, or Aϕ/8)
- Eight data registers (16 registers for two units)
  A/D conversion results are held in a 16-bit data register for each channel
- Sample and hold function
- Three ways of starting A/D conversion
  Conversion can be started by software, a conversion start trigger from the 16-bit timer pulse unit (TPU), or an external trigger signal.
- Interrupt source
  A/D conversion end interrupt (ADI) request can be generated.
- Module stop mode can be set
- Checking of analog input signal disconnection by the analog port pull-down function
- Self-diagnosis of the A/D converter



[Legend]

ADCR_0: A/D control register_0
ADCSR_0: A/D control/status register_0
ADDRA_0: A/D data register A_0
ADDRB_0: A/D data register B_0
ADDRC_0: A/D data register C_0
ADDRD_0: A/D data register D_0
ADDRE_0: A/D data register E_0
ADDRF_0: A/D data register F_0
ADDRG_0: A/D data register G_0
ADDRH_0: A/D data register H_0
APPDCR_0: Analog port pull-down control register_0
ADDIAGR_0: A/D self-diagnosis register_0

Figure 18.1 Block Diagram of A/D Converter (Unit 0/AD_0)



[Legend]

ADCR_1: A/D control register_1
ADCSR_1: A/D control/status register_1
ADDRA_1: A/D data register A_1
ADDRB_1: A/D data register B_1
ADDRC_1: A/D data register C_1
ADDRD_1: A/D data register D_1
ADDRE_1: A/D data register E_1
ADDRF_1: A/D data register F_1
ADDRG_1: A/D data register G_1
ADDRH_1: A/D data register H_1
APPDCR_1: Analog port pull-down control register_1
ADDIAGR_1: A/D self-diagnosis register_1

Figure 18.2 Block Diagram of A/D Converter (Unit 1/AD_1)

18.2 Input/Output Pins

Table 18.1 shows the pin configuration of the A/D converter.

Table 18.1 Pin Configuration

Unit	Abbr.	Pin Name	Symbol	I/O	Function
0	AD_0	Analog input pin 0	AN0	Input	Analog input
		Analog input pin 1	AN1	Input	
		Analog input pin 2	AN2	Input	
		Analog input pin 3	AN3	Input	
		Analog input pin 4	AN4	Input	
		Analog input pin 5	AN5	Input	
		Analog input pin 6	AN6	Input	
		Analog input pin 7	AN7	Input	
		A/D external trigger input pin 0	$\overline{\text{ADTRG0}}$	Input	External trigger input for starting A/D conversion
		Analog power supply pin 0	$AV_{CC}0$	Input	Analog block power supply and reference voltage
1	AD_1	Analog input pin 8	AN8	Input	Analog input
		Analog input pin 9	AN9	Input	
		Analog input pin 10	AN10	Input	
		Analog input pin 11	AN11	Input	
		Analog input pin 12	AN12	Input	
		Analog input pin 13	AN13	Input	
		Analog input pin 14	AN14	Input	
		Analog input pin 15	AN15	Input	
		A/D external trigger input pin 1	$\overline{\text{ADTRG1}}$	Input	External trigger input for starting A/D conversion
		Analog power supply pin 1	$AV_{CC}1$	Input	Analog block power supply and reference voltage
Common		Analog ground pin	AV_{SS}	Input	Analog block ground

18.3 Register Descriptions

The A/D converter has the following registers.

The registers for unit 0 (A/D_0) and unit 1 (A/D_1) have the same functions. In this descriptions, AN8 to AN15 correspond to AN0 to AN7.

- Unit 0 (A/D_0)
  - A/D data register A_0 (ADDRA_0)
  - A/D data register B_0 (ADDRB_0)
  - A/D data register C_0 (ADDRC_0)
  - A/D data register D_0 (ADDRD_0)
  - A/D data register E_0 (ADDRE_0)
  - A/D data register F_0 (ADDRF_0)
  - A/D data register G_0 (ADDRG_0)
  - A/D data register H_0 (ADDRH_0)
  - A/D control/status register_0 (ADCSR_0)
  - A/D control register_0 (ADCR_0)
  - Analog port pull-down control register_0 (APPDCR_0)
  - A/D self-diagnosis register_0 (ADDIGR_0)
- Unit 1 (A/D_1)
  - A/D data register A_1 (ADDRA_1)
  - A/D data register B_1 (ADDRB_1)
  - A/D data register C_1 (ADDRC_1)
  - A/D data register D_1 (ADDRD_1)
  - A/D data register E_1 (ADDRE_1)
  - A/D data register F_1 (ADDRF_1)
  - A/D data register G_1 (ADDRG_1)
  - A/D data register H_1 (ADDRH_1)
  - A/D control/status register_1 (ADCSR_1)
  - A/D control register_1 (ADCR_1)
  - Analog port pull-down control register_1 (APPDCR_1)
  - A/D self-diagnosis register_1 (ADDIGR_1)

18.3.1 A/D Data Registers A to H (ADDRA to ADDRH)

There are eight 16-bit read-only ADDR registers, ADDRA to ADDRH, used to store the results of A/D conversion. The ADDR registers, which store a conversion result for each analog input channel, are shown in table 18.2.

The converted 10-bit data is stored in bits 15 to 6. The lower six bits are always read as 0.

The data bus between the CPU and the A/D converter has a 16-bit width. The data can be read directly from the CPU. ADDR must not be accessed in units of eight bits and must be accessed in units of 16 bits.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Name											—	—	—	—	—	—
Initial Value:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 18.2 Analog Input Channels and Corresponding ADDR Registers

Analog Input Channel	A/D Data Register Which Stores Conversion Result
AN0	ADDRA
AN1	ADDRB
AN2	ADDRC
AN3	ADDRD
AN4	ADDRE
AN5	ADDRF
AN6	ADDRG
AN7	ADDRH

18.3.2 A/D Control/Status Register (ADCSR)

ADCSR controls A/D conversion operations.

Bit	7	6	5	4	3	2	1	0
Bit Name	ADF	ADIE	ADST	—	CH3	CH2	CH1	CH0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R/W	R/W	R	R/W	R/W	R/W	R/W

Note: * Only 0 can be written to this bit to clear the flag.

Bit	Bit Name	Initial Value	R/W	Description
7	ADF	0	R/(W)*	A/D End Flag A status flag that indicates the end of A/D conversion. [Setting conditions] • When A/D conversion ends in single mode • When A/D conversion ends on all specified channels in scan mode [Clearing conditions] • When 0 is written after reading ADF = 1 (When the CPU is used to clear this flag by writing 0 while the corresponding interrupt is enabled, be sure to read the flag after writing 0 to it.) • When the DMAC or DTC is activated by an ADI interrupt and ADDR is read
6	ADIE	0	R/W	A/D Interrupt Enable When this bit is set to 1, an ADI interrupt by ADF is enabled.
5	ADST	0	R/W	A/D Start Clearing this bit to 0 stops A/D conversion, and the A/D converter enters wait state. Setting this bit to 1 starts A/D conversion. In single mode, this bit is cleared to 0 automatically when A/D conversion on the specified channel ends. In scan mode, A/D conversion continues sequentially on the specified channels until this bit is cleared to 0 by software, a reset, or a transition to hardware standby mode*. Note: * The hardware standby mode is not available in this LSI.

Bit	Bit Name	Initial Value	R/W	Description
4	—	0	R	Reserved This bit is always read as 0 and cannot be modified.
3	CH3	0	R/W	Channel Select 3 to 0 These bits select analog input together with bits SCANE and SCANS in ADCR. • When SCANE = 0 and SCANS = X 0000: AN0 0001: AN1 0010: AN2 0011: AN3 0100: AN4 0101: AN5 0110: AN6 0111: AN7 1×××: Setting prohibited • When SCANE = 1 and SCANS = 0 0000: AN0 0001: AN0 and AN1 0010: AN0 to AN2 0011: AN0 to AN3 0100: AN4 0101: AN4 and AN5 0110: AN4 to AN6 0111: AN4 to AN7 1×××: Setting prohibited • When SCANE = 1 and SCANS = 1 0000: AN0 0001: AN0 and AN1 0010: AN0 to AN2 0011: AN0 to AN3 0100: AN0 to AN4 0101: AN0 to AN5 0110: AN0 to AN6 0111: AN0 to AN7 1×××: Setting prohibited
2	CH2	0	R/W	
1	CH1	0	R/W	
0	CH0	0	R/W	

[Legend]

X: Don't care

Note: * Only 0 can be written to clear the flag.

18.3.3 A/D Control Register (ADCR)

ADCR enables A/D conversion to be started by an external trigger input.

Bit	7	6	5	4	3	2	1	0
Bit Name	TRGS1	TRGS0	SCANE	SCANS	CKS1	CKS0	ADSTCLR	EXTRGS
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	TRGS1	0	R/W	Timer Trigger Select 1 and 0, Expansion Trigger Select
6	TRGS0	0	R/W	These bits select enabling or disabling of the start of A/D conversion by a trigger signal.
0	EXTRGS	0	R/W	000: A/D conversion start by an external trigger is disabled 001: A/D conversion start by the conversion trigger from TPU1 is enabled 010: A/D conversion start by the conversion trigger from TPU0 is enabled 011: Setting prohibited 100: Setting prohibited 101: Setting prohibited 110: A/D conversion start by the $\overline{\text{ADTRG0}}$ pin is enabled 111: Setting prohibited
5	SCANE	0	R/W	Scan Mode
4	SCANS	0	R/W	These bits select the A/D conversion operating mode. 0X: Single mode 10: Scan mode. A/D conversion is performed continuously for channels 1 to 4. 11: Scan mode. A/D conversion is performed continuously for channels 1 to 8.

Bit	Bit Name	Initial Value	R/W	Description
3	CKS1	0	R/W	Clock Select 1 and 0
2	CKS0	0	R/W	These bits select the clock (ADCLK) used for A/D conversion. Set these bits when the ADST bit in ADCSR is 0. Then set the conversion mode. Additionally, do not select Aϕ/8 when the ACK2 to ACK0 bits in SCKCR1 are set to 011. 00: Aϕ 01: Aϕ/2 10: Aϕ/4 11: Aϕ/8
1	ADSTCLR	0	R/W	A/D Start Clear Sets automatic clearing of the ADST bit in scan mode. 0: Automatic clearing of the ADST bit in scan mode is disabled. 1: The ADST bit is automatically cleared when A/D conversion of all selected channels is completed in scan mode.

[Legend]

X: Don't care

Note: * When the $\overline{\text{ADTRG}}$ signal is used to activate an A/D conversion, clear a corresponding DDR bit to 0 and set a corresponding ICR bit to 1. For details, see section 9, I/O Ports.

18.3.4 Analog Port Pull-Down Control Register (APPDCR)

APPDCR controls the pull-down MOS of the analog input pin.

Bit	7	6	5	4	3	2	1	0
Bit Name	AN7PD	AN6PD	AN5PD	AN4PD	AN3PD	AN2PD	AN1PD	AN0PD
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	AN7PD	0	R/W	These bits control the pull-down MOS of the analog input pin. 0: The pull-down MOS is off. 1: The pull-down MOS is on.
6	AN6PD	0	R/W	
5	AN5PD	0	R/W	
4	AN4PD	0	R/W	
3	AN3PD	0	R/W	
2	AN2PD	0	R/W	
1	AN1PD	0	R/W	
0	AN0PD	0	R/W	

18.3.5 A/D Self-Diagnosis Register (ADDIAGR)

ADDIAGR is used to detect faults in the A/D converter. A value for internally generated voltage to be A/D-converted can be selected.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	DIAG1	DIAG0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 2	—	0	R/W	Reserved These bits are always read as 0. The write value should always be 0.
1	DIAG1	0	R/W	A/D Self-Diagnosis
0	DIAG0	0		00: Self-diagnosis is off 01: A/D conversion of AVcc0 × 0 voltage value is enabled 10: A/D conversion of AVcc0 × 1/2 voltage value is enabled 11: A/D conversion of AVcc0 × 1 voltage value is enabled

18.4 Operation

The A/D converter operates by successive approximation with 10-bit resolution. It has two operating modes: single mode and scan mode. When changing the operating mode or analog input channel, to prevent incorrect operation, first clear the ADST bit in ADCSR to 0 to halt A/D conversion. The ADST bit can be set to 1 at the same time as the operating mode or analog input channel is changed.

18.4.1 Single Mode

In single mode, A/D conversion is to be performed only once on the analog input of the specified single channel.

1. A/D conversion for the selected channel is started when the ADST bit in ADCSR is set to 1 by software, TPU, or an external trigger input.
2. When A/D conversion is completed, the A/D conversion result is transferred to the corresponding A/D data register of the channel.
3. When A/D conversion is completed, the ADF bit in ADCSR is set to 1. If the ADIE bit is set to 1 at this time, an ADI interrupt request is generated.
4. The ADST bit remains set to 1 during A/D conversion, and is automatically cleared to 0 when A/D conversion ends. The A/D converter enters wait state. If the ADST bit is cleared to 0 during A/D conversion, A/D conversion stops and the A/D converter enters wait state.

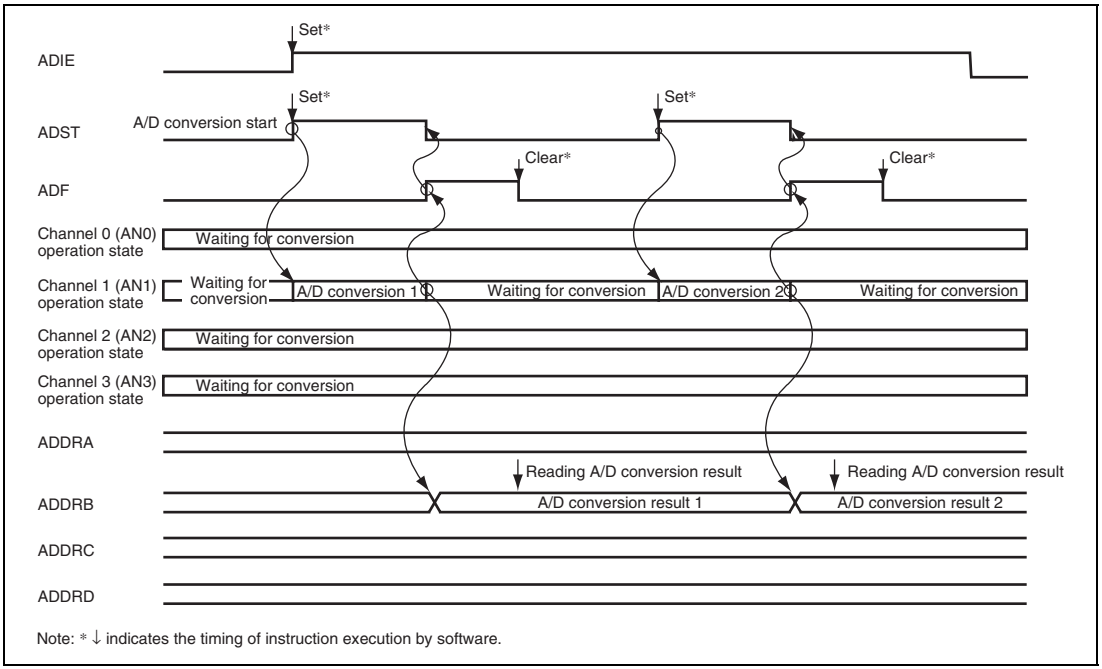


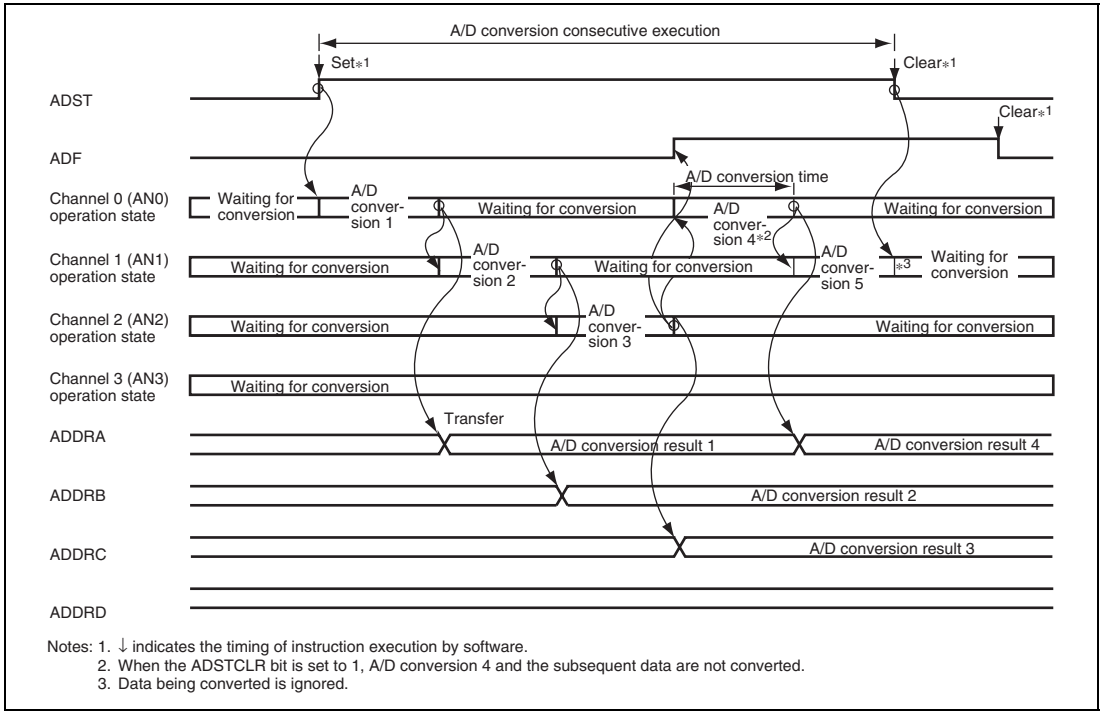
Figure 18.3 Example of A/D Converter Operation (Single Mode, Channel 1 Selected)

18.4.2 Scan Mode

In scan mode, A/D conversion is to be performed sequentially on the analog inputs of the specified channels of up to four or eight channels.

1. When the ADST bit in ADCSR is set to 1 by software, TPU, or an external trigger input, A/D conversion starts on the first channel in the group. Consecutive A/D conversion on a maximum of four channels (SCANE and SCANS = B'10) or on a maximum of eight channels (SCANE and SCANS = B'11) can be selected. When consecutive A/D conversion is performed on four channels, A/D conversion starts on AN0 when CH3 and CH2 = B'00 or on AN4 when CH3 and CH2 = B'01. When consecutive A/D conversion is performed on eight channels, A/D conversion starts on AN0 when CH3 = B'0.
2. When A/D conversion for each channel is completed, the A/D conversion result is sequentially transferred to the corresponding ADDR of each channel.
3. When A/D conversion of all selected channels is completed, the ADF bit in ADCSR is set to 1. If the ADIE bit is set to 1 at this time, an ADI interrupt request is generated. A/D conversion of the first channel in the group starts again.

4. The ADST bit is not cleared automatically, and steps [2] and [3] are repeated as long as the ADST bit remains set to 1. When the ADST bit is cleared to 0, A/D conversion stops and the A/D converter enters wait state. If the ADST bit is later set to 1, A/D conversion starts again from the first channel in the group.
5. If the ADSTCLR bit in ADCR is set to 1, the ADST bit is automatically cleared when A/D conversion of all selected channels is completed. Then A/D conversion stops and the A/D converter enters wait state.



Notes: 1. ↓ indicates the timing of instruction execution by software.
2. When the ADSTCLR bit is set to 1, A/D conversion 4 and the subsequent data are not converted.
3. Data being converted is ignored.

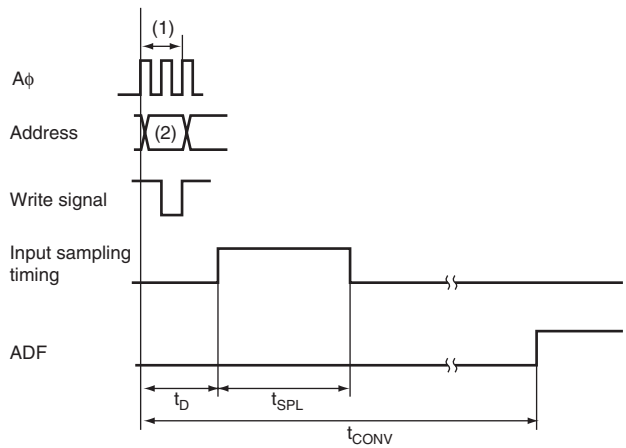
Figure 18.4 Example of A/D Converter Operation (Scan Mode, Three Channels (AN0 to AN2) Selected)

18.4.3 Input Sampling and A/D Conversion Time

The A/D converter has a built-in sample-and-hold circuit. The A/D converter samples the analog input when the A/D conversion start delay time (t_D) elapses after the ADST bit in ADCSR is set to 1, then starts A/D conversion. Figure 18.5 shows the A/D conversion timing. Table 18.3 indicates the A/D conversion time.

As indicated in figure 18.5, the A/D conversion time (t_{CONV}) includes t_D and the input sampling time (t_{SPL}). The length of t_D varies depending on the timing of the write access to ADCSR. The total conversion time therefore varies within the ranges indicated in table 18.3.

In scan mode, the values given in table 18.3 apply to the first conversion time. The values given in table 18.4 apply to the second and subsequent conversions. In either case, bits CKS1 and CKS0 in ADCR should be set so that the conversion time is within the ranges indicated by the A/D conversion characteristics.



[Legend]
(1): ADCSR write cycle
(2): ADCSR address
t_D: A/D conversion start delay time
t_{SPL}: Input sampling time
t_{CONV}: A/D conversion time

Figure 18.5 A/D Conversion Timing

Table 18.3 A/D Conversion Characteristics

		CKS1 = 0						CKS1 = 1					
		CKS0 = 0			CKS0 = 1			CKS0 = 0			CKS0 = 1		
Item	**Symbol**	**Min.**	**Typ.**	**Max.**	**Min.**	**Typ.**	**Max.**	**Min.**	**Typ.**	**Max.**	**Min.**	**Typ.**	**Max.**
A/D conversion start delay time	t_D	3	—	5	3	—	6	3	—	8	3	—	12
Input sampling time	t_{SPL}	—	15	—	—	30	—	—	60	—	—	120	—
A/D conversion time	t_{CONV}	45	—	47	85	—	88	165	—	170	325	—	334

Table 18.4 A/D Conversion Time (Scan Mode)

CKS1	CKS0	Conversion Time (Number of States)
0	0	40 (fixed)
	1	80 (fixed)
1	0	160 (fixed)
	1	320 (fixed)

18.4.4 External Trigger Input Timing

A/D conversion can be externally triggered. When the TRGS1, TRGS0, and EXTRGS bits are set to B'110 in ADCR, an external trigger is input from the $\overline{\text{ADTRG0}}$ pin. A/D conversion starts when the ADST bit in ADCSR is set to 1 on the falling edge of the $\overline{\text{ADTRG0}}$ pin. Other operations, in both single and scan modes, are the same as when the ADST bit has been set to 1 by software. Figure 18.6 shows the timing.

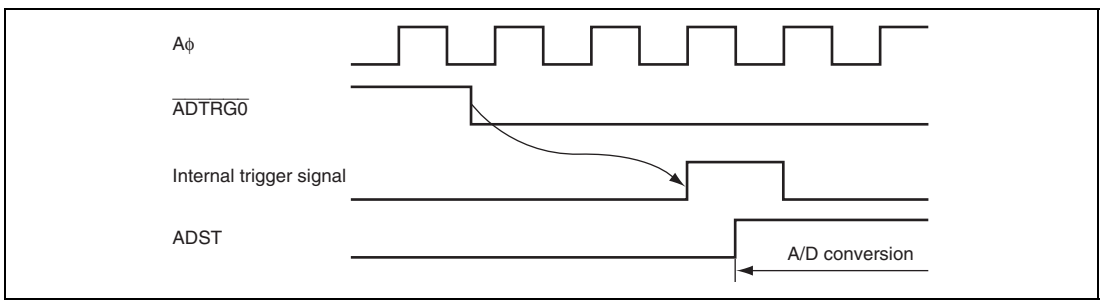


Figure 18.6 External Trigger Input Timing

18.5 Interrupt Source

The A/D converter generates an A/D conversion end interrupt (ADI) at the end of A/D conversion. Setting the ADIE bit to 1 when the ADF bit in ADCSR is set to 1 after A/D conversion is completed enables ADI interrupt requests. The data transfer controller (DTC) and the DMA controller (DMAC) can be activated by an ADI interrupt. Having the converted data read by the DTC or the DMAC in response to an ADI interrupt enables continuous conversion to be achieved without imposing a load on software.

Table 18.5 A/D Converter Interrupt Source

Abbr.	Interrupt Source	Interrupt Flag	DTC Activation	DMAC Activation
ADI	A/D conversion end	ADF	Possible	Possible

18.6 A/D Conversion Accuracy Definitions

This LSI's A/D conversion accuracy definitions are given below.

- Resolution

  The number of A/D converter digital output codes.
- Quantization error

  The deviation inherent in the A/D converter, given by 1/2 LSB (see figure 18.7).
- Offset error

  The deviation of the analog input voltage value from the ideal A/D conversion characteristic when the digital output changes from the minimum voltage value B'0000000000 (H'000) to B'0000000001 (H'001) (see figure 18.8).
- Full-scale error

  The deviation of the analog input voltage value from the ideal A/D conversion characteristic when the digital output changes from B'1111111110 (H'3FE) to B'1111111111 (H'3FF) (see figure 18.8).
- Nonlinearity error

  The error with respect to the ideal A/D conversion characteristic between the zero voltage and the full-scale voltage. Does not include the offset error, full-scale error, or quantization error (see figure 18.8).
- Absolute accuracy

  The deviation between the digital value and the analog input value. Includes the offset error, full-scale error, quantization error, and nonlinearity error.

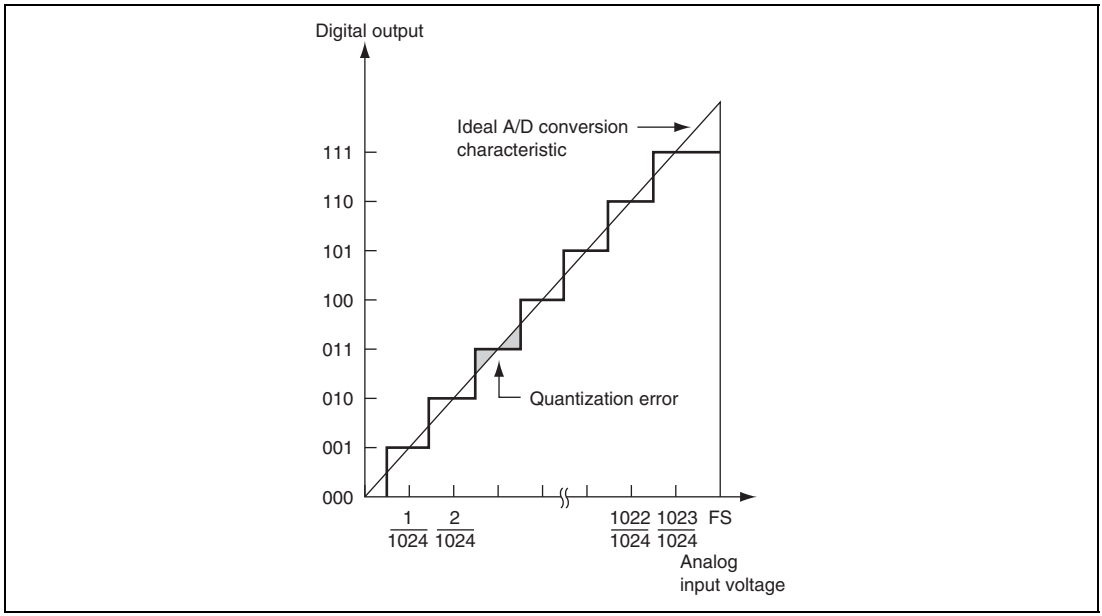


Figure 18.7 A/D Conversion Accuracy Definitions

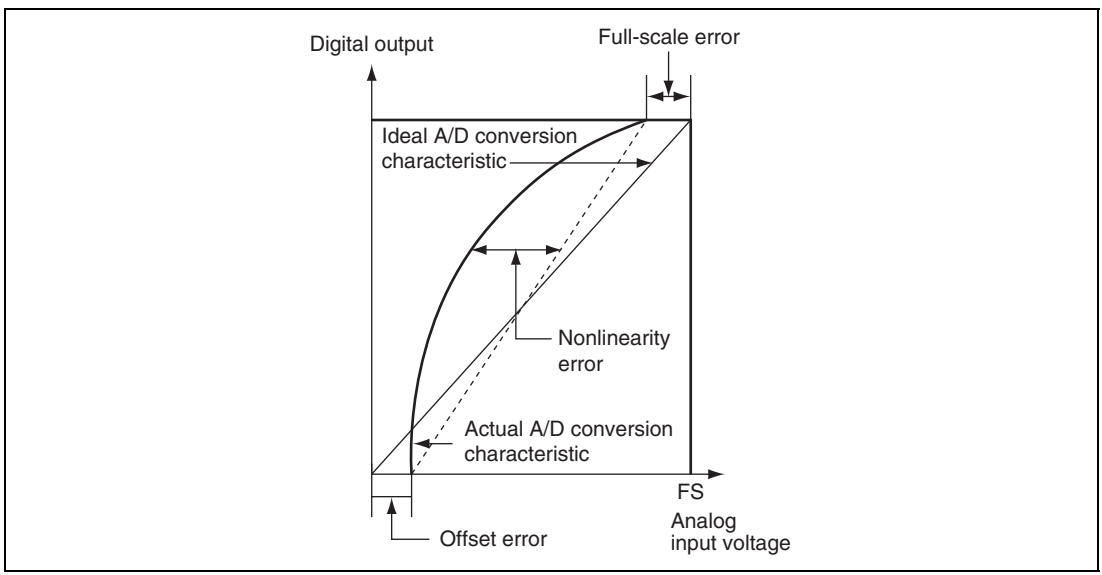


Figure 18.8 A/D Conversion Accuracy Definitions

18.7 Analog Port Pull-Down Function

Each of the analog port pins has a pull-down MOS transistor, which can be turned on by setting the corresponding bits from AN7PD to AN0PD in APPDCR to 1 (initial value is 0).

This allows, when voltage is applied to the analog pin from an external circuit, checking for disconnection of the signal line between the analog port pin and the external unit from the results of A/D conversion.

- Example

  When normal: The obtained result of conversion is close to the voltage that is applied by the external circuitry.

  When abnormal (disconnection): The obtained result of conversion is close to AVss.

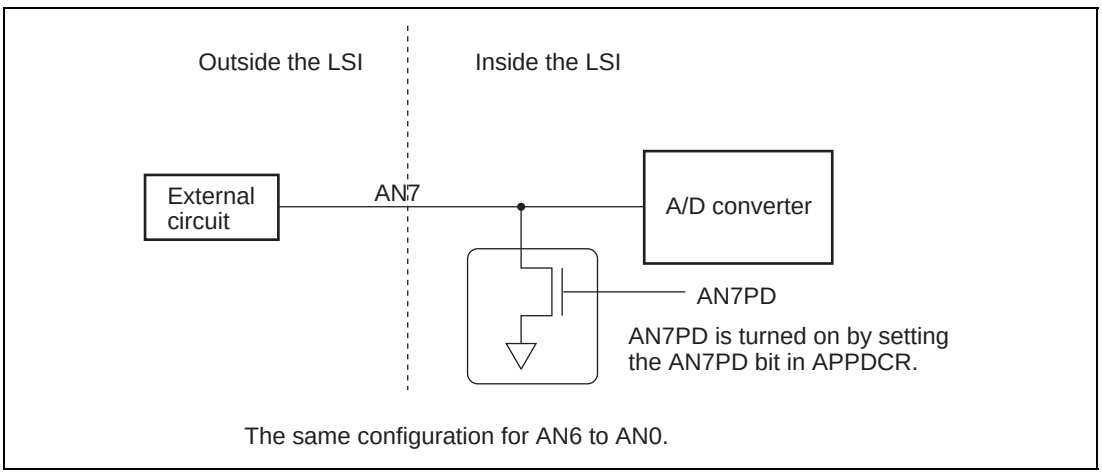


Figure 18.9 Overview of Analog Port Pull-Down Function

18.8 Self-Diagnosis of A/D Converter

Self-diagnosis of the A/D converter by software is possible.

A/D conversion is performed on the internally generated voltage value selected by the ADDIAGR register. After A/D conversion ends, whether the converted value is within the normal range (normal) or not (error) can be found out from the values read from the ADDR and ADDIAGR registers by software.

The analog inputs are ignored while the internal voltage value is selected for A/D conversion. To prevent malfunction, ADDIAGR should be set while the ADST bit in ADCSR is 0.

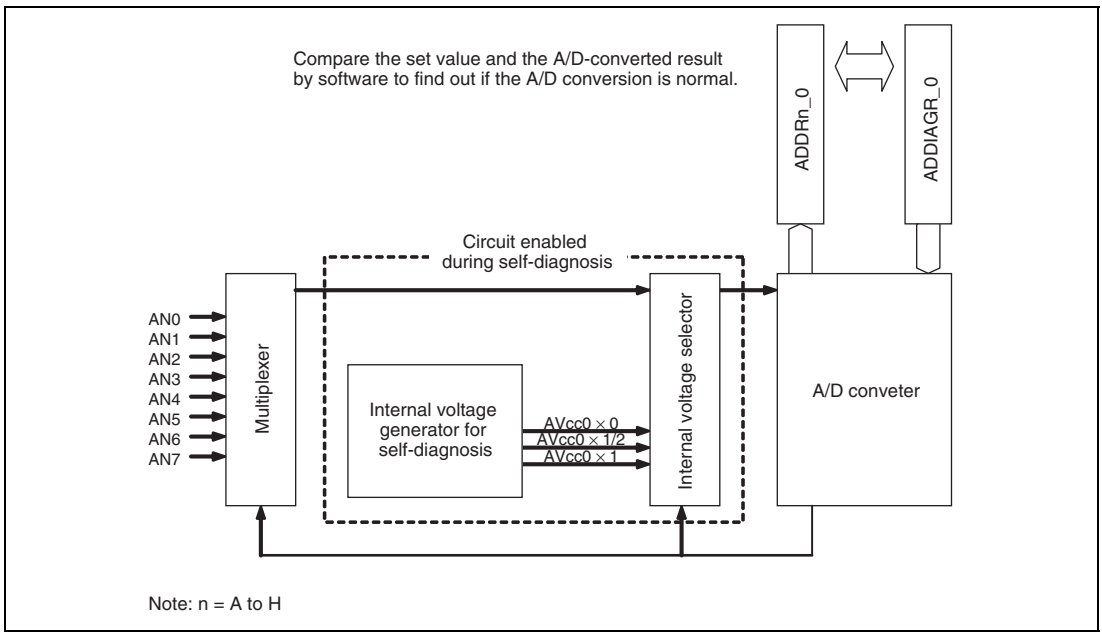


Figure 18.10 Overview of Self-Diagnosis of the A/D Converter

Table 18.6 Ideal Values for the Result of A/D Conversion (within Normal Range)

Selected Internal Voltage	ADDR Value (within Normal Range)
AVcc0 × 0	H'0000 (H'0000 to H'0A00*)
AVcc0 × 1/2	H'7FC0 (H'75C0 to H'89C0*)
AVcc0 × 1	H'FFC0 (H'F5C0 to H'FFC0*)

Note: * These values include the error of the internal voltage generator.

18.9 Usage Notes

18.9.1 Module Stop Mode Setting

Operation of the A/D converter can be disabled or enabled using the module stop control register. The initial setting is for operation of the A/D converter to be disabled. Register access is enabled by clearing module stop mode. When placing the A/D converter in module stop mode after A/D conversion has ended, clear all of the ADST, TRGS1, TRGS0, and EXTRGS bits and the APPDCR and ADDIAGR registers to 0 to disable A/D conversion. After that, dummy-read one word and then set the module stop control register. For details, refer to section 23, Power-Down Modes.

18.9.2 A/D Conversion Hold Function in Software Standby Mode

When this LSI enters software standby mode with A/D conversion enabled, the A/D conversion is retained, and the analog power supply current is equal to as during A/D conversion. If the analog power supply current needs to be reduced in software standby mode, clear all of the ADST, TRGS1, TRGS0, and EXTRGS bits and the APPDCR and ADDIAGR registers to 0 to disable A/D conversion. After that, dummy-read one word and then place the LSI in software standby mode.

18.9.3 Permissible Signal Source Impedance

This LSI's analog input is designed so that the conversion accuracy is guaranteed for an input signal for which the signal source impedance is 5 kΩ or less. This specification is provided to enable the A/D converter's sample-and-hold circuit input capacitance to be charged within the sampling time; if the sensor output impedance exceeds 5 kΩ, charging may be insufficient and it may not be possible to guarantee the A/D conversion accuracy. However, if a large capacitance is provided externally for conversion in single mode, the input load will essentially comprise only the internal input resistance of 10 kΩ, and the signal source impedance is ignored. However, since a low-pass filter effect is obtained in this case, it may not be possible to follow an analog signal with a large differential coefficient (e.g., 5 mV/µs or greater) (see figure 18.11). For converting a high-speed analog signal or conversion in scan mode, a low-impedance buffer should be inserted.

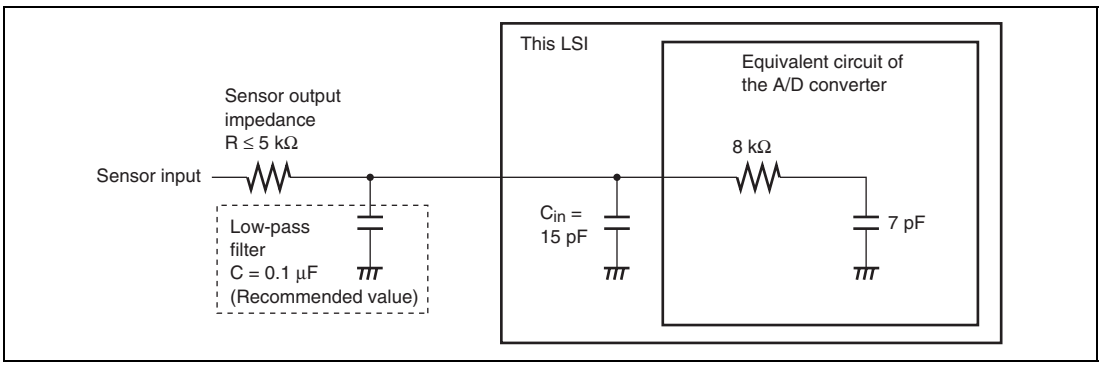


Figure 18.11 Example of Analog Input Circuit

18.9.4 Influences on Absolute Accuracy

Adding capacitance results in coupling with GND, and therefore noise in GND may adversely affect absolute accuracy. Be sure to make the connection to an electrically stable GND such as AVss.

Care is also required to insure that digital signals on the board do not interfere with filter circuits and filter circuits do not act as antennas.

18.9.5 Setting Range of Analog Power Supply and Other Pins

If the conditions shown below are not met, the reliability of the LSI may be adversely affected.

- Analog input voltage range

  The voltage applied to analog input pin ANn during A/D conversion should be in the range AVss ≤ V_{AN} ≤ AVcc0 and AVss ≤ V_{AN} ≤ AVcc1.
- Relation between AVcc0, AVcc1, and AVss and Vcc and Vss

  As the relationship between AVcc0, AVcc1, and AVss and Vcc and Vss, set AVcc0 = Vcc±0.3V, AVcc1 = Vcc±0.3V, and AVss = Vss. If the A/D converter is not used, set AVcc0 = Vcc, AVcc1 = Vcc, and AVss = Vss.

18.9.6 Notes on Board Design

In board design, digital circuitry and analog circuitry should be as mutually isolated as possible, and layout in which digital circuit signal lines and analog circuit signal lines cross or are in close proximity should be avoided as far as possible. Failure to do so may result in incorrect operation of the analog circuitry due to inductance, adversely affecting A/D conversion values. Digital circuitry must be isolated from the analog input pins (AN0 to AN15), and analog power supply voltage pins (AVcc0, AVcc1) by the analog ground (AVss). Also, the analog ground (AVss) should be connected at one point to a stable ground (Vss) on the board.

18.9.7 Notes on Noise Countermeasures

A protection circuit connected to prevent damage due to an abnormal voltage such as an excessive surge at the analog input pins (AN0 to AN15) should be connected between AVcc0 and AVss and between AVcc1 and AVss as shown in figure 18.12. Also, the bypass capacitors connected to AVcc0 and AVcc1 and the filter capacitor connected to pins AN0 to AN15 must be connected to AVss.

If a filter capacitor is connected, the input currents at pins AN0 to AN15 are averaged, and so an error may arise. Also, when A/D conversion is performed frequently, as in scan mode, if the current charged and discharged by the capacitance of the sample-and-hold circuit in the A/D converter exceeds the current input via the input impedance (R_{in}), an error will arise in the analog input pin voltage. Careful consideration is therefore required when deciding the circuit constants.

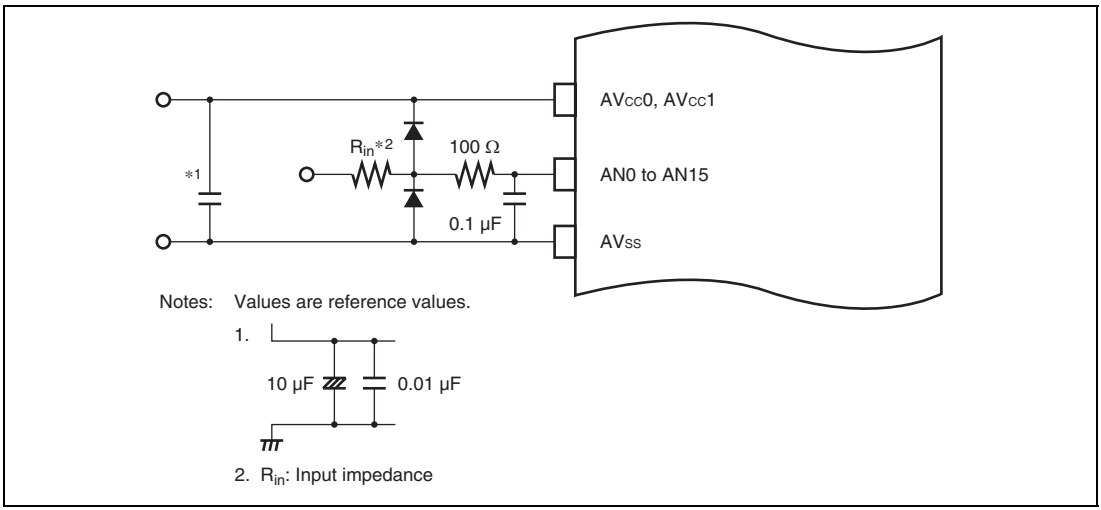
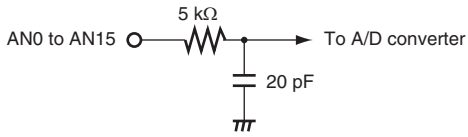


Figure 18.12 Example of Analog Input Protection Circuit

Table 18.7 Analog Pin Specifications

Item	Min.	Max.	Unit
Analog input capacitance	—	20	pF
Permissible signal source impedance	—	5	kΩ



Note: Values are reference values.

Figure 18.13 Analog Input Pin Equivalent Circuit

18.9.8 Points for Caution in Procedures for Stopping the A/D Converter

Clearing the A/D start (ADST) bit while A/D conversion by the A/D converter is in progress may lead to the results of conversion at the destination for storage being incorrect, the interrupt flag being erroneously set when conversion is restarted, etc.

Avoid such effects through the following countermeasures.

(1) Single Mode and Scan Mode (Single-Cycle Mode)

Since the ADST bit is automatically cleared on completion of A/D conversion, do not have software clear the ADST bit while A/D conversion is in progress.

(2) Scan Mode (Continuous Scan Mode)

(a) When Operation of the A/D Converter is Initiated by Software

Do not have software clear the ADST bit while A/D conversion is in progress. Stop A/D conversion by changing the SCANE bit from the setting for scan mode to the setting for single mode. The A/D converter is thus stopped without explicit clearing of the ADST bit by software.

However, up to 1.5 times the time for A/D conversion on a single channel may elapse after the change to the setting of the SCANE bit until A/D conversion is stopped and the A/D end flag (ADF) is set to 1. Furthermore, do not use the value in ADDR after A/D conversion has stopped.

(b) When Operation of the A/D Converter is Initiated by an External Trigger

Do not have software clear the ADST bit while A/D conversion is in progress. Disable A/D conversion by an external trigger and then stop A/D conversion by changing the SCANE bit from the setting for scan mode to the setting for single mode. The A/D converter is thus stopped without explicit clearing of the ADST bit by software.

However, up to 1.5 times the time for A/D conversion on a single channel may elapse after the change to the setting of the SCANE bit until A/D conversion is stopped and the A/D end flag (ADF) is set to 1. Furthermore, do not use the value in ADDR after A/D conversion has stopped.

See figures 18.14 and 18.15 for details on the settings under (2) above.

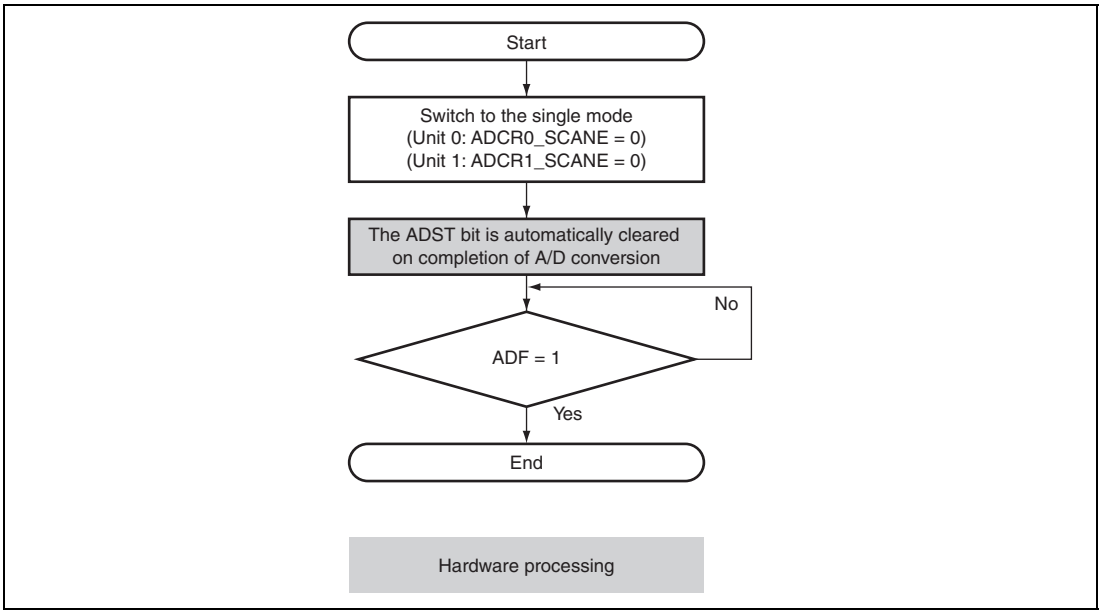


Figure 18.14 When Continuous Scan Mode is Stopped by Software

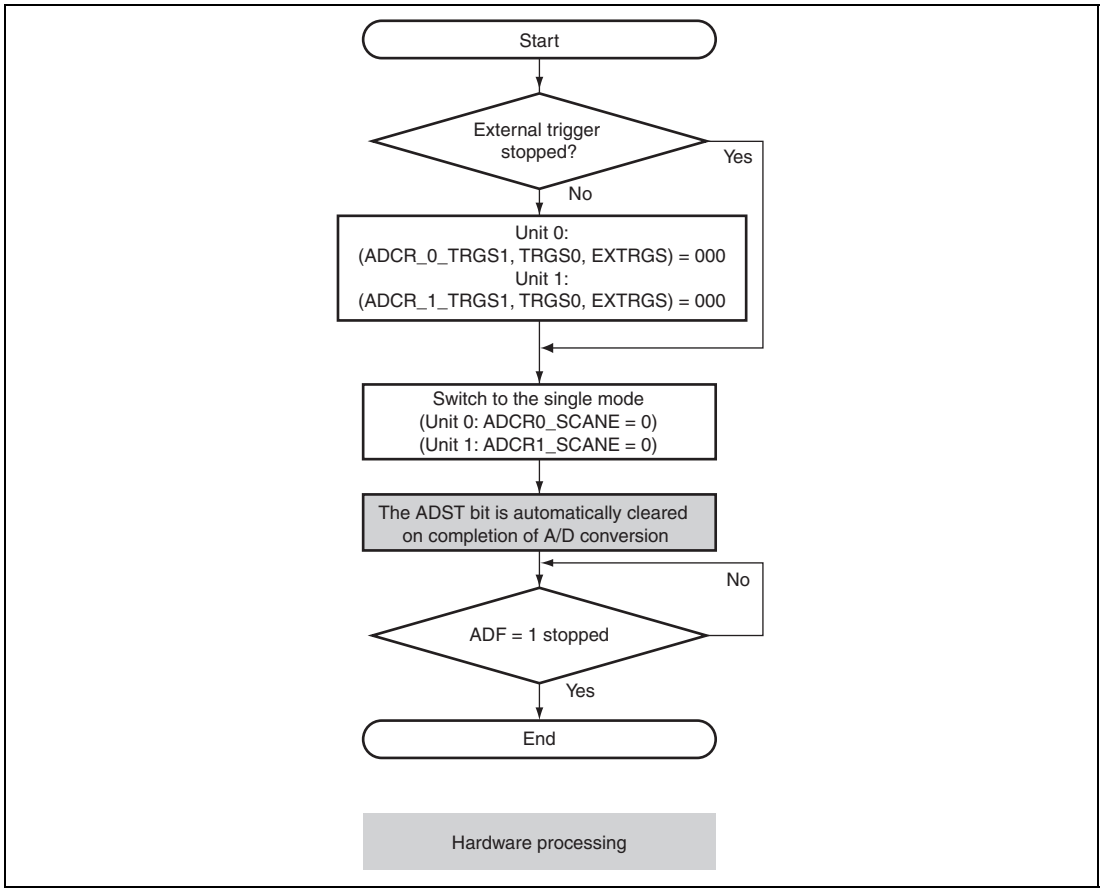


Figure 18.15 When Continuous Scan Mode is Stopped by an External Trigger

Section 19 RAM

This LSI has an on-chip RAM of up to 40 Kbytes. The on-chip RAM is connected to the 32-bit internal bus and is accessed in 1 cycle of Iϕ for reading and 1 or 2 cycle(s) of Iϕ for writing, regardless of whether the operation is for a byte, word, or longword.

Table 19.1 lists the on-chip RAM address space, and table 19.2 lists the on-chip RAM areas.

The description in this section refers to the 40-Kbyte version. In the 24-Kbyte version, areas 3 and 4 are spaces to which access is disabled.

19.1 Features

- Access

  The CPU, DMAC, and DTC can access on-chip RAM in 8, 16, or 32 bits. Data in the on-chip RAM can be effectively used as program area or stack area data necessary for access at high speed.
- ECC

  The ECC error correction function can be enabled or disabled by a register setting. In the initial state, the ECC error correction function is enabled; in 32-bit data, a 1-bit error can be corrected and a 2-bit error can be detected. When the ECC error correction function is enabled, writing takes two cycles.
- Parity

  When the ECC error correction function is disabled, the function for error detection by parity is enabled. This can detect a 1-bit error in 8-bit data. When the ECC error correction function is disabled (parity error detection function is enabled), writing time is set to one cycle.
- Error flag

  This LSI includes a flag that indicates the occurrence of a RAM error (ECC error or parity error).
- Interrupts

  Whether or not an interrupt is requested for a RAM error (ECC error or parity error) can be set by a register.
- RAM access protection

  Access to the on-chip RAM can be enabled or disabled for each area of on-chip RAM addresses.
- RAM write protection

  Writing to the on-chip RAM can be enabled or disabled for each area of on-chip RAM addresses.

- Control register rewrite

  The method for writing to the RAM control registers is different from that for the general registers; this prevents easy overwriting of values in the registers.

Table 19.1 On-Chip RAM Address Space

Product Classification		RAM Size	RAM Address
Flash memory version	H8SX/1727S	40 Kbytes	H'FF2000 to H'FFBFFF
	H8SX/1725S	24 Kbytes	H'FF6000 to H'FFBFFF

Table 19.2 On-Chip RAM Area

RAM Area	Address
Area 4 (8 Kbytes)	H'FF2000 to H'FF3FFF*
Area 3 (8 Kbytes)	H'FF4000 to H'FF5FFF*
Area 2 (8 Kbytes)	H'FF6000 to H'FF7FFF
Area 1 (8 Kbytes)	H'FF8000 to H'FF9FFF
Area 0 (8 Kbytes)	H'FFA000 to H'FFBFFF

Note: * Not supported by the 24-Kbyte version.

19.2 Register Descriptions

The on-chip RAM has the registers shown in table 19.3.

Table 19.3 Register Configuration

Register Name	Abbreviation	R/W	Initial Value	Address	Access Size
RAM enable control register	RAMEN	R/W	H'00FF	H'FFE400	8, (16)*
RAM write enable control register	RAMWEN	R/W	H'00FF	H'FFE402	8, (16)*
RAM ECC enable control register	RAMECC	R/W	H'0000	H'FFE404	8, (16)*
RAM error status register	RAMERR	R	H'00	H'FFE406	8
RAM error interrupt control register	RAMINT	R/W	H'00	H'FFE410	8
RAM access cycle set register	RAMACYC	R/W	H'0020	H'FFE412	8, (16)*

Note: * When writing to these registers, be sure to write key data in the upper byte and access these registers in words.

19.2.1 RAM Enable Control Register (RAMEN)

RAMEN is a 16-bit readable/writable register that enables or disables the access to the on-chip RAM. RAMEN is initialized to H'00FF by a reset or in the standby state. RAMEN can be written to in words, and can be read in bytes or words.

If a RAME bit corresponding to the area is set to 1, accesses to the on-chip RAM becomes enabled; while if the RAME bit is cleared to 0, the on-chip RAM cannot be accessed. In the access disabled state, an undefined data is read if the area is read or if an instruction in the area is fetched, and a write to the area is ignored. The initial value of the RAME bit is 1.

To rewrite the RAME bits, word size data with upper byte as H'96 and lower byte as write data must be written.

When the upper byte (bits 15 to 8) of RAMEN is read, H'00 is always read.

An instruction to access the on-chip RAM must not be placed immediately after an instruction to write to RAMEN. Otherwise, correct access to the on-chip RAM cannot be guaranteed.

To enable the access to the on-chip RAM by setting the RAME bit to 1, an instruction to read RAMEN must be placed immediately after an instruction to write to RAMEN.

Bit	15	14	13	12	11	10	9	8
Bit Name	RNKEY7	RNKEY6	RNKEY5	RNKEY4	RNKEY3	RNKEY2	RNKEY1	RNKEY0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)
Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	RAME4*	RAME3*	RAME2	RAME1	RAME0
Initial Value:	1	1	1	1	1	1	1	1
R/W:	R	R	R	R/W	R/W	R/W	R/W	R/W

Note: To avoid erroneous rewriting, the way of writing data to this register is different from that to other general registers. For details, see section 19.2.7, Notes on Register Access.
* Not supported by the 24-Kbyte version.

Bit	Bit Name	Initial Value	R/W	Description
15	RNKEY7	0	R/(W)	These bits enable or disable write to the RAME bit. H'96: Enable write to bits RAME7 to RAME0. The write data is not retained and these bits are always read as H'00. Other than H'96: Disable write to bits RAME7 to RAME0.
14	RNKEY6	0	R/(W)	
13	RNKEY5	0	R/(W)	
12	RNKEY4	0	R/(W)	
11	RNKEY3	0	R/(W)	
10	RNKEY2	0	R/(W)	
9	RNKEY1	0	R/(W)	
8	RNKEY0	0	R/(W)	
7 to 5	—	All 1	R	Reserved These bits are always read as 1. The write value should always be 1.
4	RAME4*	1	R/W	RAM Enable 4 Enables or disables access to area 4 in the on-chip RAM. 0: Disables access to area 4 in the on-chip RAM 1: Enables access to area 4 in the on-chip RAM [Clearing condition] • Writing 0 (H'96 is written to the upper byte simultaneously.) [Setting conditions] • Reset or standby • Writing 1 (H'96 is written to the upper byte simultaneously.)

Bit	Bit Name	Initial Value	R/W	Description
3	RAME3*	1	R/W	RAM Enable 3 Enables or disables access to area 3 in the on-chip RAM. 0: Disables access to area 3 in the on-chip RAM 1: Enables access to area 3 in the on-chip RAM [Clearing condition] • Writing 0 (H'96 is written to the upper byte simultaneously.) [Setting conditions] • Reset or standby • Writing 1 (H'96 is written to the upper byte simultaneously.)
2	RAME2	1	R/W	RAM Enable 2 Enables or disables access to area 2 in the on-chip RAM. 0: Disables access to area 2 in the on-chip RAM 1: Enables access to area 2 in the on-chip RAM [Clearing condition] • Writing 0 (H'96 is written to the upper byte simultaneously.) [Setting conditions] • Reset or standby • Writing 1 (H'96 is written to the upper byte simultaneously.)

Bit	Bit Name	Initial Value	R/W	Description
1	RAME1	1	R/W	RAM Enable 1 Enables or disables access to area 1 in the on-chip RAM. 0: Disables access to area 1 in the on-chip RAM 1: Enables access to area 1 in the on-chip RAM [Clearing condition] • Writing 0 (H'96 is written to the upper byte simultaneously.) [Setting conditions] • Reset or standby • Writing 1 (H'96 is written to the upper byte simultaneously.)
0	RAME0	1	R/W	RAM Enable 0 Enables or disables access to area 0 in the on-chip RAM. 0: Disables access to area 0 in the on-chip RAM 1: Enables access to area 0 in the on-chip RAM [Clearing condition] • Writing 0 (H'96 is written to the upper byte simultaneously.) [Setting conditions] • Reset or standby • Writing 1 (H'96 is written to the upper byte simultaneously.)

Note: * Not supported by the 24-Kbyte version.

19.2.2 RAM Write Enable Control Register (RAMWEN)

RAMWEN is a 16-bit readable/writable register that enables or disables the access to the on-chip RAM. RAMWEN is initialized to H'00FF by a reset or in the standby state. RAMWEN can be written to in words, and can be read in bytes or words.

If a RAMWE bit corresponding to the area to be accessed is set to 1, writing to the on-chip RAM becomes enabled; while if the RAMWE bit is cleared to 0, the on-chip RAM cannot be written to. In the access disabled state, a write to the on-chip RAM is ignored. The initial value of the RAMWE bit is 1.

To rewrite the RAMWE bits, word size data with upper byte as H'69 and lower byte as write data must be written.

When the upper byte (bits 15 to 8) of RAMWEN is read, the read value is always H'00.

An instruction to access the on-chip RAM must not be placed immediately after an instruction to write to RAMEN. Otherwise, correct access to the on-chip RAM cannot be guaranteed.

To enable the access to the on-chip RAM by setting the RAMWE bit to 1, an instruction to read RAMWEN must be placed immediately after an instruction to write to RAMWEN.

Bit	15	14	13	12	11	10	9	8
Bit Name	RWNKEY7	RWNKEY6	RWNKEY5	RWNKEY4	RWNKEY3	RWNKEY2	RWNKEY1	RWNKEY0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	RAMWE4*	RAMWE3*	RAMWE2	RAMWE1	RAMWE0
Initial Value:	1	1	1	1	1	1	1	1
R/W:	R	R	R	R/W	R/W	R/W	R/W	R/W

Note: To avoid erroneous rewriting, the way of writing data to this register is different from that to other general registers. For details, see section 19.2.7, Notes on Register Access.

* Not supported by the 24-Kbyte version.

Bit	Bit Name	Initial Value	R/W	Description
15	RWNKEY7	0	R/(W)	These bits enable or disable write to the RAMWE bit.
14	RWNKEY6	0	R/(W)	H'69: Enable write to bits RAMWE7 to RAMWE0. The write data is not retained and these bits are always read as H'00.
13	RWNKEY5	0	R/(W)	
12	RWNKEY4	0	R/(W)	
11	RWNKEY3	0	R/(W)	Other than H'69: Disable write to bits RAMWE7 to RAMWE0.
10	RWNKEY2	0	R/(W)	
9	RWNKEY1	0	R/(W)	
8	RWNKEY0	0	R/(W)	
7 to 5	—	All 1	R	Reserved These bits are always read as 1. The write value should always be 1.
4	RAMWE4*	1	R/W	RAM Write Enable 4 Enables or disables write to area 4 in the on-chip RAM. 0: Disables write to area 4 in the on-chip RAM 1: Enables write to area 4 in the on-chip RAM [Clearing condition] • Writing 0 (H'69 is written to the upper byte simultaneously.) [Setting conditions] • Reset or standby • Writing 1 (H'69 is written to the upper byte simultaneously.)

Bit	Bit Name	Initial Value	R/W	Description
3	RAMWE3*	1	R/W	RAM Write Enable 3 Enables or disables write to area 3 in the on-chip RAM. 0: Disables write to area 3 in the on-chip RAM 1: Enables write to area 3 in the on-chip RAM [Clearing condition] • Writing 0 (H'69 is written to the upper byte simultaneously.) [Setting conditions] • Reset or standby • Writing 1 (H'69 is written to the upper byte simultaneously.)
2	RAMWE2	1	R/W	RAM Write Enable 2 Enables or disables write to area 2 in the on-chip RAM. 0: Disables write to area 2 in the on-chip RAM 1: Enables write to area 2 in the on-chip RAM [Clearing condition] • Writing 0 (H'69 is written to the upper byte simultaneously.) [Setting conditions] • Reset or standby • Writing 1 (H'69 is written to the upper byte simultaneously.)

Bit	Bit Name	Initial Value	R/W	Description
1	RAMWE1	1	R/W	RAM Write Enable 1 Enables or disables write to area 1 in the on-chip RAM. 0: Disables write to area 1 in the on-chip RAM 1: Enables write to area 1 in the on-chip RAM [Clearing condition] • Writing 0 (H'69 is written to the upper byte simultaneously.) [Setting conditions] • Reset or standby • Writing 1 (H'69 is written to the upper byte simultaneously.)
0	RAMWE0	1	R/W	RAM Write Enable 0 Enables or disables write to area 0 in the on-chip RAM. 0: Disables write to area 0 in the on-chip RAM 1: Enables write to area 0 in the on-chip RAM [Clearing condition] • Writing 0 (H'69 is written to the upper byte simultaneously.) [Setting conditions] • Reset or standby • Writing 1 (H'69 is written to the upper byte simultaneously.)

Note: * Not supported by the 24-Kbyte version.

19.2.3 RAM ECC Enable Control Register (RAMECC)

RAMECC enables or disables the ECC correction function. RAMECC is initialized by a reset or in the standby state.

RAMECC can be written to in words, and can be read in bytes or words. To rewrite the RAMECC bits, word size data with upper byte as H'76 and lower byte as write data must be written.

When the upper byte (bits 15 to 8) of RAMECC is read, the read value is always H’00.

Do not place an instruction to access to the on-chip RAM immediately after an instruction to write to RAMECC; otherwise correct access to the on-chip RAM cannot be guaranteed.

When disabling ECC error correction by setting the RECCA bit to 1, place an instruction to read RAMECC immediately after an instruction to write to RAMECC.

Bit	15	14	13	12	11	10	9	8
Bit Name	REKEY7	REKEY6	REKEY5	REKEY4	REKEY3	REKEY2	REKEY1	REKEY0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	RECCA
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R/W

Note: To avoid erroneous rewriting, the way of writing data to this register is different from that to other general registers. For details, see section 19.2.7, Notes on Register Access.

Bit	Bit Name	Initial Value	R/W	Description
15	REKEY7	0	R/(W)	These bits enable or disable write to the RECCA bit.
14	REKEY6	0	R/(W)	H'76: Enable write to the RECCA bit. The write data is not retained and these bits are always read as H'00.
13	REKEY5	0	R/(W)	
12	REKEY4	0	R/(W)	Other than H'76: Disable write to the RECCA bit.
11	REKEY3	0	R/(W)	
10	REKEY2	0	R/(W)	
9	REKEY1	0	R/(W)	
8	REKEY0	0	R/(W)	
7 to 1	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
0	RECCA	0	R/W	Enables or disables the ECC error correction function. 0: Enables ECC error correction function 1: Disables ECC error correction function (parity error detection function is enabled) [Clearing conditions] • Reset or standby • Writing 0 (H'76 is written to the upper byte simultaneously.) [Setting condition] • Writing 1 (H'76 is written to the upper byte simultaneously.)

19.2.4 RAM Error Status Register (RAMERR)

RAMERR monitors ECC error occurrence when the ECC error correction function is enabled by RAMECC and also monitors the parity error occurrence when the ECC error correction function is disabled by RAMECC.

RAMERR is initialized by a reset or in the standby state. RAMERR can be read or written to in bytes.

If an ECC error occurs while reading the on-chip RAM with the ECC error correction function enabled, the RERRC bit is set to 1. If a 2-bit error is detected, the RDTCT bit is set to 1. If a parity error occurs while reading the on-chip RAM with the ECC error correction function disabled, the RERRC bit is set to 1.

If the ECC error correction function is disabled by RAMECC after the RERRC bit is set to 1, the RERRC bit remains set to 1.

When bits 7 to 1 of RAMERR are read, the read value is always 0.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	RDTCT	RERRC
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 2	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
1	RDTCT	0	R/W	RAM 2-Bit Error Detection Monitor Monitors whether or not 2-bit error detection occurs when the ECC error correction function is enabled. 0: No 2-bit error detection has occurred 1: 2-bit error detection has occurred [Clearing conditions] • Reset or standby • Writing 0 to this bit after reading 1 from it [Setting condition] • Occurrence of 2-bit error detection
0	RERRC	0	R/W	Monitors whether or not an ECC error or parity error occurs. 0: No ECC error or parity error occurs after clear 1: An ECC error or parity error occurs [Clearing conditions] • Reset or standby • Writing 0 [Setting condition] • Occurrence of ECC error or parity error

19.2.5 RAM Error Interrupt Control Register (RAMINT)

RAMINT enables or disables a RAM error interrupt. The RAM error includes an ECC error and parity error.

If a RAM error interrupt is enabled by setting the RINTC bit in this register to 1 and the ECC error correction function is enabled by the RAM ECC enable control register (RAMECC), an interrupt is generated when 1-bit error correction or 2-bit error detection occurs. An interrupt is also generated when 2-bit error detection occurs while the REDIE bit in this register is set to 1 to enable a 2-bit error detection interrupt.

If the ECC error correction function is disabled by the RAM ECC enable control register (RAMECC), an interrupt is generated when a parity error occurs.

RAMINT is initialized by a reset or in the standby state. RAMINT can be read or written to in bytes.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	REDIE	RINTC
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 2	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
1	REDIE	0	R/W	RAM 2-Bit Error Detection Interrupt Enables or disables an interrupt that will be generated on occurrence of 2-bit error detection when the ECC error correction function is enabled. 0: Disables a 2-bit error detection interrupt 1: Enables a 2-bit error detection interrupt
0	RINTC	0	R/W	RAM Error Interrupt Enable Enables or disables an interrupt that will be generated on occurrence of 1-bit error correction when the ECC error correction function is enabled and on occurrence of a parity error. An interrupt is also generated when a 2-bit error is detected. (RAM 1-bit error correction and parity errors are collectively referred to as RAM errors.) 0: Disables a RAM error interrupt 1: Enables a RAM error interrupt [Clearing conditions] • Reset or standby • Writing 0 [Setting condition] • Writing 1

19.2.6 RAM Access Cycle Set Register (RAMACYC)

RAMACYC sets the RAM read or write cycle. The cycle setting depends on the enabling or disabling of the ECC error correction function. Use the recommended setting. Otherwise, operation is not guaranteed. The WRCYC bit is initialized by a reset or in the standby state.

To write the RAMACYC bits, word size data with upper byte as H'78 and lower byte as write data must be written. RAMACYC can be written to only in words, and can be read from in bytes or words.

When the upper byte (bits 15 to 8) of RAMACYC is read, the read value is always H'00.

Do not write to RAMCYC while RAM is being accessed. Also, do not place the instruction that accesses the on-chip RAM immediately after the instruction that writes to RAMACYC; otherwise normal access is not guaranteed.

When setting an access cycle by setting bits WRCYC1 and WRCYC0 to 1, place the instruction that reads from RAMACYC immediately after the instruction that writes to RAMACYC.

Bit	15	14	13	12	11	10	9	8
Bit Name	RAKEY7	RAKEY6	RAKEY5	RAKEY4	RAKEY3	RAKEY2	RAKEY1	RAKEY0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	WRCYC1	WRCYC0	—	—	—	—
Initial Value:	0	0	1	0	0	0	0	0
R/W:	R	R	R/W	R/W	R	R	R	R

Note: To avoid erroneous rewriting, the way of writing data to this register is different from that to other general registers. For details, see section 19.2.7, Notes on Register Access.

Bit	Bit Name	Initial Value	R/W	Description
15	RAKEY7	0	R/(W)	These bits enable or disable write to bits WRCYC[1] and WRCYC[0]. H'78: Enable write to bits WRCYC[1] and WRCYC[0]. The write data is not retained and these bits are always read as H'00. Other than H'78: Disable write to bits WRCYC[1] and WRCYC[0].
14	RAKEY6	0	R/(W)	
13	RAKEY5	0	R/(W)	
12	RAKEY4	0	R/(W)	
11	RAKEY3	0	R/(W)	
10	RAKEY2	0	R/(W)	
9	RAKEY1	0	R/(W)	
8	RAKEY0	0	R/(W)	
7, 6	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
5	WRCYC1	1	R/W	These bits set the RAM write cycle.
4	WRCYC0	0	R/W	00: Setting prohibited 01: Setting prohibited 10: Set write cycle to 2 cycles (Initial value: ECC error correction function enabled, parity error detection function disabled) 11: Set write cycle to 1 cycle (ECC error correction function disabled, parity error detection function enabled) [Clearing conditions] • Reset or standby • Writing B'00 (H'78 is written to the upper byte simultaneously.) [Setting condition] • H'78 is written to the upper byte simultaneously.
3 to 0	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.

19.2.7 Notes on Register Access

The method of writing data to the RAM enable control register (RAMEN), RAM write enable control register (RAMWEN), and RAM ECC enable control register (RAMECC) is different from that for the general registers; this prevents easy overwriting of values in the registers.

To write these registers, use the following methods. In addition, note that RAMEN, RAMWEN, RAMECC, and RAMACYC must be written in words. These registers cannot be written in byte or longword instructions. As shown in figure 19.1, key data should be written in the upper byte.

- To write data to RAMEN, transfer data with upper byte as H'96 and lower byte as write data.
- To write data to RAMWEN, transfer data with upper byte as H'69 and lower byte as write data.
- To write data to RAMECC, transfer data with upper byte as H'76 and lower byte as write data.
- To write data to RAMACYC, transfer data with upper byte as H'78 and lower byte as write data.

When the upper bytes (bits 15 to 8) of RAMEN, RAMWEN, RAMECC, and RAMACYC are read, the read value is always H'00.

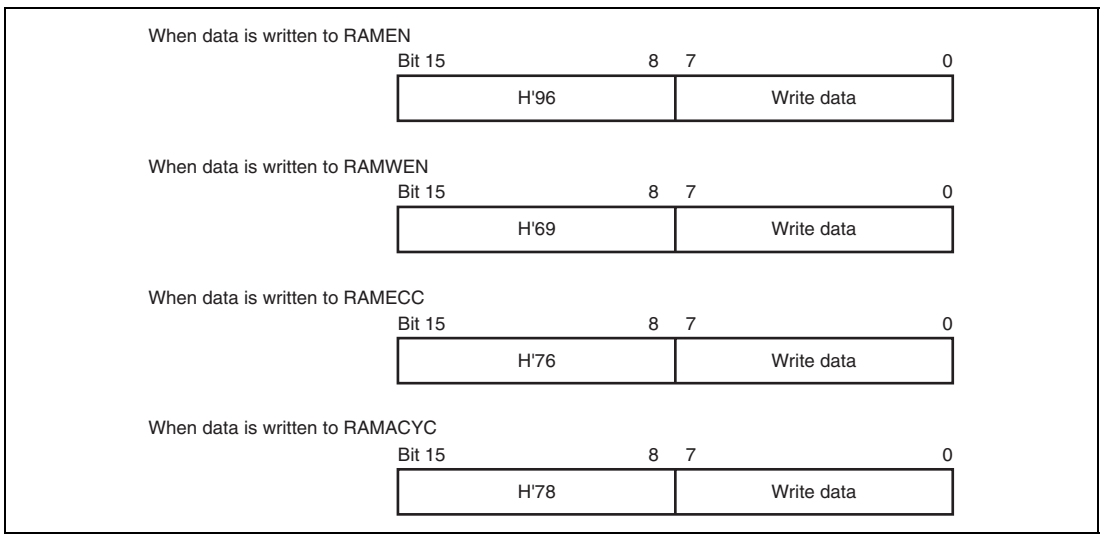


Figure 19.1 Writing Data to RAMEN, RAMWEN, RAMECC, and RAMACYC

19.3 Operations

Access to the on-chip RAM is controlled by the RAM enable control register (RAMEN) and RAM write enable control register (RAMWEN).

Accessing each area of the on-chip RAM is enabled or disabled by the RAME bits in the RAM enable control register (RAMEN). When the RAME bit of RAMEN is cleared to 0, the on-chip RAM cannot be accessed. In this case, values read from the on-chip RAM are undefined and the on-chip RAM cannot be modified.

Writing to each area of the on-chip RAM is enabled or disabled by the RAMWE bits in the RAM write enable control register (RAMWEN).

In addition, the ECC error correction function can be enabled or disabled according to the register settings. In the initial state, the ECC error correction function is enabled.

When the ECC error correction function is enabled, 1-bit error correction and 2-bit error detection can be performed. The writing period should be set to two cycles. This LSI provides a flag (RAM error/status register) to indicate an error correction when the ECC is used in error correction. If the RAM error/status register is set, an interrupt can be generated.

When the ECC error correction function is disabled, error detection is performed with a parity bit. Use the RAM access cycle set register (RAMACYC) to select one cycle as the period of writing. This LSI provides a flag (RAM error/status register) to indicate the occurrence of parity errors. If the RAM error/status register is set, an interrupt can be generated.

The generation of an interrupt can be enabled or disabled by the RAM error interrupt control register (RAMINT).

19.4 RAM Data Retention

19.4.1 Data Retention at Reset

If a low level signal is input on the $\overline{\text{RES}}$ pin from an external device while this LSI is in operation, this LSI enters the power-on reset state. In this case, if the on-chip RAM is accessed, data in the RAM address being accessed may be destroyed because the bus cycle cannot be completed normally.

It is difficult to input reset signals except for on-chip RAM from the external devices. Accordingly, to retain all data items in the on-chip RAM at reset, invalidate the RAM by the RAM enable control register (RAMEN).

19.5 Notes on Usage

19.5.1 RAM Initialization After Turning on Power

After turning on the power, all data items in the on-chip RAM including ECC error correction data and parity are undefined. Accordingly, correspondence among RAM data, error correction data, and parity may not be correct.

To enable correct (initialize) correspondence among RAM data, error correction data, and parity after power-on, write longword (32 bits) data to all on-chip RAM areas. If RAM is read without initialization, an ECC error or parity error may occur. Note that no ECC error or parity error occurs in RAM writes.

Section 20 Flash Memory

The H8SX/1727S and H8SX/1725S incorporate 512 Kbytes and 256 Kbytes of flash memory, respectively, for storing instruction code. The flash memory has the following features.

20.1 Features

- Flash-memory MAT
  User MAT: 512 Kbytes (H8SX/1727S), 256 Kbytes (H8SX/1725S)

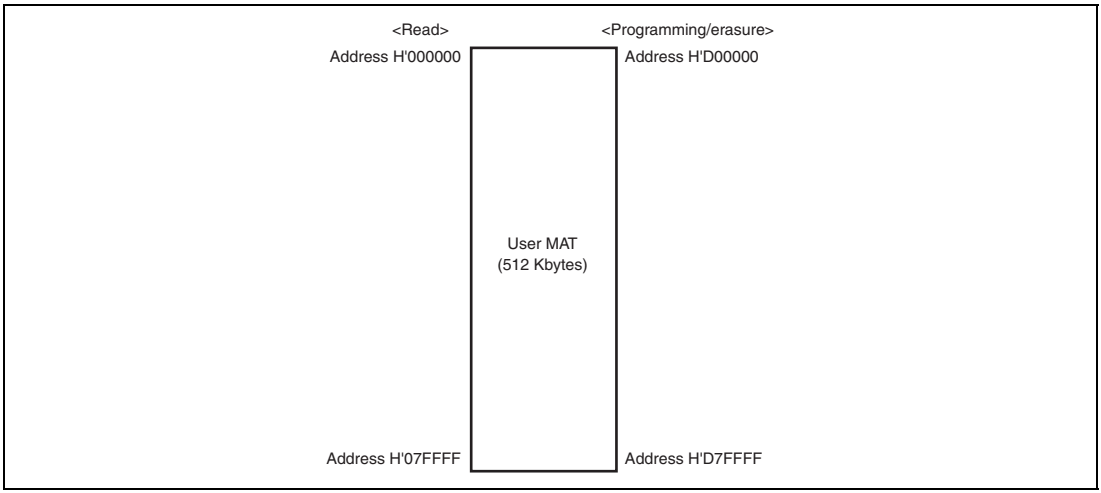


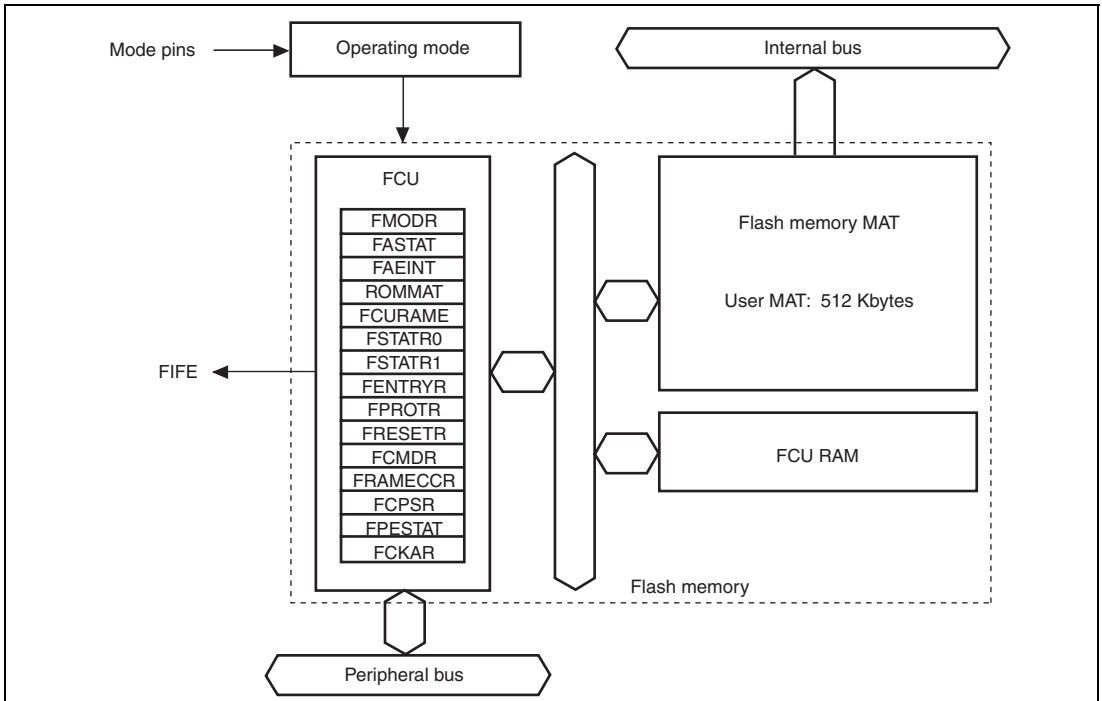
Figure 20.1 Memory MAT Configuration in Flash Memory

Notes: 1. This section describes only the 512-Kbyte flash memory.
2. For the 256-Kbyte memory MAT configuration, refer to section 3.3.4, Address Map.

- Programming and erasing methods

  The flash memory can be programmed and erased by commands issued through the peripheral bus to the flash memory/EEPROM-dedicated sequencer (FCU).

  While the FCU is programming or erasing the flash memory, the CPU can execute a program located outside the flash memory. While the FCU is programming or erasing the EEPROM, the CPU can execute a program in the flash memory. When the FCU suspends programming or erasure, the CPU can execute a program in the flash memory, and then the FCU can resume programming or erasure. While the FCU suspends erasure, areas other than the erasure-suspended area can be programmed.



[Legend]

FMODR:	Flash mode register
FASTAT:	Flash access status register
FAEINT:	Flash access error interrupt enable register
ROMMAT:	Flash memory MAT select register
FCURAME:	FCU RAM enable register
FSTATR0, FSTATR1:	Flash status registers 0 and 1
FENTRYR:	Flash P/E mode entry register
FPROTR:	Flash protect register
FRESETR:	Flash reset register
FCMDR:	FCU command register
FRAMECCR:	FCU RAM ECC error control register
FCPSR:	FCU processing switch register
FPESTAT:	Flash P/E status register
FCKAR:	Flash clock notification register
FIFE:	Flash interface error interrupt

Figure 20.2 Block Diagram of Flash Memory

- Programming/erasing unit

  The user MAT is programmed in 128-byte units. In a mode other than programmer mode, the user MAT can be erased in block units. In programmer mode, the entire area of the user MAT is erased at one time.

  Figure 20.3 shows the block configuration of the user MAT. The user MAT is divided into eight 4-Kbyte blocks, one 32-Kbyte block, and seven 64-Kbyte blocks in the H8SX/1727S, or eight 4-Kbyte blocks, one 32-Kbyte block, and three 64-Kbyte blocks in the H8SX/1725S.

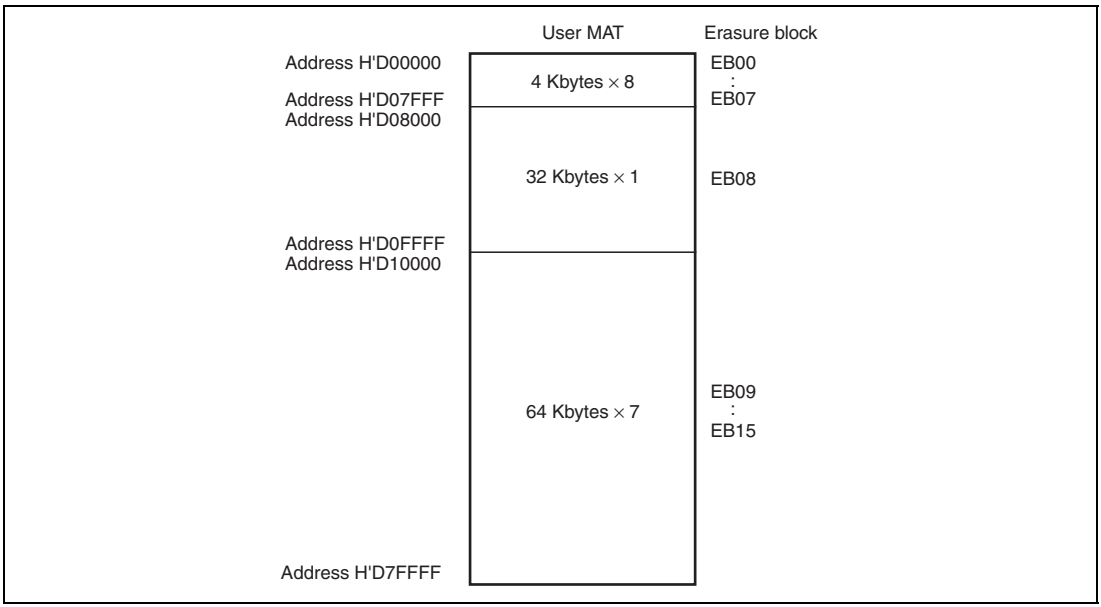


Figure 20.3 Block Configuration of User MAT

- Two types of on-board programming modes
  - — Boot mode

    The user MAT can be programmed using the SCI. The bit rate for SCI communication between the host and this LSI can be automatically adjusted.
  - — User program mode

    The user MAT can be programmed with a desired interface.
- One off-board programming mode
  - — Programmer mode

    The user MAT can be programmed using a PROM programmer.

- Protection modes

  This LSI supports two modes to protect memory against programming or erasure: hardware protection by the levels on the mode pin and software protection by the FENTRY0 bit or lock bit settings. The FENTRY0 bit enables or disables flash memory programming or erasure by the FCU. A lock bit is included in each erasure block of the user MAT to protect memory against programming or erasure.

  The LSI also provides a function to suspend programming or erasure when abnormal operation is detected during programming or erasure.
- Programming and erasing time and count

  Refer to section 25, Electrical Characteristics.

20.2 Input/Output Pins

Table 20.1 shows the input/output pins used for the flash memory. The combination of MD1 and MD0 pin levels determines the flash memory programming mode (see section 20.4, Overview of Flash Memory-Related Modes).

Table 20.1 Pin Configuration

Pin Name	Symbol	I/O	Function
Reset	$\overline{\text{RES}}$	Input	This LSI enters the reset state when this signal goes low.
Mode	MD1, MD0	Input	These pins specify the operating mode.
Receive data in SCI	RxD4	Input	Receives data through SCI channel 4
Transmit data in SCI	TxD4	Output	Transmits data through SCI channel 4

20.3 Register Descriptions

Table 20.2 shows the flash memory-related registers. Some of these registers have EEPROM-related bits, but this section only describes the flash memory-related bits. For the EEPROM-related bits, refer to section 21.3, Register Descriptions, in section 21, Data Flash (EEPROM).

Table 20.2 Register Configuration

Register Name	Symbol	R/W*1	Initial Value	Address	Access Size
Flash mode register	FMODR	R/W	H'00	H'FFE002	8
Flash access status register	FASTAT	R/(W)*2	H'00	H'FFE010	8
Flash access error interrupt enable register	FAEINT	R/W	H'9F	H'FFE011	8
Flash memory MAT select register	ROMMAT	R/(W)*3	H'0000 H'0001	H'FFE020	8, 16
FCU RAM enable register	FCURAME	R/(W)*3	H'0000	H'FFE054	8, 16
Flash status register 0	FSTATR0	R*5	H'80	H'FFE100	8, 16
Flash status register 1	FSTATR1	R*5	H'00	H'FFE101	8, 16
Flash P/E mode entry register	FENTRYR	R/(W)*4*5	H'0000	H'FFE102	8, 16
Flash protect register	FPROTR	R/(W)*4*5	H'0000	H'FFE104	8, 16
Flash reset register	FRESETR	R/(W)*3	H'0000	H'FFE106	8, 16
FCU command register	FCMDR	R*5	H'FFFF	H'FFE10A	8, 16
FCU RAM ECC error control register	FRAMECCR	R/W	H'02	H'FFE10C	8
FCU processing switch register	FCPSR	R/W*5	H'0000	H'FFE118	8, 16
Flash P/E mode status register	FPESTAT	R*5	H'0000	H'FFE11C	8, 16
Flash clock notification register	FCKAR	R/W*5	H'0000	H'FFE138	16

Notes: 1. In on-chip ROM disabled mode, the flash memory-related registers are always read as 0 and writing to them is ignored. (This LSI does not support on-chip ROM disabled mode.)
2. This register consists of the bits where only 0 can be written to clear the flags and the read-only bits.
3. This register can be written to only when a specified value is written to the upper byte in word access. The data written to the upper byte is not stored in the register.
4. This register can be written to only when a specified value is written to the upper byte in word access; the register is initialized when a value not allowed for the register is written to the upper byte. The data written to the upper byte is not stored in the register.
5. This register can be initialized by setting the FRESET bit in FRESETR to 1 as well as applying a reset through the $\overline{\text{RES}}$ pin.

20.3.1 Flash Mode Register (FMODR)

FMODR specifies the FCU operation mode.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	FRDMD	—	—	—	—
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R/W	R	R	R	R

Bit	Bit Name	Initial Value	R/W	Description
7 to 5	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
4	FRDMD	0	R/W	FCU Read Mode Select Selects a mode of flash memory/EEPROM reading. For flash memory, this bit is used to select the mode for reading the lock bit value (see section 20.6.1, FCU Command List, and (12) Reading Lock Bit in section 20.6.3). For EEPROM, this bit must be set appropriately when using the blank check command (see section 21, Data Flash (EEPROM)). 0: Memory area read mode Select this mode to read the lock bit in flash memory by flash memory lock bit read mode. 1: Register read mode Select this mode to read the lock bit in flash memory by using the lock bit read 2 command.
3 to 0	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.

20.3.2 Flash Access Status Register (FASTAT)

FASTAT indicates the access error status for the flash memory and EEPROM. If any bit in FASTAT is set to 1, the FCU enters command-locked state (see section 20.8.3, Error Protection). To cancel command-locked state, set FASTAT to H'10, and then issue a status-clear command to the FCU.

Bit	7	6	5	4	3	2	1	0
Bit Name	ROMAE	—	—	CMDLK	EEPAE	—	EEPRPE	EEPWPE
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R	R	R	R/(W)*	R	R/(W)*	R/(W)*

Note: * Only 0 can be written to clear the flag after 1 is read.

Bit	Bit Name	Initial Value	R/W	Description
7	ROMAE	0	R/(W)*	Flash Memory Access Error Indicates whether an access error has been generated for the flash memory. If this bit becomes 1, the ILGLERR bit in FSTATR0 is set to 1 and the FCU enters command-locked state. 0: No flash memory access error has occurred 1: A flash memory access error has occurred [Setting conditions] • A read access command is issued to flash memory program/erase addresses H'D00000 to H'D7FFFF while the FENTRY0 bit in FENTRYR is 1 in flash memory P/E normal mode. • An access command is issued to flash memory program/erase addresses H'D00000 to H'D7FFFF while the FENTRY0 bit in FENTRYR is 0. • A read access command is issued to flash memory read addresses H'000000 to H'07FFFF while the FENTRYR register holds H'0001. [Clearing condition] • 0 is written to this bit after reading ROMAE = 1.

Bit	Bit Name	Initial Value	R/W	Description
6, 5	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
4	CMDLK	0	R	FCU Command Lock Indicates whether the FCU is in command-locked state (see section 20.8.3, Error Protection). 0: The FCU is not in command-locked state 1: The FCU is in command-locked state [Setting condition] • The FCU detects an error and enters command-locked state. [Clearing condition] • The FCU completes the status-clear command processing while FASTAT is H'10.
3	EEPAE	0	R/(W)*	EEPROM Access Error Refer to section 21, Data Flash (EEPROM).
2	—	0	R	Reserved This bit is always read as 0. The write value should always be 0.
1	EEPRPE	0	R/(W)*	EEPROM Access Error Refer to section 21, Data Flash (EEPROM).
0	EEPWPE	0	R/(W)*	EEPROM Access Error Refer to section 21, Data Flash (EEPROM).

Note: * Only 0 can be written to clear the flag after 1 is read.

20.3.3 Flash Access Error Interrupt Enable Register (FAEINT)

FAEINT enables or disables output of flash interface error (FIFE) interrupts.

Bit	7	6	5	4	3	2	1	0
Bit Name	ROMAEIE	—	—	CMDLKIE	EEPAEIE	—	EEPRPEIE	EEPWPEIE
Initial Value:	1	0	0	1	1	1	1	1
R/W:	R/W	R	R	R/W	R/W	R	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	ROMAEIE	1	R/W	Flash Memory Access Error Interrupt Enable Enables or disables an FIFE interrupt request when a flash memory access error occurs and the ROMAE bit in FASTAT becomes 1. 0: Does not generate an FIFE interrupt request when ROMAE = 1 1: Generates an FIFE interrupt request when ROMAE = 1
6, 5	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
4	CMDLKIE	1	R/W	FCU Command Lock Interrupt Enable Enables or disables an FIFE interrupt request when FCU command-locked state is entered and the CMDLK bit in FASTAT becomes 1. 0: Does not generate an FIFE interrupt request when CMDLK = 1 1: Generates an FIFE interrupt request when CMDLK = 1
3	EEPAEIE	1	R/W	EEPROM Access Error Interrupt Enable Refer to section 21, Data Flash (EEPROM).
2	—	1	R	Reserved This bit is always read as 1. The write value should always be 1.

Bit	Bit Name	Initial Value	R/W	Description
1	EEPRPEIE	1	R/W	EEPROM Read Protect Error Interrupt Enable Refer to section 21, Data Flash (EEPROM).
0	EEPWPEIE	1	R/W	EEPROM Program/Erase Protect Error Interrupt Enable Refer to section 21, Data Flash (EEPROM).

20.3.4 Flash Memory MAT Select Register (ROMMAT)

ROMMAT switches memory MATs in the flash memory. Since this LSI does not have a user boot MAT, the value for writing to the ROMSEL bit must always be 0.

Bit	15	14	13	12	11	10	9	8
Bit Name	RMKEY7	RMKEY6	RMKEY5	RMKEY4	RMKEY3	RMKEY2	RMKEY1	RMKEY0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*
Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	ROMSEL
Initial Value:	0	0	0	0	0	0	0	0/1
R/W:	R	R	R	R	R	R	R	R/W

Note: * Written data is not stored in these bits.

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	RMKEY7 to RMKEY0	H'00	R/(W)*	Key Code These bits enable or disable ROMSEL bit modification. The data written to these bits are not stored.
7 to 1	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
0	ROMSEL	0/1	R/W	Flash Memory MAT Select Selects a memory MAT in the flash memory. The initial value is 1 when the LSI is started in user boot mode; otherwise, the initial value is 0. Writing to this bit is enabled only when this register is accessed in word size and H'3B is written to the RMKEY bits. 0: Selects the user MAT 1: Selects the user boot MAT Since this LSI does not have a user boot MAT, the value for writing to this bit must always be 0.

Note: * Written data is not stored in these bits.

20.3.5 FCU RAM Enable Register (FCURAME)

FCURAME enables or disables access to the FCU RAM area.

Bit	15	14	13	12	11	10	9	8
Bit Name	FCKEY7	FCKEY6	FCKEY5	FCKEY4	FCKEY3	FCKEY2	FCKEY1	FCKEY0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*
Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	FCRME
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R/W

Note: * Written data is not stored in these bits.

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	FCKEY7 to FCKEY0	H'00	R/(W)*	Key Code These bits enable or disable FCRME bit modification. The data written to these bits are not stored.
7 to 1	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
0	FCRME	0/1	R/W	FCU RAM Enable Enables or disables access to the FCU RAM. Writing to this bit is enabled only when this register is accessed in word size and H'C4 is written to the FCKEY bits. Before writing to the FCU RAM, clear FENTRYR to H'0000 to stop the FCU. 0: Disables access to FCU RAM 1: Enables access to FCU RAM

Note: * Written data is not stored in these bits.

20.3.6 Flash Status Register 0 (FSTATR0)

FSTATR0 indicates the FCU status and is initialized by setting the FRESET bit in FRESETR to 1 as well as applying a reset through the $\overline{\text{RES}}$ pin.

Bit	7	6	5	4	3	2	1	0
Bit Name	FRDY	ILGLERR	ERSERR	PRGERR	SUSRDY	—	ERSSPD	PRGSPD
Initial Value:	1	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R

Bit	Bit Name	Initial Value	R/W	Description
7	FRDY	1	R	Flash Ready Indicates the processing state in the FCU. 0: Programming or erasure processing, programming or erasure suspension processing, lock bit read 2 command processing, or EEPROM blank check processing (see section 21, Data Flash (EEPROM)) is in progress 1: None of the above is in progress
6	ILGLERR	0	R	Illegal Command Error Indicates that the FCU has detected an illegal command or illegal flash memory or EEPROM access. When this bit is 1, the FCU is in command-locked state (see section 20.8.3, Error Protection). 0: The FCU has not detected any illegal command or illegal flash memory/EEPROM access 1: The FCU has detected an illegal command or illegal flash memory/EEPROM access [Setting conditions] • The FCU has detected an illegal command. • The FCU has detected an illegal flash memory/EEPROM access (any one of bits ROMAE, EEPAE, EEPRPE, and EEPWPE in FASTAT is 1). • The FENTRYR setting is illegal. [Clearing condition] • The FCU completes the status-clear command processing while FASTAT is H'10.

Bit	Bit Name	Initial Value	R/W	Description
5	ERSERR	0	R	Erasure Error Indicates the result of flash memory or EEPROM erasure by the FCU. When this bit is 1, the FCU is in command-locked state (see section 20.8.3, Error Protection). 0: Erasure processing has been completed successfully 1: An error has occurred during erasure [Setting conditions] • An error has occurred during erasure. • A block erase command has been issued for the area protected by a lock bit. [Clearing condition] • The FCU has processed a status-clear command.
4	PRGERR	0	R	Programming Error Indicates the result of flash memory or EEPROM programming by the FCU. When this bit is 1, the FCU is in command-locked state (see section 20.8.3, Error Protection). 0: Programming has been completed successfully 1: An error has occurred during programming [Setting conditions] • An error has occurred during programming. • A programming command has been issued for the area protected by a lock bit. [Clearing condition] • The FCU has processed a status-clear command.

Bit	Bit Name	Initial Value	R/W	Description
3	SUSRDY	0	R	Suspend Ready Indicates whether the FCU is ready to accept a P/E suspend command. 0: The FCU cannot accept a P/E suspend command 1: The FCU can accept a P/E suspend command [Setting condition] • After programming/erasure processing is started, the FCU has entered a state in which acceptance of a P/E suspend command is possible. [Clearing conditions] • The FCU has accepted a P/E suspend command. • The FCU has entered a command-locked state during programming/erasure.
2	—	0	R	Reserved This bit is always read as 0. The write value should always be 0.
1	ERSSPD	0	R	Erasure-Suspended Status Indicates that the FCU is performing erasure-suspension processing or has entered an erasure-suspended state (see section 20.6.4, Suspending Operation). 0: States other than below 1: The FCU is performing erasure-suspension processing or in an erasure-suspended state [Setting condition] • The FCU has started erasure-suspension processing. [Clearing condition] • The FCU has accepted a resume command.

Bit	Bit Name	Initial Value	R/W	Description
0	PRGSPD	0	R	Programming-Suspended Status Indicates that the FCU is performing programming-suspension processing or has entered a programming-suspended state (see section 20.6.4, Suspending Operation). 0: States other than below 1: The FCU is performing programming-suspension processing or in a programming-suspended state [Setting condition] • The FCU has started programming-suspension processing. [Clearing condition] • The FCU has accepted a resume command.

20.3.7 Flash Status Register 1 (FSTATR1)

FSTATR1 indicates the FCU status and is initialized by setting the FRESET bit in FRESETR to 1 as well as applying a reset through the $\overline{\text{RES}}$ pin

Bit	7	6	5	4	3	2	1	0
Bit Name	FCUERR	—	—	FLOCKST	—	—	FRDTCT	FRCRCT
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R

Bit	Bit Name	Initial Value	R/W	Description
7	FCUERR	0	R	FCU Error Indicates an error has occurred during the CPU processing in the FCU. 0: No error has occurred during the CPU processing in the FCU 1: An error has occurred during the CPU processing in the FCU When FCUERR is 1, set the FRESET bit to 1 to initialize the FCU, and then copy the FCU firmware again from the FCU firmware area to the FCU RAM area.
6, 5	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
4	FLOCKST	0	R	Lock Bit Status Reflects the lock bit data read through lock bit read 2 command execution. When the FRDY bit becomes 1 after the lock bit read 2 command is issued, valid data is stored in this bit. This bit value is retained until the next lock bit read 2 command is completed. 0: Protected state 1: Non-protected state
3, 2	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.

Bit	Bit Name	Initial Value	R/W	Description
1	FRDTCT	0	R	FCU RAM 2-Bit Error Detection Monitor Indicates that a 2-bit error has been detected when the FCU is reading RAM. 0: No 2-bit error has been detected. 1: A 2-bit error has been detected. When FRDTCT is 1, set the FRESET bit to 1 to initialize the FCU, and then copy the FCU firmware again from the FCU firmware area to the FCU RAM area.
0	FRCRCT	0	R	FCU RAM 1-Bit Error Correction Monitor Indicates that a 1-bit error has been corrected when the FCU is reading RAM. 0: No 1-bit error has been corrected. 1: A 1-bit error has been corrected. When FRCRCT is 1, set the FRESET bit to 1 to initialize the FCU, and then copy the FCU firmware again from the FCU firmware area to the FCU RAM area.

20.3.8 FCU RAM ECC Error Control Register (FRAMECCR)

FRAMECCR enables or disables an FCU command lock request upon correction of a 1-bit error or detection of a 2-bit error when the FCU is reading the RAM. FRAMECCR enables or disables an FCU command lock request, but it does not control the FRDTCT and FRCRCT bits of the flash status register 1 (FSTATR1).

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	FRDCLE	FRCCLE
Initial Value	0	0	0	0	0	0	1	0
R/W	R	R	R	R	R	R	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7 to 2	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
1	FRDCLE	1	R/W	FCU Command Lock Enabling Bit upon FCU RAM 2-Bit Error Detection Enables or disables an FCU command-lock request upon detection of a 2-bit error when the FCU is reading RAM. If a 2-bit error is detected while this bit is set to 1, the CMDLK bit of FASTAT is set to 1. 0: Issues no FCU command-lock request upon detection of a 2-bit error. 1: Issues an FCU command-lock request upon detection of a 2-bit error.
0	FRCCLE	0	R/W	FCU Command Lock Enabling Bit upon FCU RAM 1-Bit Error Correction Enables or disables an FCU command-lock request upon correction of a 1-bit error when the FCU is reading RAM. If a 1-bit error is detected while this bit is set to 1, the CMDLK bit of FASTAT is set to 1. 0: Issues no FCU command-lock request upon correction of a 1-bit error. 1: Issues an FCU command-lock request upon correction of a 1-bit error.

20.3.9 Flash P/E Mode Entry Register (FENTRYR)

FENTRYR specifies the P/E mode for the flash memory or EEPROM. To specify the P/E mode for the flash memory so that the FCU can accept commands, set FENTRY0 to 1. FENTRYR is initialized by setting the FRESET bit in FRESETR to 1 as well as applying a reset through the $\overline{\text{RES}}$ pin.

Bit	15	14	13	12	11	10	9	8
Bit Name	FEKEY7	FEKEY6	FEKEY5	FEKEY4	FEKEY3	FEKEY2	FEKEY1	FEKEY0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*
Bit	7	6	5	4	3	2	1	0
Bit Name	FENTRYD	—	—	—	—	—	—	FENTRY0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R	R	R	R	R	R	R/W

Note: * Written data is not stored in these bits.

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	FEKEY7 to FEKEY0	H'00	R/(W)*	Key Code These bits enable or disable modification of the FENTRYD and FENTRY0 bit values. The data written to the FEKEY bits are not stored.
7	FENTRYD	0	R/W	EEPROM P/E Mode Entry Refer to section 21, Data Flash (EEPROM).
6 to 1	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.

Bit	Bit Name	Initial Value	R/W	Description
0	FENTRY0	0	R/W	Flash Memory P/E Mode Entry 0 Specifies the P/E mode for 512 Kbytes of the flash memory (read addresses: H'000000 to H'07FFFF; program/erase addresses: H'D00000 to H'D7FFFF). 0: Flash memory is in read mode. 1: Flash memory is in P/E mode. [Write enabling conditions] When all of the following conditions are satisfied: • The on-chip ROM is enabled. • The FRDY bit in FSTATR0 is 1. • H'AA is written to FEKEY in word access. [Setting condition] • 1 is written to FENTRY0 while the write enabling conditions are satisfied and FENTRYR is H'0000. [Clearing conditions] • This register is written to in byte access. • A value other than H'AA is written to FEKEY in word access. • 0 is written to FENTRY0 while the write enabling conditions are satisfied. • This register is written to while it is holding a value other than H'0000 when the write enabling conditions are satisfied.

Note: * Written data is not stored in these bits.

20.3.10 Flash Protect Register (FPROTR)

FPROTR enables or disables the protection function by the lock bits against programming and erasure. FPROTR is initialized by setting the FRESET bit in FRESETR to 1 as well as applying a reset through the $\overline{\text{RES}}$ pin.

Bit	15	14	13	12	11	10	9	8
Bit Name	FPKEY7	FPKEY6	FPKEY5	FPKEY4	FPKEY3	FPKEY2	FPKEY1	FPKEY0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*
Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	FPROTCN
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R/W

Note: * Written data is not stored in these bits.

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	FPKEY7 to FPKEY0	H'00	R/(W)*	Key Code These bits enable or disable FPROTCN bit modification. The data written to these bits are not stored.
7 to 1	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.

Bit	Bit Name	Initial Value	R/W	Description
0	FPROTCN	0	R/W	Lock Bit Protect Cancel Enables or disables protection by the lock bits against programming and erasure. 0: Enables protection by lock bits 1: Disables protection by lock bits [Setting condition] • H'55 is written to FPKEY and 1 is written to FPROTCN in word access while the FENTRYR register holds a value other than H'0000. [Clearing conditions] • This register is written to in byte access. • A value other than H'55 is written to FPKEY in word access. • H'55 is written to FPKEY and 0 is written to FPROTCN in word access. • The value of the FENTRYR register is H'0000.

Note: * Written data is not stored in these bits.

20.3.11 Flash Reset Register (FRESETR)

FRESETR is used to initialize the FCU.

Bit	15	14	13	12	11	10	9	8
Bit Name	FRKEY7	FRKEY6	FRKEY5	FRKEY4	FRKEY3	FRKEY2	FRKEY1	FRKEY0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*
Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	FRESET
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R/W

Note: * Written data is not stored in these bits.

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	FRKEY7 to FRKEY0	H'00	R/(W)*	Key Code These bits enable or disable FRESET bit modification. The data written to these bits are not stored.
7 to 1	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
0	FRESET	0	R/W	Flash Reset Setting this bit to 1 forcibly terminates programming/erasure of flash memory/EEPROM and initializes the FCU. A high voltage is applied to flash memory/EEPROM while it is being programmed or erased. To ensure enough time for the applied voltage to drop, FRESET must be held at 1 for a period of t_{RESW2} when initializing the FCU. While FRESET is held at 1, access to flash memory/EEPROM should be disabled. Note that the FCU commands cannot be used while FRESET is set to 1 because the FENTRYR register is initialized. Writing to FRESET is only possible when H'CC is written to FRKEY in word access. 0: Does not reset the FCU 1: Resets the FCU

Note: * Written data is not stored in these bits.

20.3.12 FCU Command Register (FCMDR)

FCMDR stores the commands that the FCU has accepted. FCMDR is initialized by setting the FRESET bit in FRESETR to 1 as well as applying a reset through the $\overline{\text{RES}}$ pin.

Bit	15	14	13	12	11	10	9	8
Bit Name	CMDR	CMDR	CMDR	CMDR	CMDR	CMDR	CMDR	CMDR
Initial Value:	1	1	1	1	1	1	1	1
R/W:	R	R	R	R	R	R	R	R
Bit	7	6	5	4	3	2	1	0
Bit Name	PCMDR	PCMDR	PCMDR	PCMDR	PCMDR	PCMDR	PCMDR	PCMDR
Initial Value:	1	1	1	1	1	1	1	1
R/W:	R	R	R	R	R	R	R	R

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	CMDR	H'FF	R	Command Register These bits store the latest command accepted by the FCU.
7 to 0	PCMDR	H'FF	R	Precommand Register These bits store the previous command accepted by the FCU.

Table 20.3 shows the FCMDR status after each command has been accepted.

Table 20.3 FCMDR Status after a Command is Accepted

Command	CMDR	PCMDR
Transition to normal mode	H'FF	Previous command
Transition to status read mode	H'70	Previous command
Transition to lock bit read mode (lock bit read 1)	H'71	Previous command
Flash clock notification command	H'E9	Previous command
Program	H'E8	Previous command
Block erase	H'D0	H'20
P/E suspend	H'B0	Previous command
P/E resume	H'D0	Previous command
Status register clear	H'50	Previous command
Lock bit read 2, blank check	H'D0	H'71
Lock bit program	H'D0	H'77

20.3.13 FCU Processing Switch Register (FCPSR)

FCPSR is a register that specifies the erasure-suspended mode of the FCU. FCPSR is initialized by setting the FRESET bit in FRESETR to 1 as well as applying a reset through the $\overline{\text{RES}}$ pin.

Bit	15	14	13	12	11	10	9	8
Bit Name	—	—	—	—	—	—	—	—
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	ESUSPMD
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R/W

Bit	Bit Name	Initial Value	R/W	Description
15 to 1	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
0	ESUSPMD	0	R/W	Erasure-Suspended Mode Selects the erasure-suspended mode to be entered when a P/E suspend command is issued while the FCU is erasing the flash memory or EEPROM (see section 20.6.4, Suspending Operation). 0: Suspension-priority mode 1: Erasure-priority mode

20.3.14 Flash P/E Status Register (FPESTAT)

FPESTAT is a register that indicates the result of flash memory/EEPROM programming/erasure. FPESTAT can be initialized by setting the FRESET bit in FRESETR to 1 as well as applying a reset through the $\overline{\text{RES}}$ pin.

Bit	15	14	13	12	11	10	9	8
Bit Name	—	—	—	—	—	—	—	—
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R

Bit	7	6	5	4	3	2	1	0
Bit Name	PEERRST	PEERRST	PEERRST	PEERRST	PEERRST	PEERRST	PEERRST	PEERRST
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
7 to 0	PEERRST	H'00	R/W	P/E Error Status These bits indicate the cause of the error that has occurred during programming or erasure of flash memory/EEPROM. The values of these bits are only valid when the PRGERR or ERSERR bit in FSTATR0 is 1. While both of the PRGERR and ERSERR bits are 0, these bits hold the value for the latest error. H'01: Programming error occurred because programming was attempted for an area protected by the lock bit. H'02: Programming error occurred due to a cause not relating to the lock bit. H'11: Erasure error occurred because erasure was attempted for an area protected by the lock bit. H'12: Erasure error occurred due to a cause not relating to the lock bit. Other values: Reserved

20.3.15 Flash Clock Notification Register (FCKAR)

FCKAR sends information of the flash clock (FCLK) frequency setting to the sequencer when flash memory or EEPROM is written to or erased. FCKAR is initialized by setting the FRESET bit in FRESETR to 1 as well as applying a reset through the $\overline{\text{RES}}$ pin.

Bit	15	14	13	12	11	10	9	8
Bit Name	—	—	—	—	—	—	—	—
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	FCKA7	FCKA6	FCKA5	FCKA4	FCKA3	FCKA2	FCKA1	FCKA0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
7 to 0	FCKA7 to FCKA0	All 0	R/W	Flash Clock Notification These bits specify the flash clock (FCLK) for programming or erasure for flash memory or EEPROM. Set the FCKA7 to FCKA0 bits to the FCLK frequency and issue a flash clock notification command before programming or erasure. Do not change the frequency during programming or erasure for flash memory or EEPROM.

Calculate the value for the FCKA7 to FCKA0 (flash clock notification) bits as follows.

1. Convert the desired operating frequency represented in MHz units to a binary value and write it to the FCKA7 to FCKA0 bits.

   For example, when the operating frequency of the flash clock is 35.9 MHz, the setting value is calculated as follows:
2. Round up 35.9 to 36.
3. Convert 36 to a binary value and set the upper bits and lower bits of the FCKA7 to FCKA0 bits to H'00 and H'24 (0010 0100), respectively.

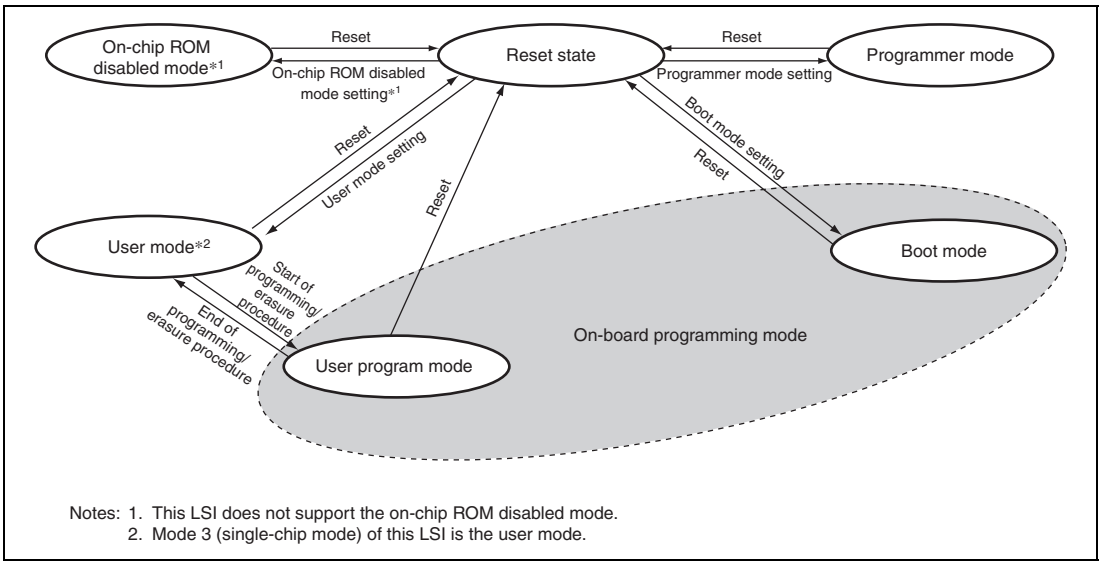
Notes: 1. When the FCKA7 to FCKA0 bits are set to a value outside the range from 8 MHz to 40 MHz, do not issue a programming command to flash memory or EEPROM.

2. When the FCKA7 to FCKA0 bits are set to a frequency that is different from the actual frequency, the data in flash memory or EEPROM may be lost.

3. Please note that the programming time depends on the frequency to some extent even if the FCKA7 to FCKA0 bits are used.

20.4 Overview of Flash Memory-Related Modes

Figure 20.4 shows the flash memory-related mode transition in this LSI. For the relationship between the LSI operating modes and the MD1 and MD0 pin settings, refer to section 3, MCU Operating Modes.



Notes: 1. This LSI does not support the on-chip ROM disabled mode.
2. Mode 3 (single-chip mode) of this LSI is the user mode.

Figure 20.4 Flash Memory-Related Mode Transition

- The flash memory cannot be read, programmed, or erased in on-chip ROM disabled mode. (This LSI does not support the on-chip ROM disabled mode.)
- The flash memory can be read but cannot be programmed or erased in user mode.
- The flash memory can be read, programmed, and erased on the board in user program mode and boot mode.
- The flash memory can be read, programmed, and erased through a PROM programmer in programmer mode.

Table 20.4 lists the comparison of programming- and erasure-related items for the boot mode, user program mode, and programmer mode.

Table 20.4 Comparison of Programming Modes

Item	Boot Mode	User Program Mode*2	Programmer Mode
Programming/erasure environment	On-board programming	On-board programming	Off-board programming
Programming/erasure enabled MAT	User MAT	User MAT	User MAT
Programming/erasure control	Host	FCU	PROM programmer
Entire area erasure	Available (automatic)	Available	Available (automatic)
Block erasure	Available*1	Available	Not available
Programming data transfer	From host via SCI	From any device via RAM	Via PROM programmer
Reset-start MAT	Embedded program stored MAT	User MAT	Embedded program stored MAT
Transition to MCU operating mode	Mode setting change and reset	End of programming/ erasure procedure	—

Notes: 1. The entire area is erased when the LSI is started. After that, a specified block can be erased.
2. In this LSI, user program mode is defined as a period between the start of the defined programming/erasure procedure in user mode and the end of the procedure. For the procedure of programming/erasure, see section 20.6, User Program Mode.

- In boot mode, the user MAT and EEPROM data MAT are all erased immediately after the LSI is started. The user MAT and data MAT can then be programmed from the host via the SCI. Reading of flash memory data also becomes possible after the erasure of the entire MAT.
- In boot mode and programmer mode, on-chip RAM is used by the boot program. Accordingly, if the LSI is reset with RAM enable register (RAMEN) set for disabling on-chip RAM and then started in boot mode or programmer mode, the data in on-chip RAM before reset will be lost (see section 19, RAM).

20.5 Boot Mode

20.5.1 System Configuration

To program or erase the user MAT and user boot MAT in boot mode, send control commands and programming data from the host. The on-chip SCI of this LSI is used in asynchronous mode for communication between the host and this LSI. The tool for sending control commands and programming data must be prepared in the host. When this LSI is started in boot mode, the program in the embedded program stored MAT is executed. This program automatically adjusts the SCI bit rate and performs communication between the host and this LSI by means of the control command method.

Figure 20.5 shows the system configuration in boot mode. The NMI and $\overline{\text{IRQ7}}$ to $\overline{\text{IRQ0}}$ interrupts are ignored in this mode, but these pins must be fixed to non-active state.

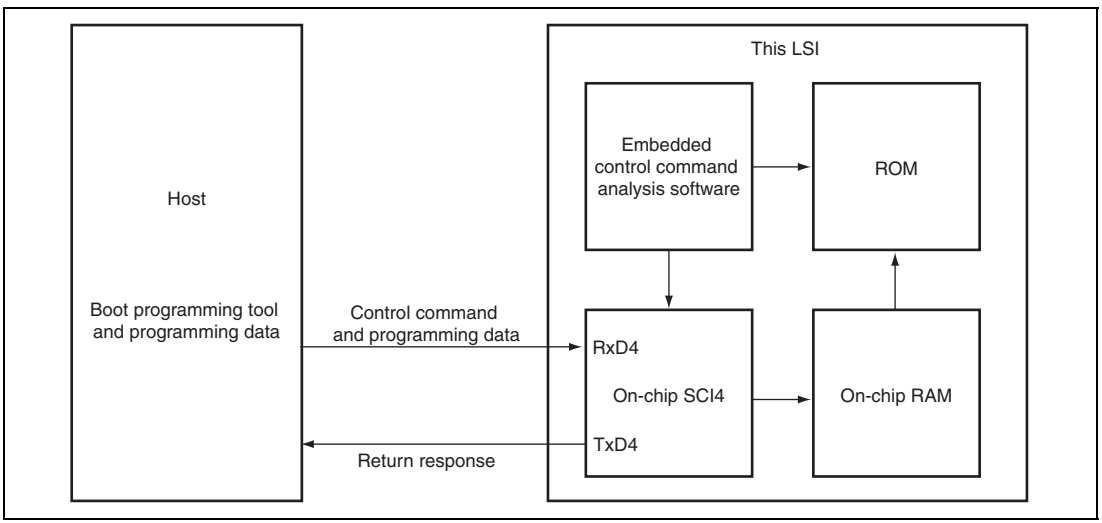


Figure 20.5 System Configuration in Boot Mode

20.5.2 State Transition in Boot Mode

Figure 20.6 shows the state transition in boot mode.

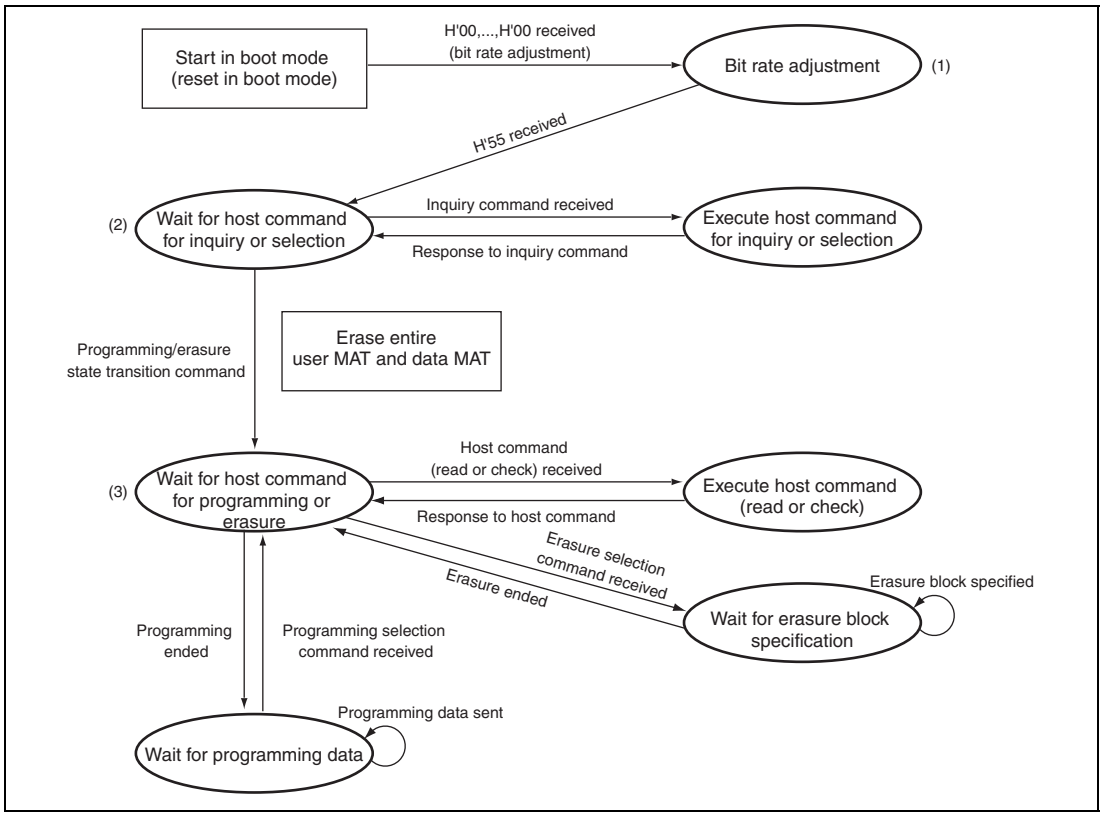


Figure 20.6 State Transition in Boot Mode

(1) Bit Rate Adjustment

After this LSI is started in boot mode, it automatically adjusts the bit rate for communication between the host and SCI4. After automatic adjustment of the bit rate, this LSI sends H'00 to the host. After this LSI has successfully received H'55 sent from the host, this LSI waits for a host command for inquiry or selection. For details on bit rate adjustment, see section 20.5.3, Automatic Adjustment of Bit Rate.

(2) Waiting for Host Command for Inquiry or Selection

In this state, the host inquires regarding MAT information (such as the size, configuration, and start address) and the supported functions, and selects the device, clock mode, and bit rate. Upon reception of a programming/erasure state transition command sent from the host, this LSI erases the entire area of the user MAT and EEPROM data MAT and waits for a host command for programming or erasure. For details of inquiry/selection host commands, see section 20.5.4, Inquiry/Selection Host Command Wait State.

(3) Waiting for Host Command for Programming or Erasure

In this state, this LSI performs programming or erasure according to the command sent from the host. This LSI enters programming data wait state, erasure block specification wait state, or command (read or check) processing state depending on the received command.

Upon reception of a programming selection command, this LSI waits for programming data. After the programming selection command, send the programming start address and programming data from the host. Specifying H'FFFFFF as the programming start address terminates programming processing and this LSI makes a transition from the programming data wait state to programming/erasure command wait state.

Upon reception of an erasure selection command, this LSI waits for erasure block specification. After the erasure selection command, send the erasure block number from the host. Specifying H'FF as the erasure block number terminates erasure processing and this LSI makes a transition from the erasure block specification wait state to programming/erasure host command wait state. As the entire area of the user MAT and EEPROM data MAT is erased before this LSI enters programming/erasure host command wait state after it is started in boot mode, erasure processing is not needed except for the case when the data programmed in boot mode should be erased without resetting this LSI.

In addition to programming and erasing commands, many other host commands are provided for use in programming/erasure command wait state; these include commands for checksum, blank check (erasure check), memory read, and status inquiry for the user MAT. For details on these host commands, see section 20.5.5, Programming/Erasing Host Command Wait State.

20.5.3 Automatic Adjustment of Bit Rate

When this LSI is started in boot mode, it measures the low-level (H'00) period of the data that is continuously sent from the host in asynchronous SCI communication. During this measurement, set the SCI transmit/receive format of the host to 8-bit data, 1 stop bit, and no parity, and set the bit rate to 9,600 bps or 19,200 bps. This LSI calculates the bit rate of the host SCI by means of the measured low-level period, and then sends H'00 to the host after completing the bit rate adjustment. When the host has received H'00 successfully, it must send H'55 to this LSI. If the host has failed to receive H'00, restart this LSI in boot mode to calculate and adjust the bit rate again. When this LSI has received H'55, it returns H'E6 to the host, or when it has failed to receive H'55, it returns H'FF.

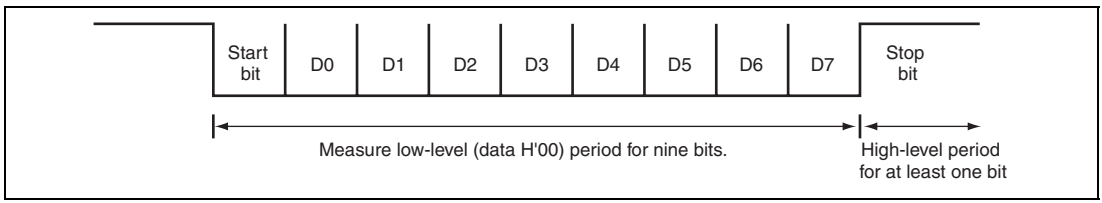


Figure 20.7 SCI Transmit/Receive Format for Automatic Adjustment of Bit Rate

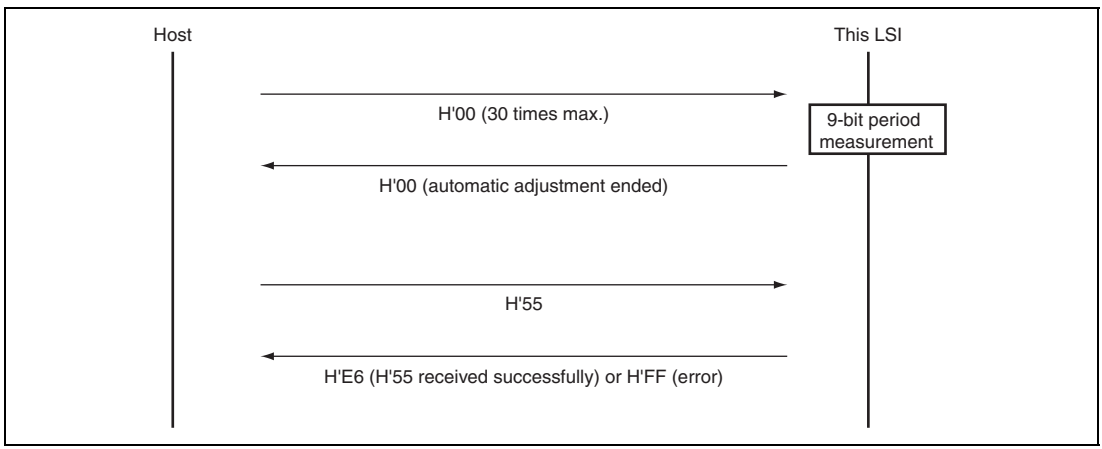


Figure 20.8 Communication Sequence between Host and this LSI

The bit rate may not be adjusted correctly depending on the bit rate of the host SCI or the peripheral clock frequency of this LSI. Satisfy the SCI communication condition as shown in table 20.5.

Table 20.5 Condition for Automatic Adjustment of Bit Rate

Host Bit Rate	Peripheral Clock Frequency of this LSI	External Clock Input Frequency
9,600 bps	16 to 20 MHz	8 to 10 MHz
19,200 bps	16 to 20 MHz	8 to 10 MHz

20.5.4 Inquiry/Selection Host Command Wait State

Table 20.6 shows the host commands available in inquiry/selection host command wait state. The boot program status inquiry command can also be used in programming/erasure host command wait state. The other commands can only be used in inquiry/selection host command wait state.

Table 20.6 Inquiry/Selection Host Commands

Host Command Name	Function
Supported device inquiry	Inquires regarding the device codes and the product codes for the embedded programs
Device selection	Selects a device code
Clock mode inquiry	Checks the clock mode
Clock mode selection	Selects a clock mode
Multiplication ratio inquiry	Inquires regarding the number of clock types, the number of multiplication/division ratios, and the multiplication /division ratios
Operating frequency inquiry	Inquires regarding the number of clock types and the maximum and minimum operating frequencies
User boot MAT information inquiry*	Inquires regarding the number of user boot MATs and the start and end addresses
User MAT information inquiry	Inquires regarding the number of user MATs and the start and end addresses
Erasure block information inquiry	Inquires regarding the number of blocks and the start and end addresses
Programming size inquiry	Inquires regarding the size of programming data

Host Command Name	Function
Simultaneous two-MAT programming information inquiry	Inquires regarding the availability of simultaneous two-MAT programming function
New bit rate selection	Modifies the bit rate of SCI communication between the host and this LSI
Programming/erasure state transition	Erases the entire area of each of the user MAT, user boot MAT, and EEPROM data MAT and makes this LSI enter programming/erasure host command wait state
Boot program status inquiry	Inquires regarding the state of this LSI

Note: * This LSI does not have user boot MAT.

If the host has sent an undefined command, this LSI returns a response indicating a command error in the format shown below. The command field holds the first byte of the undefined command sent from the host.

Error response	H'80	Command

In inquiry/selection host command wait state, send selection commands from the host in the order of device selection, clock mode selection, and new bit rate selection to set up this LSI according to the responses to inquiry commands. Note that the supported device inquiry and clock mode inquiry commands are the only inquiry commands that can be sent before the clock mode selection command; other inquiry commands must not be issued before the clock mode selection command. If commands are issued in an incorrect order, this LSI returns a response indicating a command error. Figure 20.9 shows an example of the procedure to use inquiry/selection host commands.

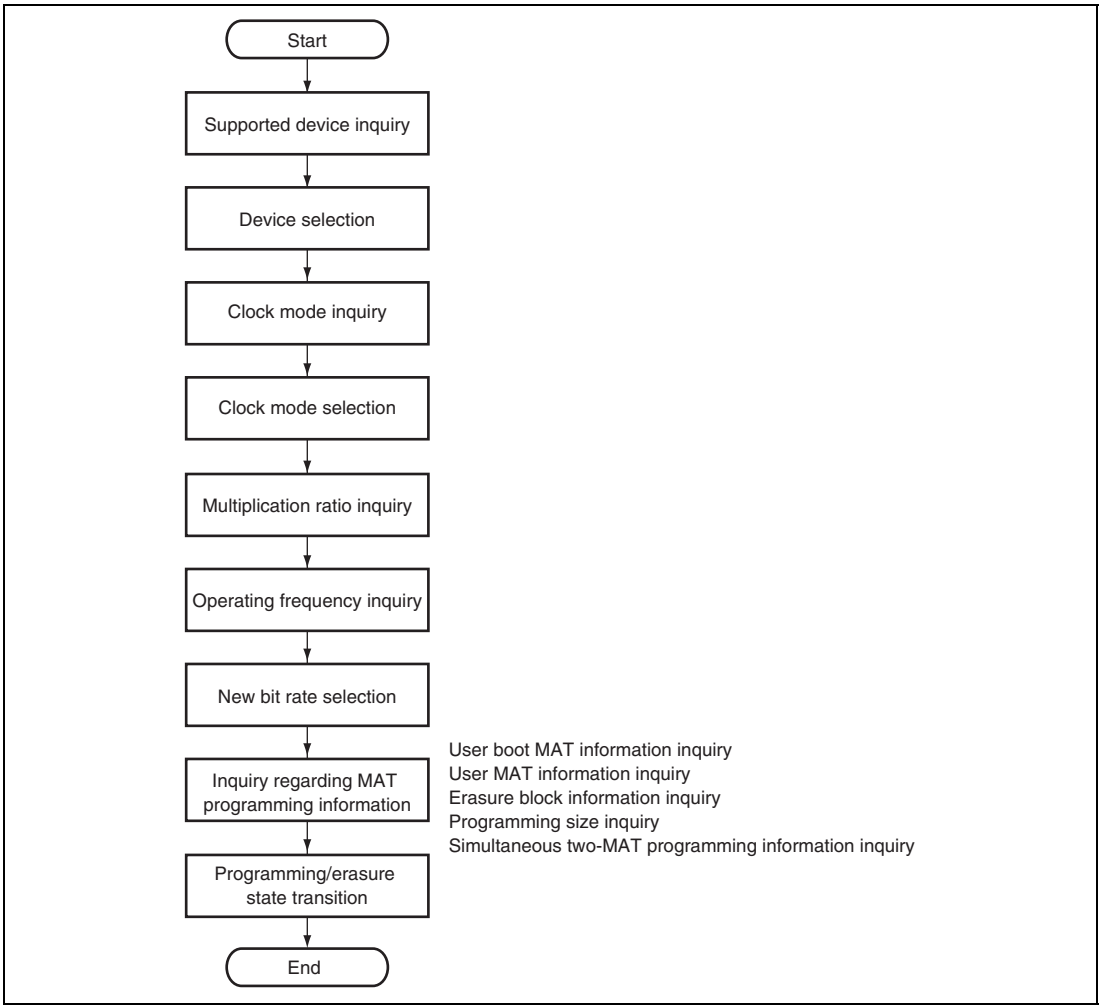


Figure 20.9 Example of Procedure to Use Inquiry/Selection Host Commands

Each host command is described in detail below. The "command" in the description indicates a command sent from the host to this LSI and the "response" indicates a response sent from this LSI to the host. The "checksum" is byte-size data calculated so that the sum of all bytes to be sent by this LSI becomes H'00.

(1) Supported Device Inquiry

In response to a supported device inquiry command sent from the host, this LSI returns the information concerning the devices supported by the embedded program for boot mode. If the supported device inquiry command comes after the host has selected a device, this LSI only returns the information concerning the selected device.

Command

H'20

Response

H'30	Size	Device count	
Character count	Device code		Product code
Character count	Device code		Product code
:	:		:
Character count	Device code		Product code
SUM			

[Legend]

Size (1 byte): Total number of bytes in the device count, character count, device code, and product code fields

Device count (1 byte): Number of device types supported by the embedded program for boot mode

Character count (1 byte): Number of characters included in the device code and product code fields

Device code (4 bytes): ASCII code for the product code of the embedded program

Product code (n bytes): ASCII code for the name of the supported MCU

SUM (1 byte): Checksum

(2) Device Selection

In response to a device selection command sent from the host, this LSI checks if the selected device is supported. When the selected device is supported, this LSI specifies this device as the device for use and returns a response (H'06). If the selected device is not supported or the sent command is illegal, this LSI returns an error response (H'90).

Even when H'01 has been returned as the number of supported devices in response to a supported device inquiry command, issue a device selection command to specify the device code that has been returned as the result of the inquiry.

Command	H'10	Size	Device code	SUM
Response	H'06			
Error response	H'90	Error		

[Legend]

Size (1 byte): Number of characters in the device code field (fixed at four)

Device code (4 bytes): ASCII code for the product name of the chip (one of the device codes returned in response to the supported device inquiry command)

SUM (1 byte): Checksum

Error (1 byte): Error code

H'11: Checksum error (illegal command)

H'21: Incorrect device code error

(3) Clock Mode Inquiry

In response to a clock mode inquiry command sent from the host, this LSI returns the supported clock modes. If the clock mode inquiry command comes after the host has selected a clock mode, this LSI only returns the information concerning the selected clock mode.

Command	H'21

Response				
	H'31	Size		
	Mode	Mode	. . .	Mode
	SUM			

[Legend]

Size (1 byte): Total number of bytes in the mode count and mode fields

Mode (1 byte): Supported clock mode (for example, H'01 indicates clock mode 1)

SUM (1 byte): Checksum

(4) Clock Mode Selection

In response to a clock mode selection command sent from the host, this LSI checks if the selected clock mode is supported. When the selected mode is supported, this LSI specifies this clock mode for use and returns a response (H'06). If the selected mode is not supported or the sent command is illegal, this LSI returns an error response (H'91).

Be sure to issue a clock mode selection command only after issuing a device selection command. Even when H'00 or H'01 has been returned as the number of supported clock modes in response to a clock mode inquiry command, issue a clock mode selection command to specify the clock mode that has been returned as the result of the inquiry.

Command	H'11	Size	Mode	SUM
Response	H'06			
Error response	H'91	Error		

[Legend]

Size (1 byte): Number of characters in the mode field (fixed at 1)

Mode (1 byte): Clock mode (one of the clock modes returned in response to the clock mode inquiry command)

SUM (1 byte): Checksum

Error (1 byte): Error code

H'11: Checksum error (illegal command)

H'22: Incorrect clock mode error

(5) Multiplication Ratio Inquiry

In response to a multiplication ratio inquiry command sent from the host, this LSI returns the clock types, the number of multiplication/division ratios, and the multiplication division ratios supported.

Command

H'22

Response

H'32	Size	Clock type count		
Multiplication ratio count	Multiplication ratio	Multiplication ratio	. . .	Multiplication ratio
Multiplication ratio count	Multiplication ratio	Multiplication ratio	. . .	Multiplication ratio
:	:	:	. . .	:
Multiplication ratio count	Multiplication ratio	Multiplication ratio	. . .	Multiplication ratio
SUM				

[Legend]

Size (1 byte): Total number of bytes in the clock type count, multiplication ratio count, and multiplication ratio fields

Clock type count (1 byte): Number of clock types (for example, H'02 indicates two clock types; that is, an internal clock and a peripheral clock)

Multiplication ratio count (1 byte): Number of supported multiplication/division ratios (for example, H'03 indicates that three multiplication ratios that are supported for the internal clock (x4, x6, and x8))

Multiplication ratio (1 byte): A positive value indicates a multiplication ratio (for example, H'04 = 4 = multiplication by 4)

A negative value indicates a division ratio (for example, H'FE = –2 = division by 2)

SUM (1 byte): Checksum

(6) Operating Frequency Inquiry

In response to an operating clock frequency inquiry command sent from the host, this LSI returns the minimum and maximum frequencies for each clock.

Command

H'23

Response

H'33	Size	Clock type count
Minimum frequency		Maximum frequency
Minimum frequency		Maximum frequency
:		:
Minimum frequency		Maximum frequency
SUM		

[Legend]

Size (1 byte): Total number of bytes in the clock type count, minimum frequency, and maximum frequency fields

Clock type count (1 byte): Number of clock types (for example, H'02 indicates two clock types; that is, an internal clock and a peripheral clock)

Minimum frequency (2 bytes): Minimum value of the operating frequency (for example, H'07D0 indicates 20.00 MHz).

This value should be calculated by multiplying the frequency value (MHz) to two decimal places by 100.

Maximum frequency (2 bytes): Maximum value of the operating frequency represented in the same format as the minimum frequency

SUM (1 byte): Checksum

(7) User Boot MAT Information Inquiry

This LSI does not have a user boot MAT, so user boot MAT information inquiry function is not supported. In response to a user boot MAT information inquiry command sent from the host, this LSI returns the number of user boot MATs and their addresses.

Command

H'24

Response

H'34	Size	MAT count
MAT start address		
MAT end address		
MAT start address		
MAT end address		
:		
MAT start address		
MAT end address		
SUM		

[Legend]

Size (1 byte): Total number of bytes in the MAT count, MAT start address, and MAT end address fields

MAT count (1 byte): Number of user boot MATs (consecutive areas are counted as one MAT)

MAT start address (4 bytes): Start address of a user boot MAT

MAT end address (4 bytes): End address of a user boot MAT

SUM (1 byte): Checksum

(8) User MAT Information Inquiry

In response to a user MAT information inquiry command sent from the host, this LSI returns the number of user MATs and their addresses.

Command	H'25		
Response	H'35	Size	MAT count
	MAT start address		
	MAT end address		
	MAT start address		
	MAT end address		
	:		
	MAT start address		
	MAT end address		
	SUM		

[Legend]

Size (1 byte): Total number of bytes in the MAT count, MAT start address, and MAT end address fields

MAT count (1 byte): Number of user MATs (consecutive areas are counted as one MAT)

MAT start address (4 bytes): Start address of a user MAT

MAT end address (4 bytes): End address of a user MAT

SUM (1 byte): Checksum

(9) Erasure Block Information Inquiry

In response to an erasure block information inquiry command sent from the host, this LSI returns the number of erasure blocks in the user MAT and their addresses.

Command	H'26		
Response	H'36	Size	Block count
	Block start address		
	Block end address		
	Block start address		
	Block end address		
	:		
	Block start address		
	Block end address		
	SUM		

[Legend]

Size (1 byte): Total number of bytes in the block count, block start address, and block end address fields

Block count (1 byte): Number of erasure blocks in the user MAT

Block start address (4 bytes): Start address of an erasure block

Block end address (4 bytes): End address of an erasure block

SUM (1 byte): Checksum

(10) Programming Size Inquiry

In response to a programming size inquiry command sent from the host, this LSI returns the programming size.

Command	H'27			
Response	H'37	Size	Programming size	SUM

[Legend]

Size (1 byte): Number of characters included in the programming size field (fixed at two)

Programming size (2 bytes): Programming unit (bytes)

SUM (1 byte): Checksum

(11) Simultaneous Two-MAT Programming Information Inquiry

In response to an inquiry from the host concerning simultaneous two-MAT programming information, this LSI returns whether two MATs can be programmed simultaneously and the start address of the available MATs. This LSI does not support simultaneous programming of two MATs.

Command	H'28

Response	H'38	Size	Mode
	First MAT start address		
	Second MAT start address		
	SUM		

[Legend]

Size (1 byte): Total number of bytes in the mode, first MAT start address, and second MAT start address fields (fixed at 5 in this LSI)

Mode (1 byte): Programming mode (H'01 in this LSI)
H'01: One-MAT programming
H'10: Simultaneous two-MAT programming

First MAT start address (1 byte): Start address of the first MAT (H'0000,0000 in this LSI)

Second MAT start address (1 byte): Start address of the second MAT (no data sent in this LSI)
No data is sent in one-MAT programming mode.

SUM (1 byte): Checksum

(12) New Bit Rate Selection

In response to a new bit rate selection command sent from the host, this LSI checks if the on-chip SCI can be set to the selected new bit rate. When the SCI can be set to the new bit rate, this LSI returns a response (H'06) and sets the SCI to the new bit rate. If the SCI cannot be set to the new bit rate or the sent command is illegal, this LSI returns an error response (H'BF). Upon reception of response H'06, the host waits for a one-bit period in the previous bit rate with which the new bit rate selection command has been sent, and then sets the host bit rate to the new one. After that, the host sends confirmation data (H'06) in the new bit rate, and this LSI returns a response (H'06) to the confirmation data.

Be sure to issue a new bit rate selection command only after a clock mode selection command.

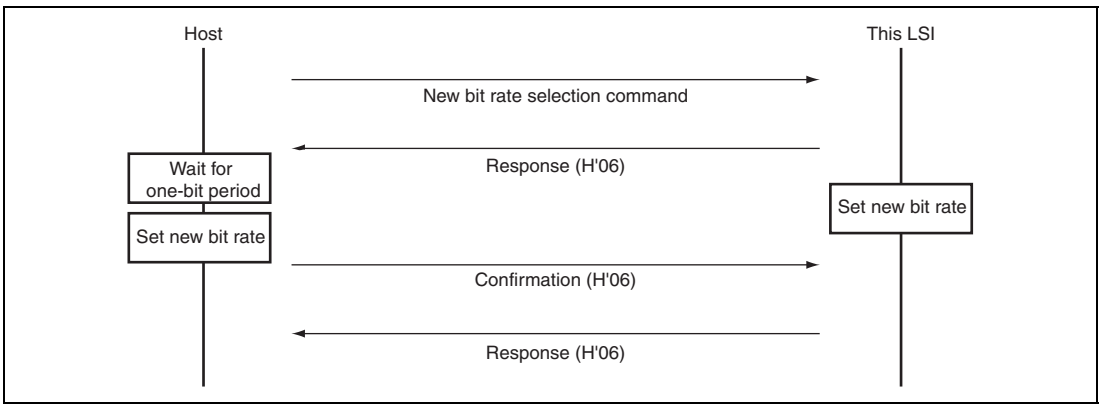


Figure 20.10 New Bit Rate Selection Sequence

Command	H'3F	Size	Bit rate	Input frequency
	Clock type count	Multiplication ratio 1	Multiplication ratio 2	
	SUM			
Response	H'06			
Error response	H'BF	Error		
Confirmation	H'06			
Response	H'06			

[Legend]

Size (1 byte): Total number of bytes in the bit rate, input frequency, clock type count, and multiplication ratio fields

Bit rate (2 bytes): New bit rate (for example, H'00C0 indicates 19200 bps)

1/100 of the new bit rate value should be specified.

Input frequency (2 bytes): Clock frequency input to this LSI (for example, H'03E8 indicates 10.00 MHz)

This value should be calculated by multiplying the input frequency value to two decimal places by 100.

Clock type count (1 byte): Number of clock types (for example, H'02 indicates two clock types; that is, an internal clock and a peripheral clock)

Multiplication ratio 1 (1 byte): Multiplication/division ratio of the input frequency to obtain the internal clock

A positive value indicates a multiplication ratio (for example, H'04 = 4 = multiplication by 4)

A negative value indicates a division ratio (for example, HFE = –2 = division by 2)

Multiplication 2 (1 byte): Multiplication/division ratio of the input frequency to obtain the peripheral clock.

This value is represented in the same format as multiplication ratio 1.

SUM (1 byte): Checksum

Error: Error code

H'11: Checksum error

H'24: Bit rate selection error

H'25: Input frequency error

H'26: Multiplication ratio error

H'27: Operating frequency error

- Bit rate selection error

  A bit rate selection error occurs when the bit rate selected through a new bit rate selection command cannot be set for the SCI of this LSI within an error of 4%. The bit rate error can be obtained by the following equation from the bit rate (B) selected through a new bit rate selection command, the input frequency (f_{EX}), multiplication ratio 2 ($M_{P\phi}$), the BRR_4 setting (N) in SCI4, and the CKS1 and CKS0 bit value (n) in SMR_4.

$$\text{Error (\%)} = \left\{ \frac{f_{EX} \times M_{P\phi} \times 10^6}{(N+1) \times B \times 64 \times 2^{n-1}} \right\} \times 100$$

- Input frequency error

  An input frequency error occurs when the input frequency specified through a new bit rate selection command is outside the range from the minimum to maximum input frequencies for the clock mode selected through a clock mode selection command.
- Multiplication ratio error

  A multiplication ratio error occurs when the multiplication ratio specified through a new bit rate selection command does not match the clock mode selected through a clock mode selection command. To check the selectable multiplication ratios, issue a multiplication ratio inquiry command.
- Operating frequency error

  An operating frequency error occurs when this LSI cannot operate at the operating frequencies selected through a new bit rate selection command. This LSI calculates the operating frequencies from the input frequency and multiplication ratios specified through a new bit rate selection command and checks if each calculated frequency is within the range from the minimum to maximum frequencies for the respective clock. To check the minimum and maximum operating frequencies for each clock, issue an operating clock frequency inquiry command.

(13) Programming/Erasure State Transition

In response to a programming/erasure state transition command sent from the host, this LSI erases the entire area of each of the user MAT, user boot MAT, and EEPROM data MAT. After completing erasure, this LSI returns a response (H'06) and waits for a programming/erasure host command. If this LSI has failed to complete erasure due to an error, it returns an error response (sends H'C0 and H'51 in that order).

Do not issue a programming/erasure state transition command before device selection, clock mode selection, and new bit rate selection commands.

Command	H'40	
Response	H'06	
Error response	H'C0	H'51

(14) Boot Program Status Inquiry

In response to a boot program status inquiry command sent from the host, this LSI returns its current status. The boot program status inquiry command can be issued in both inquiry/selection host command wait state and programming/erasure host command wait state.

Command	H'4F				
Response	H'5F	Size	Status	Error	Sum

[Legend]

Size (1 byte): Total number of bytes in the status and error fields (fixed at two)

Status (1 byte): Current status in this LSI (see table 20.7)

Error (1 byte): Error status in this LSI (see table 20.8)

SUM (1 byte): Checksum

Table 20.7 Status Code

Code	Description
H'11	Waiting for device selection
H'12	Waiting for clock mode selection
H'13	Waiting for bit rate selection
H'1F	Waiting for transition to programming/erasure host command wait state (bit rate has been selected)
H'31	Erasing the user MAT
H'3F	Waiting for a programming/erasure host command
H'4F	Waiting for reception of programming data
H'5F	Waiting for erasure block selection

Table 20.8 Error Code

Code	Description
H'00	No error
H'11	Checksum error
H'21	Incorrect device code error
H'22	Incorrect clock mode error
H'24	Bit rate selection error
H'25	Input frequency error
H'26	Multiplication ratio error
H'27	Operating frequency error
H'29	Block number error
H'2A	Address error
H'2B	Data size error
H'51	Erasure error
H'52	Incomplete erasure error
H'53	Programming error
H'54	Selection error
H'80	Command error
H'FF	Bit rate adjustment verification error

20.5.5 Programming/Erasing Host Command Wait State

Table 20.9 shows the host commands available in programming/erasure host command wait state.

Table 20.9 Programming/Erasure Host Commands

Host Command Name	Function
User boot MAT programming selection*	This LSI selects the program for user boot MAT programming.
User MAT programming selection	This LSI selects the program for user MAT programming.
Simultaneous two-user MAT programming selection	This LSI selects the program for simultaneous two-user MAT programming.
128-byte programming	Programs 128 bytes of data.
Erasure selection	This LSI selects the erasure program.
Block erasure	Erases block data.
Memory read	Reads data from memory.
User boot MAT checksum*	Performs checksum verification for the user boot MAT.
User MAT checksum	Performs checksum verification for the user MAT.
User boot MAT blank check*	Checks whether the user boot MAT is blank.
Reading lock bit status	Reads the lock bit value
Programming lock bit	Programs the lock bit
Enabling lock bit	Enables the lock bit
Disabling lock bit	Disables the lock bit
User MAT blank check	Checks whether the user MAT is blank.
Boot program status inquiry	Inquires regarding the state of this LSI.

Note: * This LSI does not have user boot MAT.

If the host has sent an undefined command, this LSI returns a response indicating a command error. For the format of this response, see section 20.5.4, Inquiry/Selection Host Command Wait State.

To program the flash memory, issue a user MAT selection command and then a 128-byte programming command from the host. After the host has transmitted a programming selection command, this LSI enters programming data wait state (see section 20.5.2, State Transition in Boot Mode). In response to a 128-byte programming command sent from the host in this state, this LSI starts programming the flash memory. When the host sends a 128-byte programming command specifying H'FFFFFFFF as the programming start address, this LSI detects it as the end of programming and enters programming/erasure host command wait state.

To erase the flash memory, issue an erasure selection command and then a block erasure command from the host. After the host has transmitted an erasure selection command, this LSI enters erasure block selection wait state (see section 20.5.2, State Transition in Boot Mode). In response to a block erasure command sent from the host in this state, this LSI erases the specified block in the flash memory. When the host sends a block erasure command specifying H'FF as the block number, this LSI detects it as the end of erasure and enters programming/erasure host command wait state.

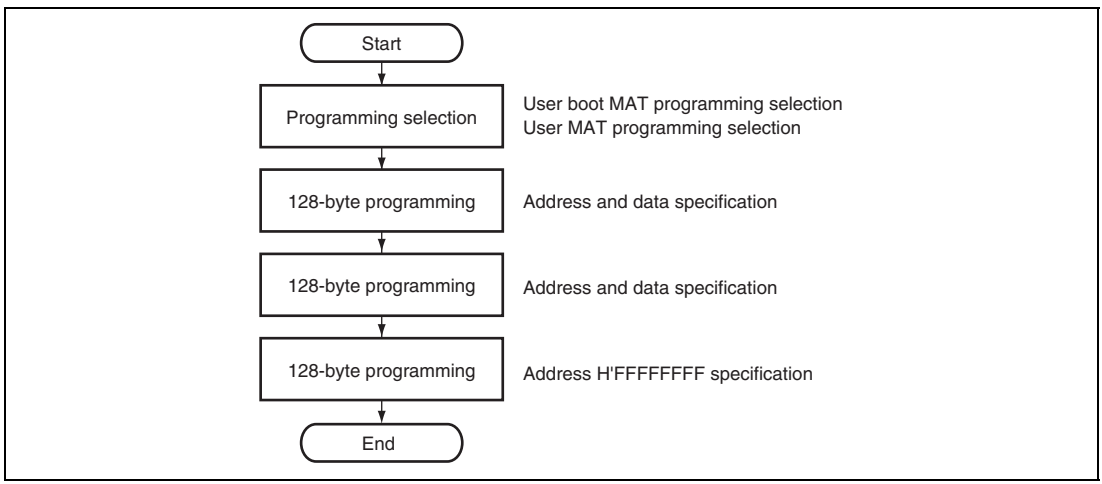


Figure 20.11 Procedure for Flash Memory Programming in Boot Mode

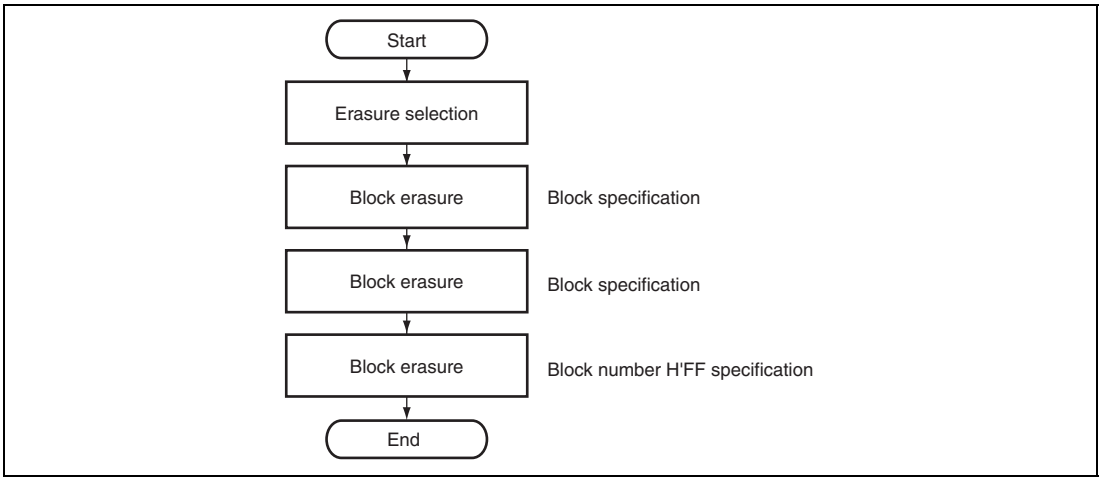


Figure 20.12 Procedure for Flash Memory Erasure in Boot Mode

Each host command is described in detail below. The "command" in the description indicates a command sent from the host to this LSI and the "response" indicates a response sent from this LSI to the host. The "checksum" is byte-size data calculated so that the sum of all bytes to be sent by this LSI becomes H'00.

(1) User Boot MAT Programming Selection

This LSI does not have a user boot MAT, so user boot MAT programming function is not supported. In response to a user boot MAT programming selection command sent from the host, this LSI selects the program for user boot MAT programming and waits for programming data.

Command	H'42
Response	H'06

(2) User MAT Programming Selection

In response to a user MAT programming selection command sent from the host, this LSI selects the program for user MAT programming and waits for programming data.

Command	H'43
Response	H'06

(3) Simultaneous Two-User MAT Programming Selection

This LSI does not provide the simultaneous two-user MAT programming function. In response to a simultaneous two-user MAT programming selection command sent from the host, this LSI returns a command error response (sends H'80 and H'44 in that order).

Command	H'44	
Response	H'80	H'44

(4) 128-Byte Programming

In response to a 128-byte programming command sent from the host, this LSI programs the flash memory. After completing flash memory programming successfully, this LSI returns a response (H'06). If an error has occurred during flash memory programming, this LSI returns an error response (H'D0).

Command	H'50	Programming Address		
	Data	Data	. . .	Data
	SUM			
Response	H'06			
Error response	H'D0	Error		

[Legend]

Programming address (4 bytes): Target address of programming
To program the flash memory, a 128-byte boundary address should be specified.
To terminate programming, H'FFFFFFFF should be specified.

Data (128 bytes): Programming data
H'FF should be specified for the bytes that do not need to be programmed.
When terminating programming, no data needs to be specified (only the programming address and SUM should be sent in that order).

SUM (1 byte): Checksum

Error (1 byte): Error code
H'11: Checksum error
H'2A: Address error (the specified address is not in the target MAT)
H'53: Programming cannot be done due to a programming error

(5) Erasure Selection

In response to an erasure selection command sent from the host, this LSI selects the erasure program and waits for erasure block specification.

Command	H'48
Response	H'06

(6) Block Erasure

In response to a block erasure command sent from the host, this LSI erases the flash memory. After completing flash memory erasure successfully, this LSI returns a response (H'06). If an error has occurred during flash memory erasure, this LSI returns an error response (H'D8).

Command	H'58	Size	Block	SUM
Response	H'06			
Error response	H'D8	Error		

[Legend]
Size (1 byte): Number of bytes in the block specification field (fixed at 1)
Block (1 byte): Block number whose data is to be erased
To terminate erasure, H'FF should be specified.
SUM (1 byte): Checksum
Error (1 byte): Error code
H'11: Checksum error
H'29: Block number error (an incorrect block number is specified)
H'51: Erasure cannot be done due to an erasure error

(7) Memory Read

In response to a memory read command sent from the host, this LSI reads data from the flash memory. After completing flash memory reading, this LSI returns the data stored in the address specified by the memory read command. If this LSI has failed to read the flash memory, this LSI returns an error response (H'D2).

Command	H'52	Size	Area	Read start address	
	Reading size				SUM

Response	H'52	Reading size		
	Data	Data	. . .	Data
	SUM			

Error response	H'D2	Error

[Legend]

Size (1 byte): Total number of bytes in the area, read start address, and reading size fields

Area (1 byte): Target MAT to be read

H'00: User boot MAT*

H'01: User MAT

[Note] * This LSI does not have user boot MAT.

Read start address (4 bytes): Start address of the area to be read

Reading size (4 bytes): Size of data to be read (bytes)

SUM (1 byte): Checksum

Data (1 byte): Data read from the flash memory

Error (1 byte): Error code

H'11: Checksum error

H'2A: Address error

- The value specified for area selection is neither H'00 nor H'01.
- The specified read start address is outside the selected MAT.

H'2B: Data size error

- H'00 is specified for the reading size.
- The reading size is larger than the MAT.
- The end address calculated from the read start address and the reading size is outside the selected MAT.

(8) User Boot MAT Checksum

This LSI does not have a user boot MAT, so user boot MAT sum check function is not supported. In response to a user boot MAT checksum command sent from the host, this LSI sums the user boot MAT data in byte units and returns the result (checksum).

Command	H'4A

Response	H'5A	Size	MAT checksum	SUM

[Legend]

Size (1 byte): Number of bytes in the MAT checksum field (fixed at 4)

MAT checksum (4 bytes): Checksum of the user boot MAT data

SUM (1 byte): Checksum (for the response data)

(9) User MAT Checksum

In response to a user MAT checksum command sent from the host, this LSI sums the user MAT data in byte units and returns the result (checksum).

Command	H'4B

Response	H'5B	Size	MAT checksum	SUM

[Legend]

Size (1 byte): Number of bytes in the MAT checksum field (fixed at 4)

MAT checksum (4 bytes): Checksum of the user MAT data

The user MAT also stores the key code for debugging function authentication. Note that the checksum includes this key code value.

SUM (1 byte): Checksum (for the response data)

(10) User Boot MAT Blank Check

This LSI does not have a user boot MAT, so user boot MAT blank check function is not supported. In response to a user boot MAT blank check command sent from the host, this LSI checks whether the user boot MAT is completely erased. When the user boot MAT is completely erased, this LSI returns a response (H'06). If the user boot MAT has an unerased area, this LSI returns an error response (sends H'CC and H'52 in that order).

Command	H'4C	
Response	H'06	
Error response	H'CC	H'52

(11) User MAT Blank Check

In response to a user MAT blank check command sent from the host, this LSI checks whether the user MAT is completely erased. When the user MAT is completely erased, this LSI returns a response (H'06). If the user MAT has an unerased area, this LSI returns an error response (sends H'CD and H'52 in that order).

Command	H'4D	
Response	H'06	
Error response	H'CD	H'52

(12) Read Lock Bit Status

In response to a read lock bit status command sent from the host, this LSI reads data from the lock bit. After completing the lock bit reading, this LSI returns the data stored in the address specified by the read lock bit status command. If this LSI has failed to read the lock bit, this LSI returns an error response (H'F1).

Command	H'71	Size	Area	Medium address	Upper address	SUM
Response	Status					
Error response	H'F1	Error				

[Legend]

Size (1 byte): Total number of bytes in the area, medium address, and upper address (fixed at 3 in this LSI)

Area (1 byte): Target MAT to be read
H'00: User boot MAT*
H'01: User MAT
[Note] * This LSI does not have user boot MAT.

Medium address (1 byte): Medium address at the end of the specified address (8 to 15 bits)

Upper address (1 byte): Upper address at the end of the specified address (16 to 23 bits)

SUM (1 byte): Checksum

Status (1 byte): Bit 6 locked at "0"
Bit 6 unlocked at "1"

Error (1 byte): Error code
H'11: Checksum error
H'2A: Address error (the specified address is not in the target MAT)

(13) Lock Bit Program

In response to a lock bit program command sent from the host, this LSI writes to a lock bit and locks the specified block. After completing the lock bit blocking, this LSI returns a response (H'06). If this LSI has failed to lock, this LSI returns an error response (H'F7).

Command	H'77	Size	Area	Medium address	Upper address	H'D0	SUM

Response	H'06

Error response	H'F7	Error

[Legend]

Size (1 byte): Total number of bytes in the area, medium address, upper address, and fixed code (H'D0) (fixed at 4 in this LSI)

Area (1 byte): Target MAT to be locked
H'00: User boot MAT*
H'01: User MAT
[Note] * This LSI does not have user boot MAT.

Medium address (1 byte): Medium address at the end of the specified address (8 to 15 bits)

Upper address (1 byte): Upper address at the end of the specified address (16 to 23 bits)

SUM (1 byte): Checksum

Error (1 byte): Error code
H'11: Checksum error
H'28: Address error (the specified address is not in the target MAT)
H'53: Locking cannot be done due to a programming error

(14) Lock Bit Enable

In response to a lock bit enable command sent from the host, this LSI enables a lock bit.

Command	H'7A

Response	H'06

(15) Lock Bit Disable

In response to a lock bit disable command sent from the host, this LSI disables a lock bit.

Command	H'75
Response	H'06

(16) Boot Program Status Inquiry

For details, refer to section 20.5.4, Inquiry/Selection Host Command Wait State.

20.6 User Program Mode

20.6.1 FCU Command List

To program or erase the user MAT in user program mode, issue FCU commands to the FCU. Table 20.10 is a list of FCU commands for flash memory programming and erasure.

Table 20.10 FCU Command List (Flash Memory-Related Commands)

Command	Function
Normal mode transition	Moves to the normal mode (see section 20.6.2, Conditions for FCU Command Acceptance)
Status read mode transition	Moves to the status read mode (see section 20.6.2, Conditions for FCU Command Acceptance)
Lock bit read mode transition (lock bit read 1)	Moves to the lock bit read mode (see section 20.6.2, Conditions for FCU Command Acceptance)
Flash clock notification	Specifies the frequency of the flash clock
Program	Programs flash memory (in 128-byte units)
Block erase	Erases flash memory (in block units; erasing the lock bit)
P/E suspend	Suspends programming or erasure
P/E resume	Resumes programming or erasure
Status register clear	Clears the ILGLERR, ERSERR, and PRGERR bits in FSTATR0 and cancels the command-locked state
Lock bit read 2	Reads the lock bit of a specified erasure block (updates the FLOCKST bit in FSTATR1 to reflect the lock bit state)
Lock bit program	Writes to the lock bit of a specified erasure block

FCU commands other than lock bit read 2 and lock bit program commands are also used in EEPEOM programming/erasure. If a lock bit command 2 is issued to EEPROM, blank checking of EEPROM is executed. If a lock bit program command is issued to EEPROM, an error occurs because it is detected as an illegal command (see section 21, Data Flash (EEPROM)).

Issuance of a command to the FCU takes place as a sequence of write access to a flash memory program/erase address via the peripheral bus. Table 20.11 shows the FCU command formats. Performing peripheral-bus write access as shown in table 20.11 under specified conditions starts each command processing in the FCU. For the conditions for FCU command acceptance, refer to section 20.6.2, Conditions for FCU Command Acceptance. For details of each FCU command, refer to section 20.6.3, FCU Command Usage.

When H'71 is sent in the first cycle of an FCU command while the FRDMD bit is 0 (memory area read mode), the FCU accepts the lock bit read mode transition command (lock bit read 1). When a flash memory program/erase address is read through the peripheral bus after transition to the lock bit read mode, the FCU copies the lock bit of the erasure block corresponding to the accessed address into all bits in the read data. When H'71 is sent in the first cycle of the FCU command while the FRDMD bit is 1 (register read mode), the FCU waits for the second-cycle data (H'D0) of the lock bit read 2 command. When H'D0 is written to a flash memory program/erase address through the peripheral bus in this state, the FCU copies the lock bit of the erasure block corresponding to the access address into the FLOCKST bit in FSTATR1.

There are two suspending modes to be initiated by the P/E suspend command; the suspension-priority mode and erasure-priority mode. For details of each mode, refer to section 20.6.4, Suspending Operation.

Table 20.11 FCU Command Format

Command	Number of Bus Cycles	First Cycle		Second Cycle		Third Cycle		Fourth and Fifth Cycles		Sixth Cycle		Seventh to 66th Cycles		67th Cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Normal mode transition	1	RA	H'FF	—	—	—	—	—	—	—	—	—	—	—	—
Status read mode transition	1	RA	H'70	—	—	—	—	—	—	—	—	—	—	—	—
Lock bit read mode transition (lock bit read 1)	1	RA	H'71	—	—	—	—	—	—	—	—	—	—	—	—
Flash clock notification	6	RA	H'E9	RA	H'03	WA	H'0F0F	RA	H'0F0F	RA	H'D0	—	—	—	—
Program	67	RA	H'E8	RA	H'40	WA	WD1	RA	WDn	RA	WDn	RA	WDn	RA	H'D0
Block erase	2	RA	H'20	BA	H'D0	—	—	—	—	—	—	—	—	—	—
P/E suspend	1	RA	H'B0	—	—	—	—	—	—	—	—	—	—	—	—
P/E resume	1	RA	H'D0	—	—	—	—	—	—	—	—	—	—	—	—
Status register clear	1	RA	H'50	—	—	—	—	—	—	—	—	—	—	—	—
Lock bit read 2	2	RA	H'71	BA	H'D0	—	—	—	—	—	—	—	—	—	—
Lock bit program	2	RA	H'77	BA	H'D0	—	—	—	—	—	—	—	—	—	—

[Legend]

RA: Flash memory program/erase address
When FENTRY0 is 1: An address in the range from H'D00000 to H'D7FFFF

WA: Start address of flash memory programming
Start address of 128-byte programming data

BA: Flash memory erasure block address
An address in the target erasure block (specified by the flash memory program/erase address)

WDn: n-th word of programming data (n = 1 to 64)

20.6.2 Conditions for FCU Command Acceptance

The FCU determines whether to accept a command depending on the FCU mode or status. Figure 20.13 is an FCU mode transition diagram.

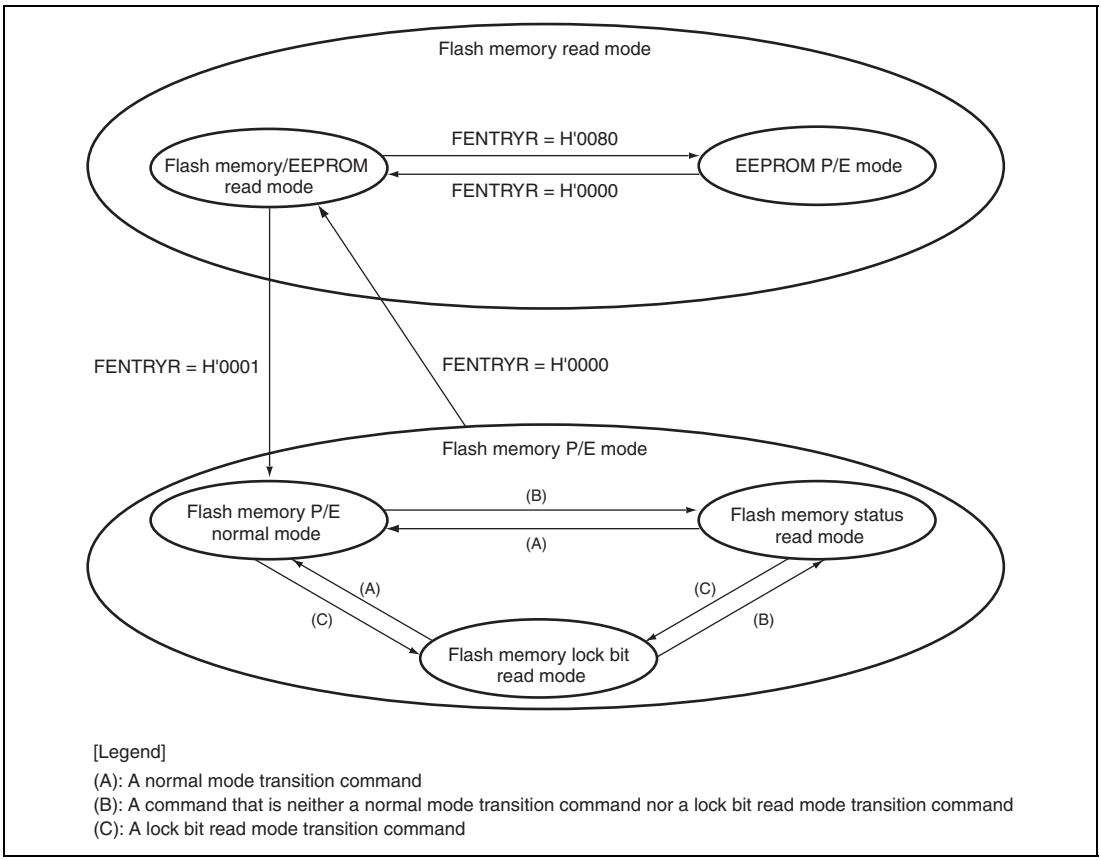


Figure 20.13 FCU Mode Transition Diagram (Flash Memory-Related Modes)

(1) Flash Memory Read Mode

- Flash memory/EEPROM read mode

  The flash memory and EEPROM can be read through the internal bus and peripheral bus, respectively, at a high speed. The FCU does not accept commands. The FCU enters this mode when the FENTRY0 bit is cleared to 0 and the FENTRYD bit to 0 in FENTRYR.

- EEPROM P/E mode

  The flash memory can be read through the internal bus at a high speed. The FCU accepts commands for EEPROM, but does not accept commands for flash memory. The FCU enters this mode when the FENTRY0 bit is cleared to 0 and the FENTRYD bit to 1. For details of the EEPROM P/E mode, refer to section 20.6.2, Conditions for FCU Command Acceptance.

(2) Flash Memory P/E Mode

- Flash memory P/E normal mode

  The FCU enters this mode when the FENTRYD bit is cleared to 0 and the FENTRY0 bit is set to 1 in flash memory read mode, or when a normal mode transition command is accepted in flash memory P/E mode. Table 20.12 shows the commands that can be accepted in this mode. High-speed read operation is not available for the flash memory. If an address in the range from H'D00000 to H'D7FFFF is read through the peripheral bus while the FENTRY0 bit is set to 1, a flash memory access error occurs and the FCU enters the command-locked state (see section 20.8.3, Error Protection).

- Flash memory status read mode

  The FCU enters this mode when the FCU accepts a command that is neither a normal mode transition command nor a lock bit read mode transition command in flash memory P/E mode. The flash memory status read mode includes the state in which the FRDY bit in FSTATR0 is 0 and the command-locked state after an error has occurred. Table 20.12 shows the commands that can be accepted in this mode. High-speed read operation is not available for the flash memory. The FENTRYR value is the same as that in flash memory P/E normal mode. If an address in the range from H'D00000 to H'D7FFFF is read through the peripheral bus while the FENTRY0 bit is set to 1, the FSTATR0 value is read.

- Flash memory lock bit read mode

  The FCU enters this mode when the FCU accepts a lock bit read mode transition command in flash memory P/E mode. Table 20.12 shows the commands that can be accepted in this mode. High-speed read operation is not available for the flash memory. The FENTRYR value is the same as that in flash memory P/E normal mode. If an address in the range from H'D00000 to H'D7FFFF is read through the peripheral bus while the FENTRY0 bit is set to 1, the lock bit value of the target erasure block is returned through all bits in the read data.

Table 20.12 shows the acceptable commands in each FCU mode/state of flash memory P/E mode. When a command that cannot be accepted is issued, the FCU enters the command-locked state (see section 20.8.3, Error Protection).

To make sure that the FCU accepts a command, enter the mode in which the FCU can accept the target command, check the values of bits FRDY, ILGLERR, ERSERR, and PRGERR in FSTATR0 and bits FCUERR, FRDTCT, and FRCRCT in FSTATR1, and then issue the target FCU command. The CMDLK bit in FASTAT holds a value obtained by ORing the IRGLERR, ERSERR, and PRGERR bits in FSTATR0 and bits FCUERR, FRDTCT, and FRCRCT in FSTATR1. Accordingly, whether any error has occurred or not in the FCU can be checked from the value of the CMDLK bit.

In table 20.12, whether an error has occurred or not is determined from the CMDLK bit. (Note that when the FRDCLE and FRCCLE bits in the FRAMECCR register are set to 0, an FIFE interrupt by the CMDLK bit does not occur.)

While the programming/erasure processing, programming/erasure suspension processing, or lock bit read 2 processing is in progress, the FRDY bit in FSTATR0 is 0. While the FRDY bit is 0, the P/E suspend command can be accepted only when the SUSRDY bit in FSTATR0 is 1.

In table 20.12, the values of bits ERSSPD, PRGSPD, and FRDY are indicated as “0/1” to make the table simpler. The ERSSPD bit is 1 during erasure-suspension processing and 0 during programming-suspension processing. The PRGSPD bit is 1 during programming-suspension processing and 0 during erasure-suspension processing. In a command-locked state, the FRDY bit retains the value before the command-locked state is entered.

Table 20.12 FCU Modes/States and Acceptable Commands

Item	P/E Normal Mode			Status Read Mode									Lock Bit Read Mode		
	Programming-Suspended	Erasure-Suspended	Other State	Programming/Erasure Processing	Programming Processing while Erasure-Suspended	Programming/Erasure Suspension Processing	Lock Bit Read 2 Processing	Programming-Suspended	Erasure-Suspended	Command Locked (FRDY = 0)	Command Locked (FRDY = 1)	Other State	Programming-Suspended	Erasure-Suspended	Other State
FRDY bit in FSTATR0	1	1	1	0	0	0	0	1	1	0	1	1	1	1	1
SUSRDY bit in FSTATR0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
ERSSPD bit in FSTATR0	0	1	0	0	1	0/1	0/1	0	1	0/1	0/1	0	0	1	0
PRGSPD bit in FSTATR0	1	0	0	0	0	0/1	0/1	1	0	0/1	0/1	0	1	0	0
CMDLK bit in FASTAT	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
Normal mode transition	A	A	A	×	×	×	×	A	A	×	×	A	A	A	A
Status read mode transition	A	A	A	×	×	×	×	A	A	×	×	A	A	A	A
Lock bit read mode transition (lock bit read 1)	A	A	A	×	×	×	×	A	A	×	×	A	A	A	A
Flash clock notification	×	×	A	×	×	×	×	×	×	×	×	A	×	×	A
Program	×	*	A	×	×	×	×	×	*	×	×	A	×	*	A
Block erase	×	×	A	×	×	×	×	×	×	×	×	A	×	×	A
P/E suspend	×	×	×	A	×	×	×	×	×	×	×	×	×	×	×
P/E resume	A	A	×	×	×	×	×	A	A	×	×	×	A	A	×
Status register clear	A	A	A	×	×	×	×	A	A	×	A	A	A	A	A
Lock bit read 2	A	A	A	×	×	×	×	A	A	×	×	A	A	A	A
Lock bit program	×	*	A	×	×	×	×	×	*	×	×	A	×	*	A

[Legend]

A: Acceptable

*: Only programming is acceptable for the areas other than the erasure-suspended block

×: Not acceptable

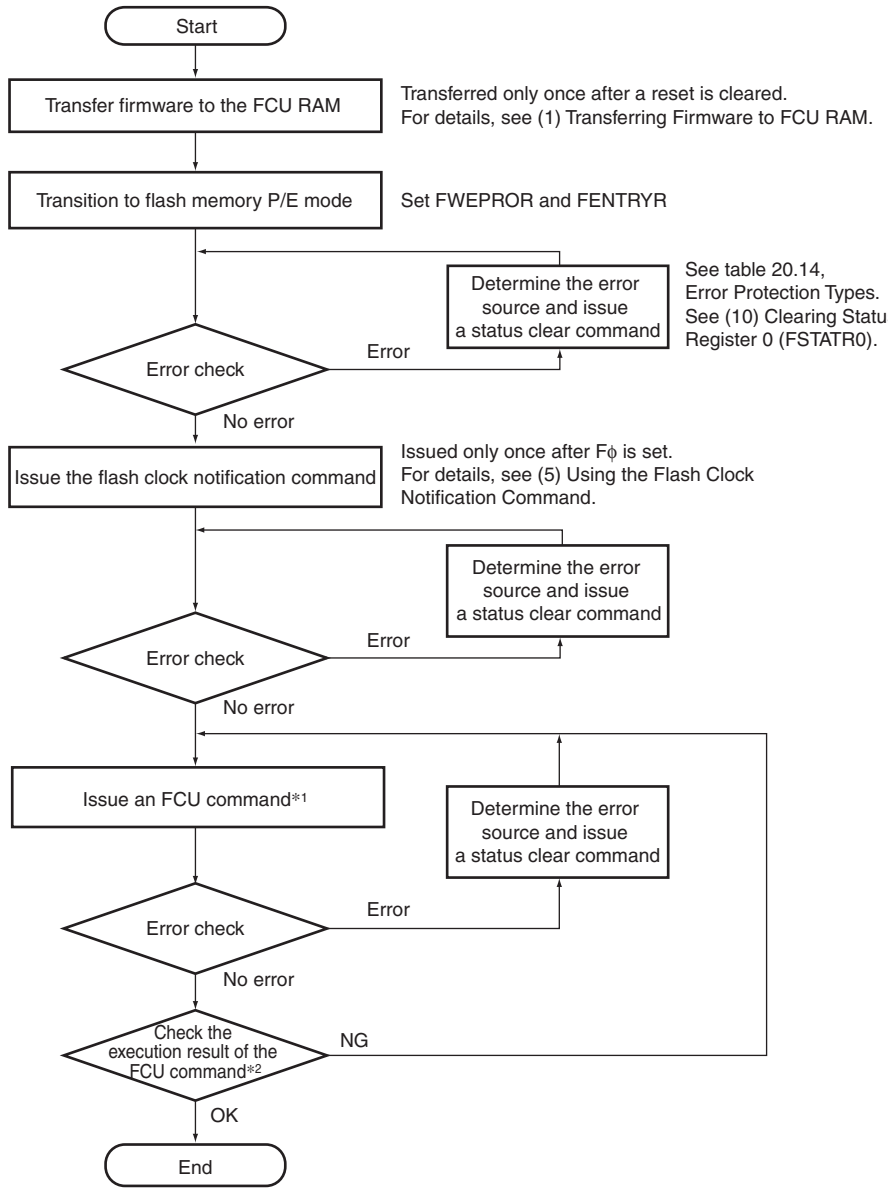
20.6.3 FCU Command Usage

This section shows examples of user processing procedures for firmware transfer to the FCU RAM and the issuing of FCU commands. In some procedures given in this section, the FCU state is not checked before an FCU command is issued but the command result is checked before the processing is completed. To make sure that the FCU accepts a command, check the FCU state before starting processing (see section 20.6.2, Conditions for FCU Command Acceptance).

In the processing procedures given in this section, the command processing state and error state of the FCU are checked from bits FRDY, ILGLERR, ERSERR, PRGERR, SUSRDY, ERSSPD, and PRGSPD in FSTATR0 and bits FCUERR, FRDTCT, and FRCRCT in FSTATR1. Since the FSTATR0 and FSTATR1 registers can be read simultaneously by a word access, only a single access is required in checking of the FCU state. On the other hand, when the FRDY bit in FSTATR0 and the CMDLK bit in FASTAT are used to check the FCU state, two register accesses are required but whether any error has occurred or not can be determined simply from the CMDLK bit.

If the FCUERR, FRDTCT, or FRCRCT bit is set to 1 while the FCU is processing a command and the FCU is thus placed in the command-locked state, the FRDY bit holds 0. Since the FCU stops processing in the command-locked state, the FRDY bit will not be changed from 0 to 1. If the FRDY bit stays 0 for a period longer than the programming/erasure time or suspend delay time (see section 25, Electrical Characteristics), it is likely that an abnormal condition has occurred: for example, FCU processing has been stopped in the command-locked state. In such cases, reset the FCU for initialization. When command processing is completed with the FRDY bit set to 1, the FCUERR, FRDTCT, and FRCRCT bits are always 0. Accordingly, using bits ILGLERR, ERSERR, and PRGERR is enough for checking the error state after completion of command processing.

A general procedure of the FCU command processing is shown below. For details, refer to the descriptions in the following pages.



Notes: 1. See table 20.10, FCU Command List (Flash Memory-Related Commands).
2. To check the result of programming or erasure processing, enter flash memory read mode and read data from the flash memory (see (2) Entering Flash Memory P/E Mode).

Figure 20.14 General Procedure of FCU Command Processing

(1) Transferring Firmware to the FCU RAM

To use FCU commands, the FCU firmware must be stored in the FCU RAM. When this LSI is started, the FCU firmware is not stored in the FCU RAM; copy the firmware stored in the FCU firmware area to the FCU RAM. If the FCUERR, FRDTCT, or FRCRCT bit in FSTATR1 is 1, the firmware stored in the FCU RAM may have been damaged; reset the FCU and copy the FCU firmware again in this case.

Figure 20.15 shows the procedure for firmware transfer to the FCU RAM. Before writing data to the FCU RAM, clear FENTRYR to H'0000 to stop the FCU. For details on the DMAC settings, refer to section 7, DMA Controller (DMAC).

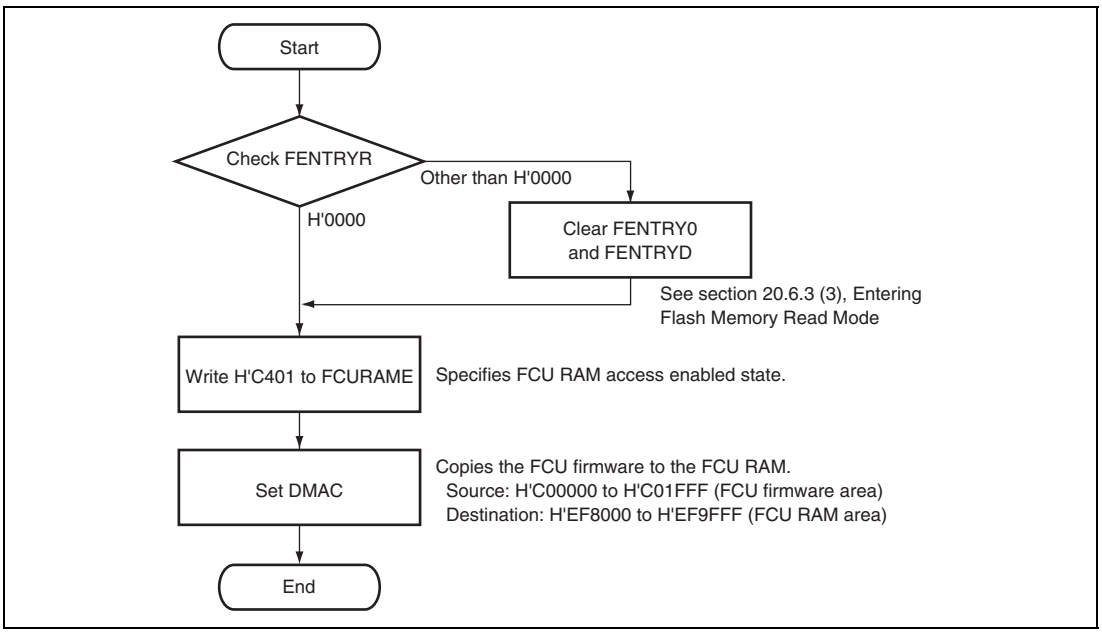


Figure 20.15 Procedure for Firmware Transfer to FCU RAM

(2) Entering Flash Memory P/E Mode

To execute flash memory-related FCU commands, set the FENTRY0 bit in FENTRYR appropriately to make the FCU enter flash memory P/E mode (see section 20.6.2, Conditions for FCU Command Acceptance). To execute FCU commands for flash memory (read addresses: H'000000 to H'07FFFF; program/erase addresses: H'D00000 to H'D7FFFF), set FENTRY0 to 1. For the conditions for writing to the FENTRY0 bit, refer to section 20.3.9, Flash P/E Mode Entry Register (FENTRYR).

After a transition from flash memory read mode to flash memory P/E mode, the FCU enters flash memory P/E normal mode.

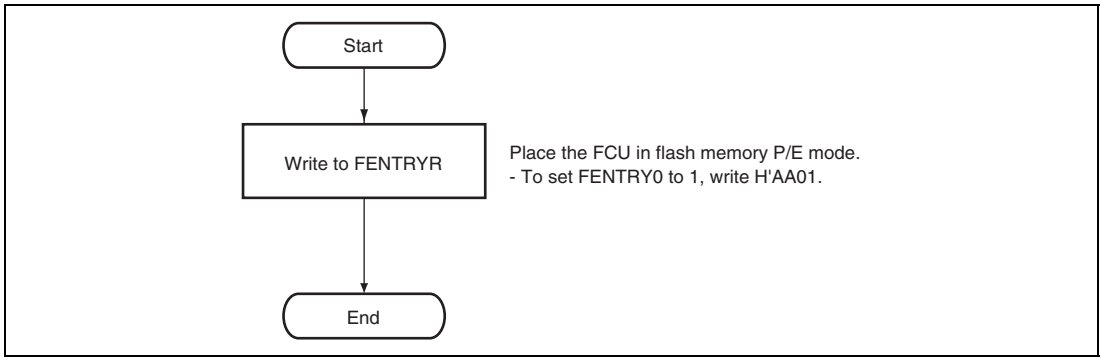


Figure 20.16 Procedure for Transition to Flash Memory P/E Mode

(3) Entering Flash Memory Read Mode

To enable high-speed flash memory read access through the internal bus, clear the FENTRY0 bit in FENTRYR to make the FCU enter flash memory read mode (see section 20.6.2, Conditions for FCU Command Acceptance). A transition from flash memory P/E mode to flash memory read mode must be made while no FCU error has been detected after FCU command processing is completed.

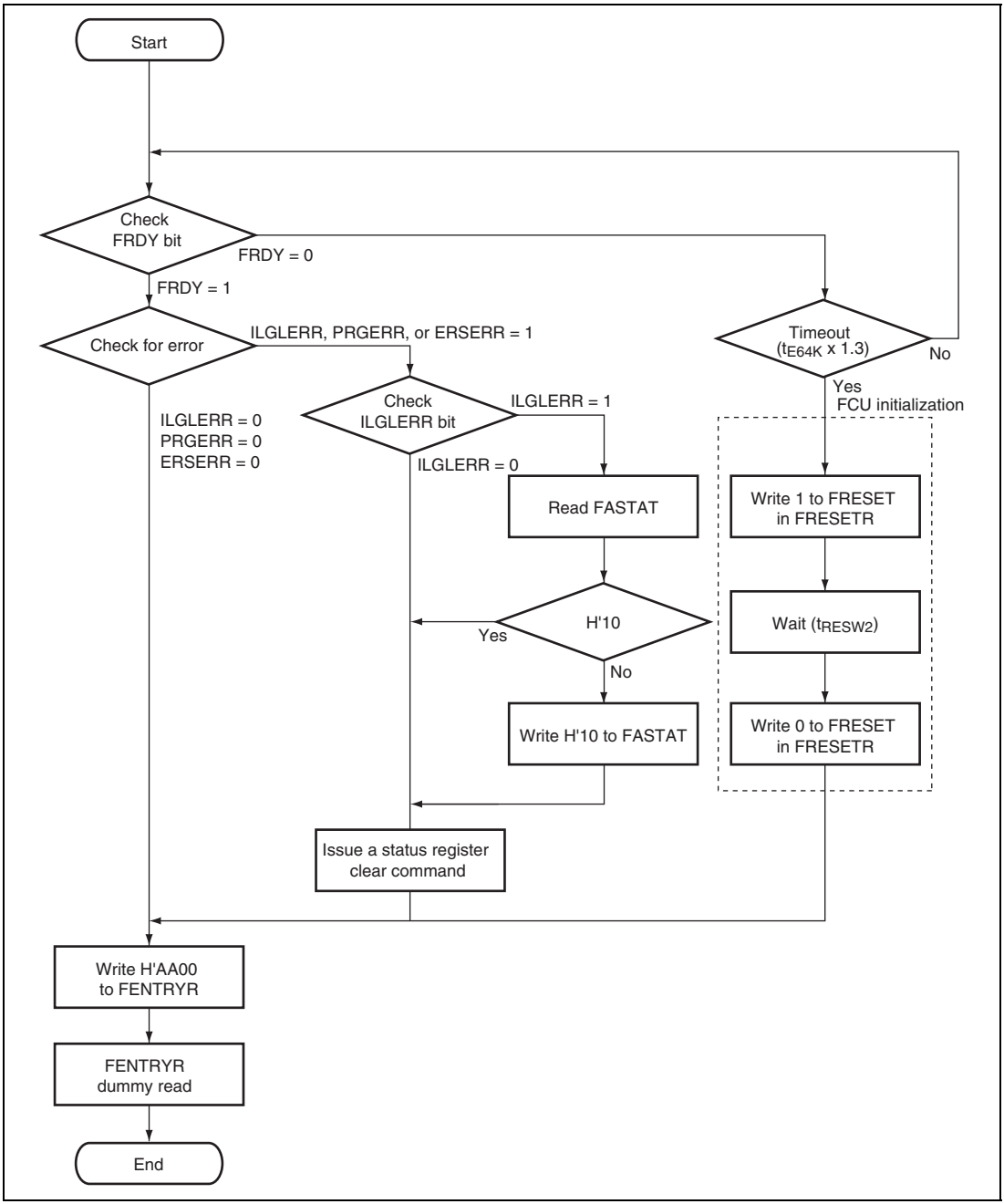


Figure 20.17 Procedure for Transition to Flash Memory Read Mode

(4) Using Flash Memory P/E Normal Mode Transition Command

The FCU can enter flash memory P/E normal mode in two ways: one is to set FENTRYR in flash memory read mode (see section 20.6.3 (1), Transferring Firmware to the FCU RAM) and the other is to issue a normal mode transition command in flash memory P/E mode (figure 20.18). The status read mode transition command and the lock bit read mode transition command can be used in the same way as the normal mode transition command.

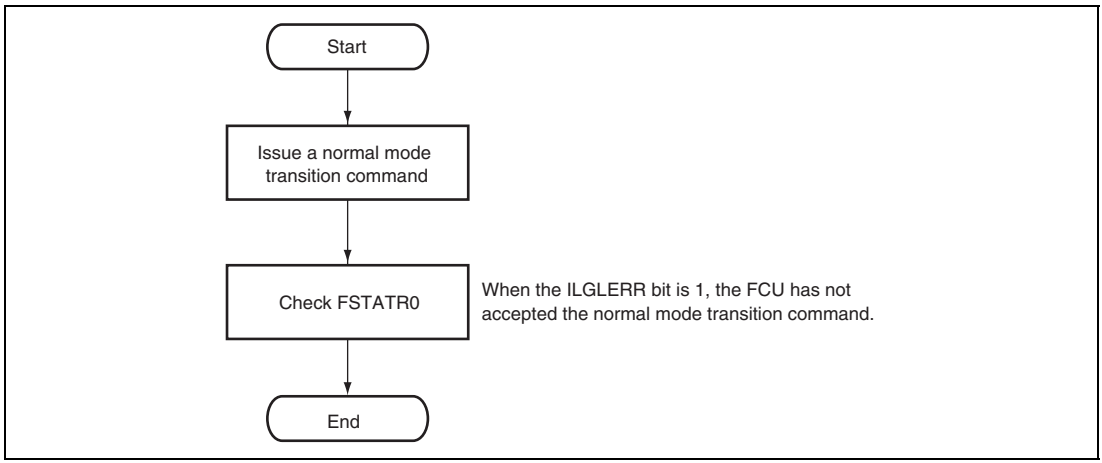


Figure 20.18 Procedure to Use Flash Memory P/E Normal Mode Transition Command

(5) Using Flash Clock Notification Command

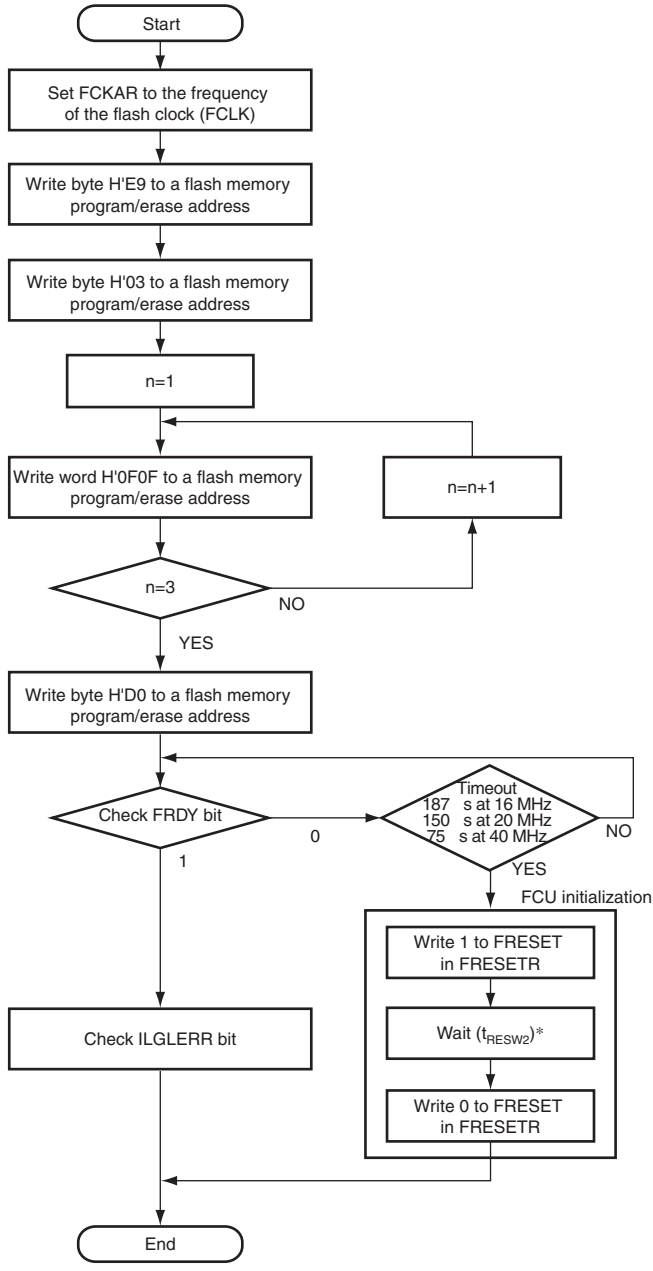
Before data is written to or erased from flash memory, the current flash clock should be specified.

First, set FCKAR to the frequency of the flash clock used. The range of settable frequencies is from 8 MHz to 40 MHz. The frequency should not be set outside this range.

Next, issue the flash clock notification command. Write byte H'E9 in the first cycle of the flash clock notification command and byte H'03 in the second cycle to the flash memory program/erase address. Access the peripheral bus in words in the third to fifth cycles of the command. Here, the start address must be a 4-byte boundary address. After writing word H'0F0F to the flash program/erase address three times, write byte H'D0 to the flash memory program/erase address in the sixth cycle; the FCU then starts the processing of peripheral clock frequency setting. Read the FRDY bit in FSTATR0 to confirm that setting is completed.

The addresses that can be specified in the first to sixth cycles are H'D00000 to H'D7FFFF when the FENTRY0 bit is set to 1. If the command is issued in incorrect combination of the address and

FENTRY0 bit settings, the FCU detects an error and enters command-locked state (see section 20.8.3, Error Protection).



Note: * t_{RESW2}: Reset pulse width during programming or erasure (see secion 25, Electrical Characteristics).

Figure 20.19 Procedure for Issuing Flash Clock Notification Command

(6) Programming

To program the flash memory, use the program command. Write byte H'E8 to a flash memory program/erase address in the first cycle of the program command and byte H'40 in the second cycle. Access the peripheral bus in words from the third to 66th cycles of the command. In the third cycle, write the programming data to the start address for flash memory programming. Here, the start address must be a 128-byte boundary address. After writing words to flash memory program/erase addresses 63 times, write byte H'D0 to a flash memory program/erase address in the 67th cycle; the FCU then starts flash memory programming. Read the FRDY bit in FSTATR0 to confirm that flash memory programming is completed.

The addresses that can be specified in the first to 67th cycles depend on the settings of the FENTRY0 bit in FENTRYR. When the FENTRY0 bit is set to 1, addresses from H'D00000 to H'D7FFFF can be specified. If a command is issued while wrong addresses are specified, the FCU detects an error and enters command-locked state (see section 20.8.3, Error Protection).

If the area accessed in the third to 66th cycles includes addresses that do not need to be programmed, write H'FFFF as the programming data for those addresses. To ignore the protection provided by the lock bit during programming, set the FPROTCN bit in FPROTR to 1 before starting programming.

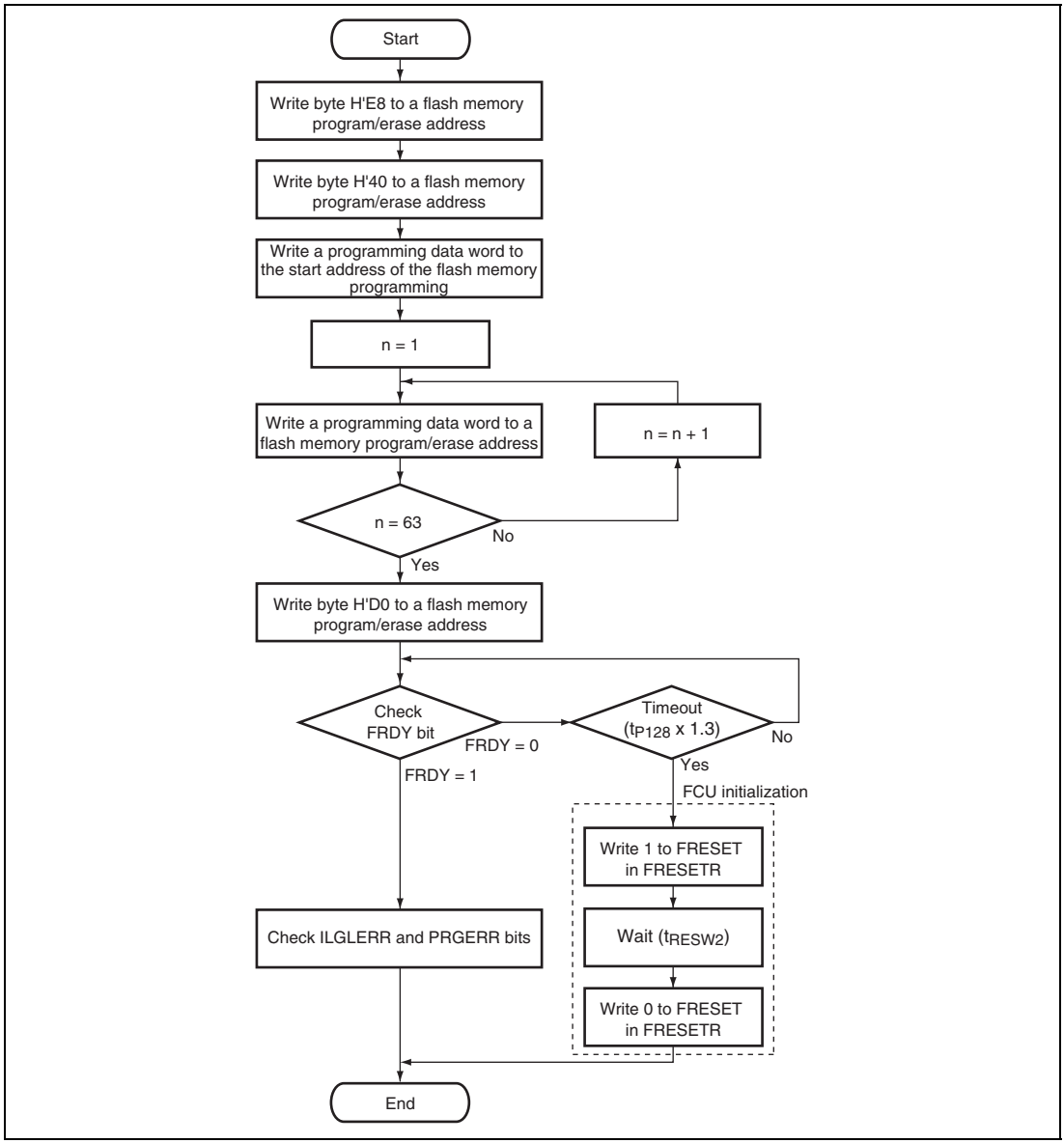


Figure 20.20 Procedure for Flash Memory Programming

(7) Erasure

To erase the flash memory, use the block erase command. Write byte H'20 to a flash memory program/erase address in the first cycle of the block erase command. Write byte H'D0 to an address in the target erasure block in the second cycle; the FCU then starts flash memory erasure. Read the FRDY bit in FSTATR0 to confirm that flash memory erasure is completed.

To ignore the protection provided by the lock bit during erasure, set the FPROTCN bit in FPROTR to 1 before starting erasure.

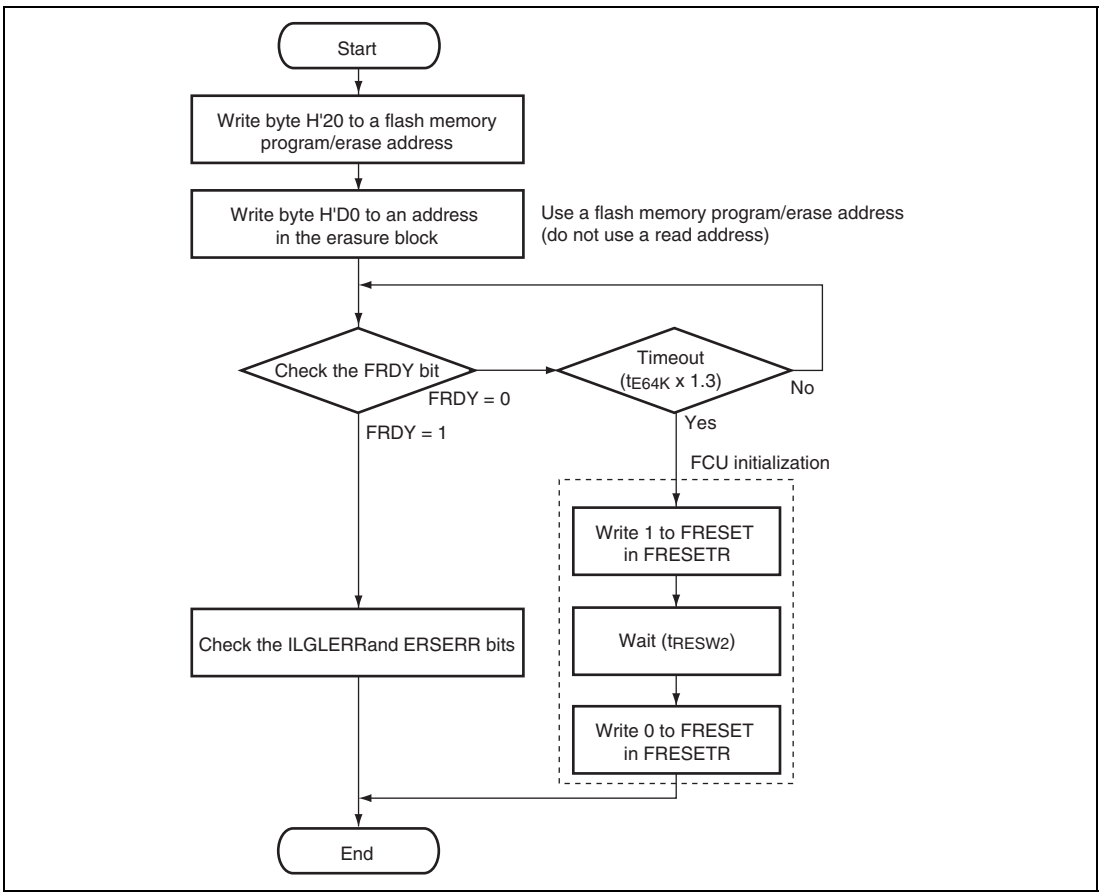


Figure 20.21 Procedure for Flash Memory Erasure

(8) Suspending Programming or Erasure

To suspend programming or erasure of the flash memory, use the P/E suspend command. Before issuing a P/E suspend command, check that the ILGLERR, PRGERR, and ERSERR bits in FSTATR0 and the FCUERR, FRDTCT, and FRCRCT bits in FSTATR1 are all 0; that is, to ensure that programming or erasure processing is being performed correctly. In addition, also check that the SUSRDY bit in FSTATR1 is 1 to make sure that the FCU can accept a suspend command. After issuing a P/E suspend command, read FSTATR0 and FSTATR1 to make sure that no error has occurred. If any error has occurred during programming/erasure, at least one of bits ILGLERR, PRGERR, ERSERR, FCUERR, FRDTCT, and FRCRCT is set to 1. In the case where programming/erasure is completed after making sure that SUSRDY bit is 1 and before the P/E suspend command is accepted, the ILGLERR bit is set to 1 because the P/E suspend command is detected as an illegal command. If the acceptance of the P/E suspend command and completion of programming/erasure coincide, no error results and the FCU does not enter the suspended state (that is, FRDY bit is 1 and ERSSPD and PRGSPD bits are both 0). When the P/E suspend command is accepted and the processing to suspend programming/erasure is completed normally, the FCU enters the suspended state with the FRDY bit set to 1 and either the ERSSPD or PRGSPD bit set to 1. After issuing the P/E suspend command, ensure that the FCU has entered the suspended state and then determine which operation to perform in the succeeding process. If a P/E resume command is issued in the succeeding process while the FCU has not entered suspended state, an illegal command error occurs and the FCU enters command-locked state (see section 20.8.3, Error Protection).

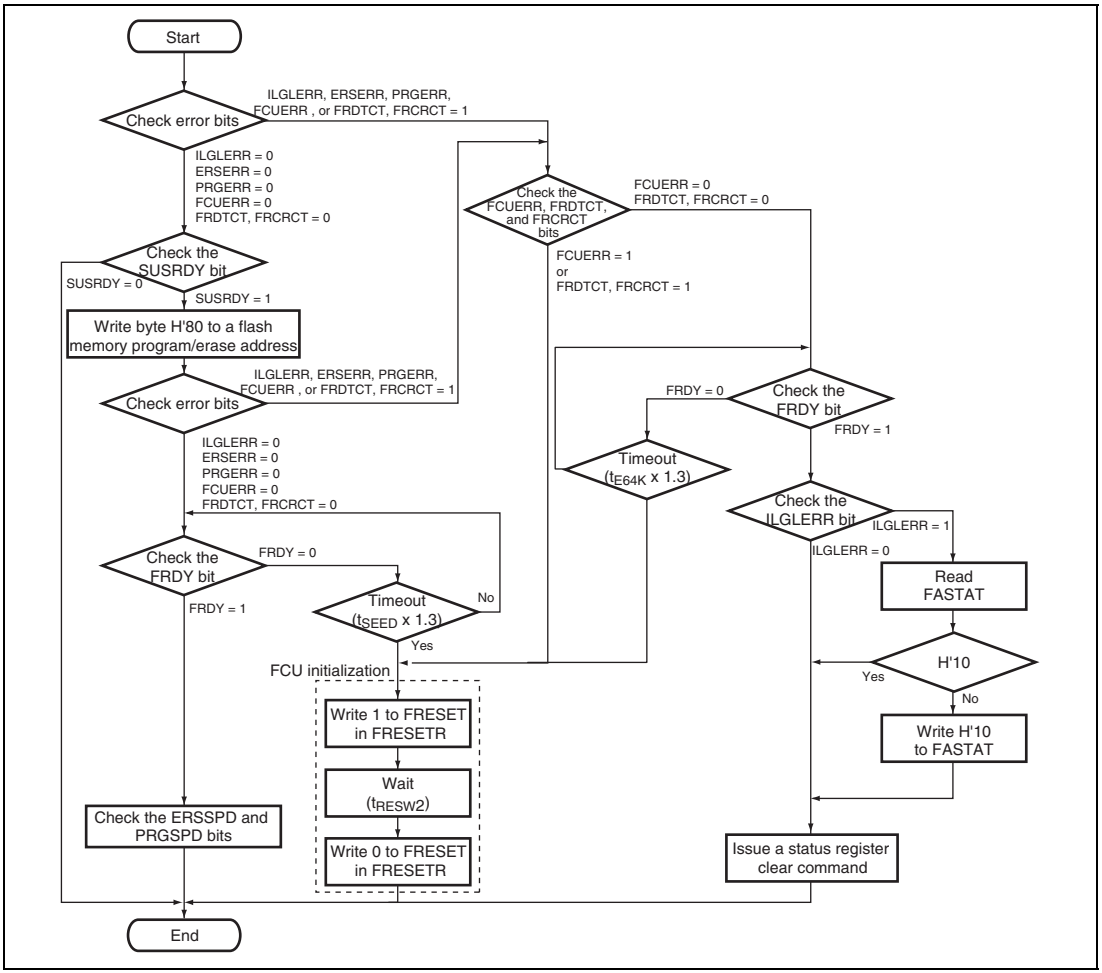


Figure 20.22 Procedure for Programming/Erasure Suspension

When the erasure-suspended state is entered, a block other than the target erasure block can be programmed. In both the programming-suspended and erasure-suspended states, the FCU can enter flash memory read mode by clearing FENTRYR.

For the operation when the FCU accepts a P/E suspend command, see section 20.6.4, Suspending Operation.

(9) Resuming Programming or Erasure

To resume programming or erasure that has been suspended, use the P/E resume command. If the FENTRYR setting has been modified during suspension, issue a P/E resume command only after resetting FENTRYR to the previous value that was held before the P/E suspension command was issued.

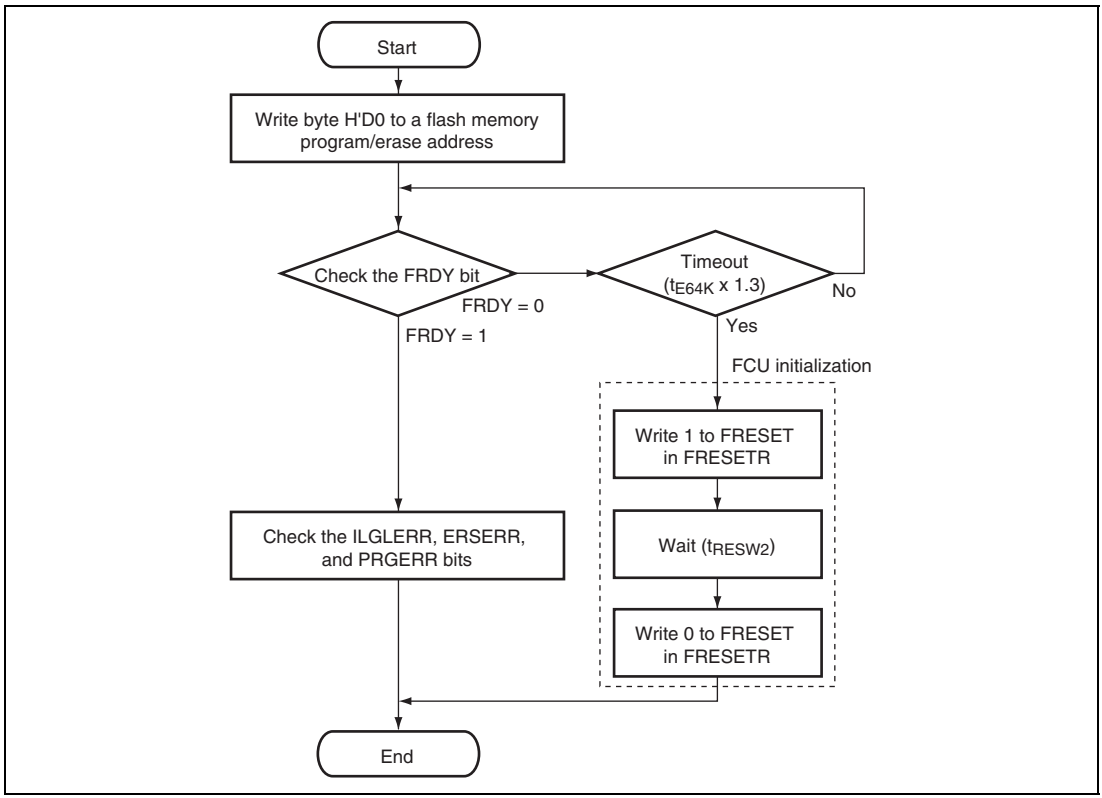


Figure 20.23 Procedure for Resuming Programming or Erasure

(10) Clearing Status Register 0 (FSTATR0)

To clear the ILGLERR, PRGERR, and ERSERR bits in FSTATR0, use the status register clear command. When any one of the ILGLERR, PRGER, and ERSERR bits is 1, the FCU is in command-locked state, in which the FCU only accepts the status register clear command and does not accept other commands. When the ILGLERR bit is 1, check also the value of the ROMAE, EEPAE, EEPRPE, and EEPWPE bits in FASTAT. If a status register clear command is issued without clearing these bits, the ILGLERR bit is not cleared.

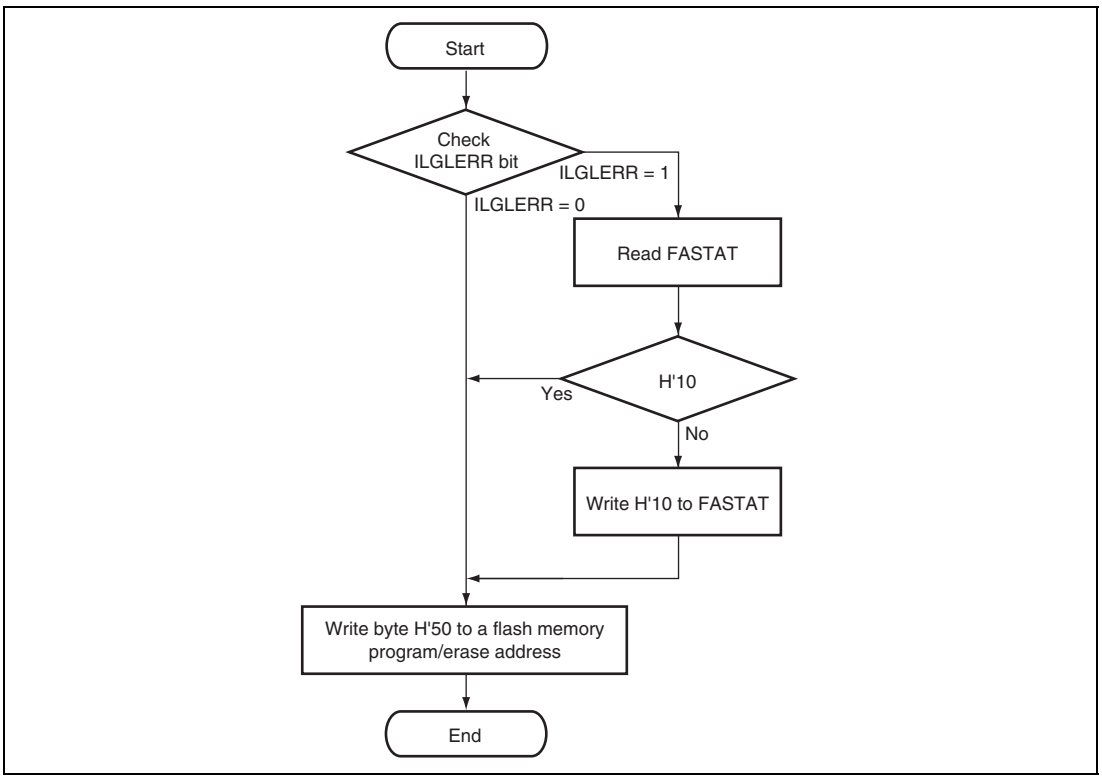


Figure 20.24 Procedure for Clearing Status Register 0

(11) Checking Status Register 0 (FSTATR0)

The FSTATR0 value can be checked in two ways: one is to directly read FSTATR0 and the other is to read a flash memory program/erase address in flash memory status read mode. After an FCU command is issued that is neither a normal mode transition command nor a lock bit read mode transition command, the FCU enters flash memory status read mode. In the example shown in figure 20.25, a status read mode transition command is issued to enter flash memory status read mode, and then a flash memory program/erase address is read to check the FSTATR0 value.

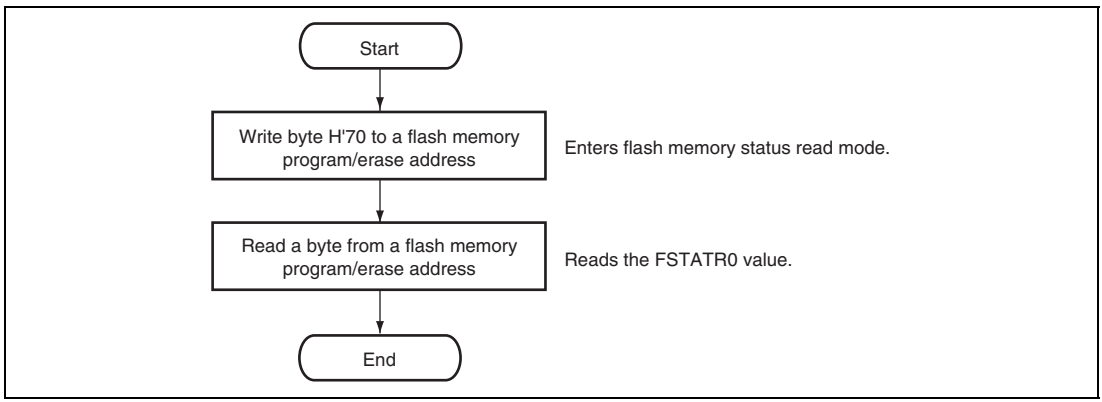


Figure 20.25 Procedure for Checking Status Register 0

(12) Reading Lock Bit

Each erasure block in the user MAT has a lock bit. While the FPROTCN bit in FPROTR is 0, the erasure block whose lock bit is set to 0 cannot be programmed or erased.

The lock bit status can be checked in either memory area read mode or register read mode. In memory area read mode (the FRDMD bit in FMODR is 0), read a flash memory program/erase address in flash memory lock bit read mode, and the lock bit value in the specified erasure block is copied to all bits in the data read through the peripheral bus. In register read mode (the FRDMD bit in FMODR is 1), issue a lock bit read 2 command, and the lock bit value in the specified erasure block is copied to the FLOCKST bit in FSTATR1.

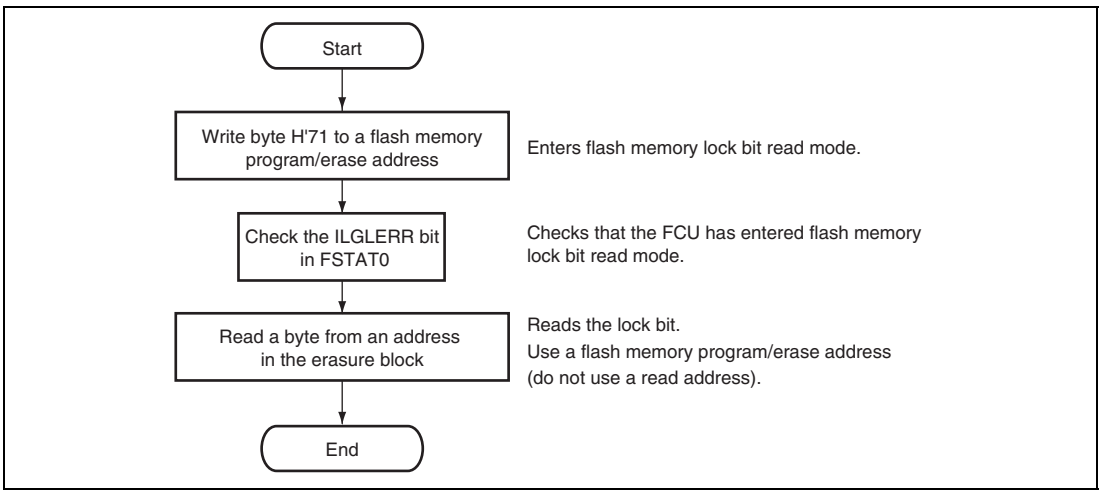


Figure 20.26 Procedure for Reading Lock Bit in Memory Area Read Mode

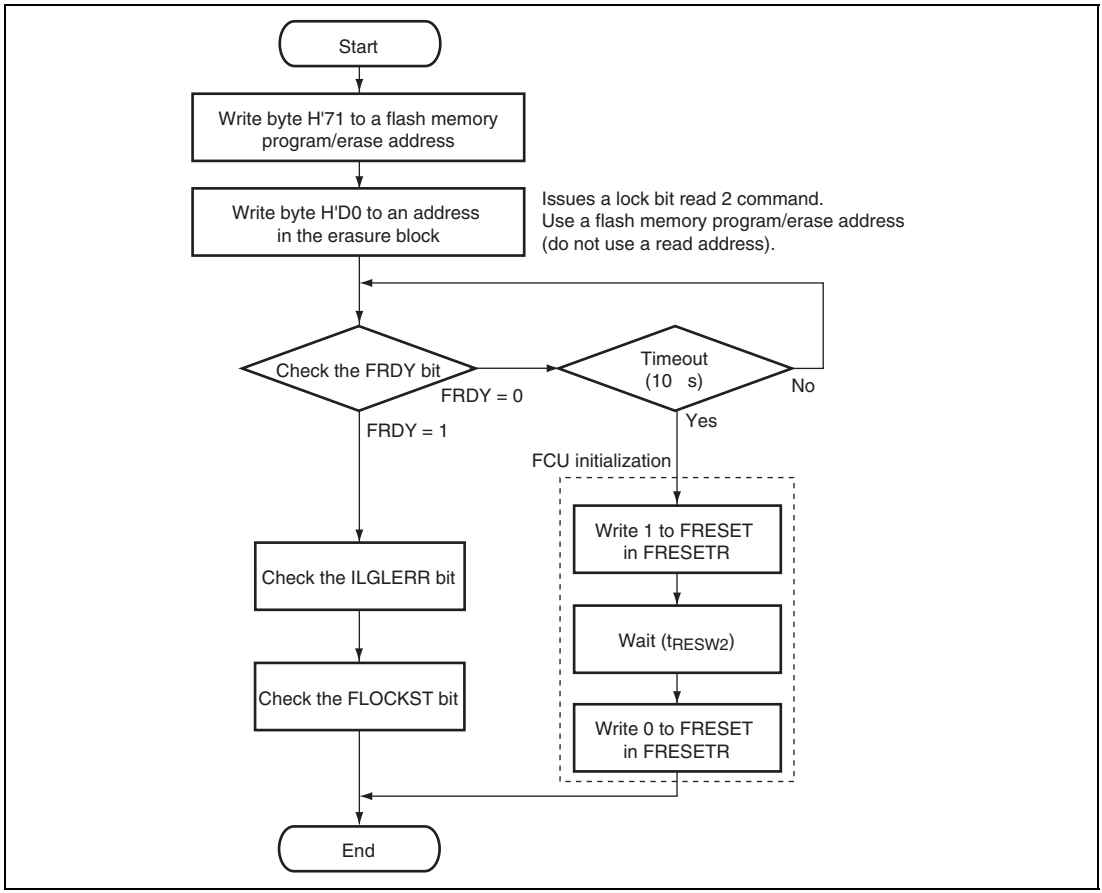


Figure 20.27 Procedure for Reading Lock Bit in Register Read Mode

(13) Writing to Lock Bit

Each erasure block in the user MAT has a lock bit. To write to a lock bit, use the lock bit program command. Write byte H'77 to a flash memory program/erase address in the first cycle of the lock bit program command. Write byte H'D0 to an address in the target erasure block whose lock bit is to be written to in the second cycle; the FCU then starts writing to the lock bit. Read the FRDY bit in FSTATR0 to confirm that writing is completed.

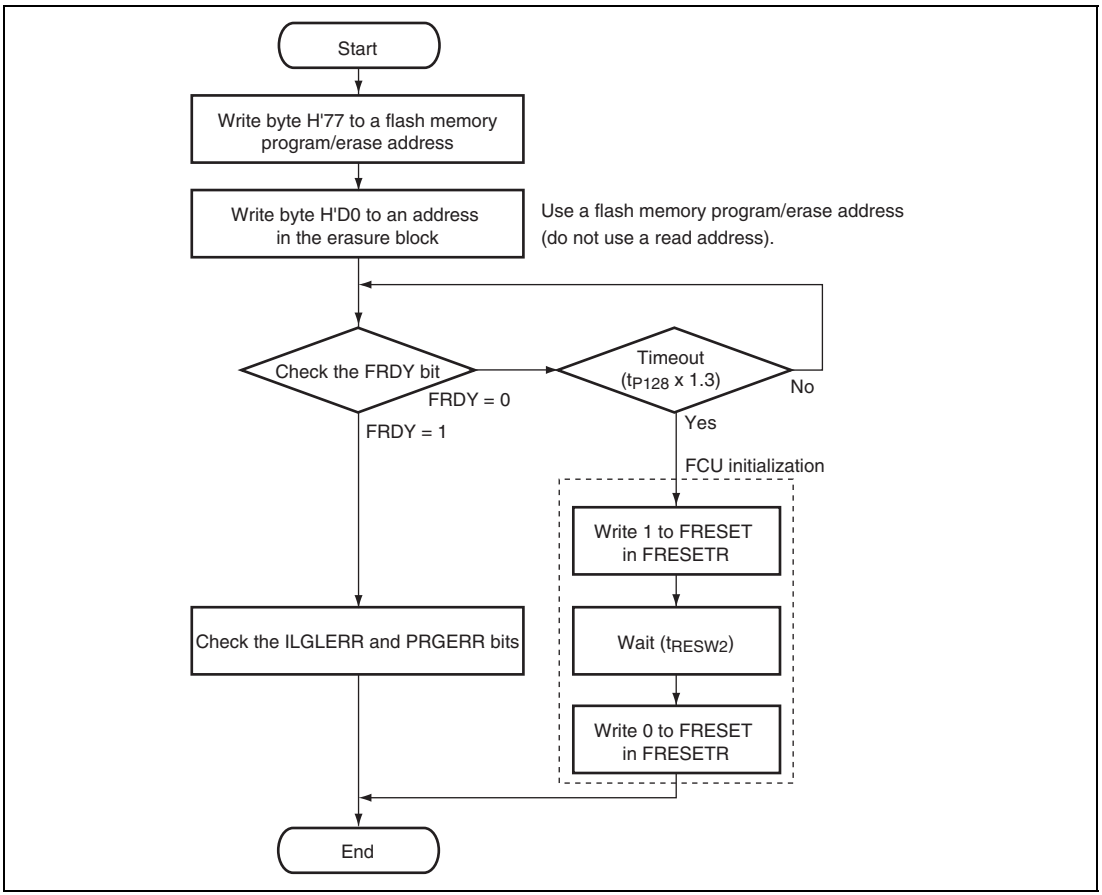


Figure 20.28 Procedure for Writing to the Lock Bit

To erase a lock bit, use the block erase command. While the FPROTCN bit in FPROTR is 0, the erasure block whose lock bit is set to 0 cannot be erased. Set the FPROTCN bit to 1, and then issue a block erase command to erase a lock bit. The block erase command erases all data in the specified erasure block; it is not possible to erase only the lock bit.

20.6.4 Suspending Operation

When a P/E suspend command is issued while flash memory is being programmed or erased, the FCU suspends the programming or erasure processing. Figure 20.29 gives an overview of operation for suspending programming. Upon acceptance of a programming command, the FCU clears the FRDY bit in FSTATR0 to 0 and starts programming processing.

If the FCU enters a state in which it can accept a P/E suspend command after programming processing is started, the SUSRDY bit is set to 1. When a P/E suspend command is issued, the FCU accepts it and clears the SUSRDY bit.

If the FCU accepts a P/E suspend command while applying a programming pulse to the flash memory, the FCU continues to apply the pulse. After the specified pulse application period has passed, the FCU stops applying the pulse, starts the processing for suspending programming, and sets the PRGSPD bit to 1. On completion of the suspending processing, the FCU sets the FRDY bit to 1 and enters the programming-suspended state. When the FCU accepts a P/E resume command in the programming-suspended state, the FCU clears the FRDY and PRGSPD bits to 0 and resumes programming processing.

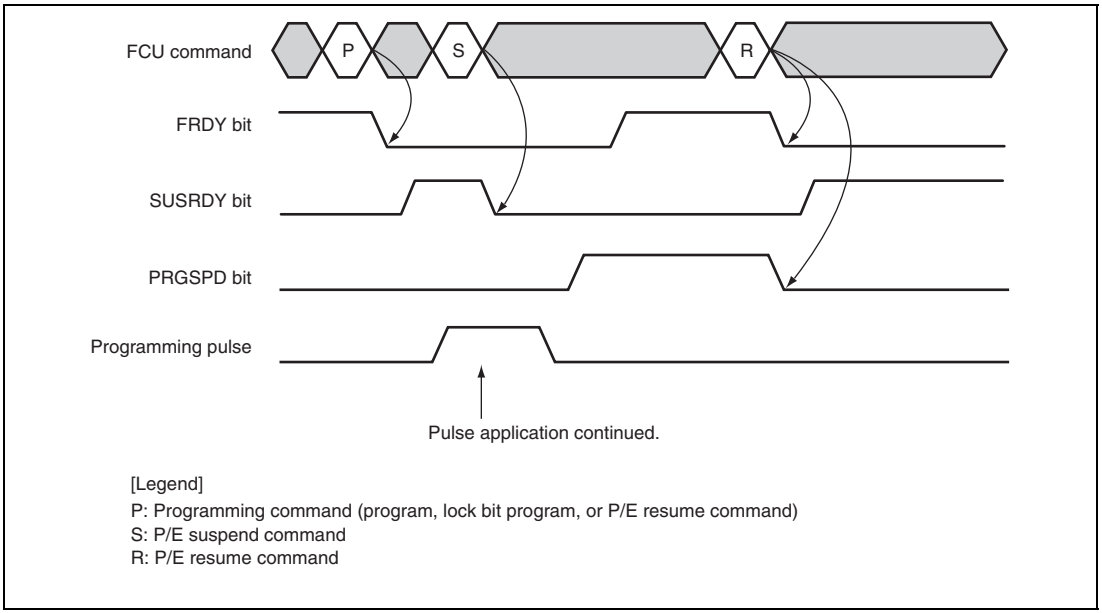


Figure 20.29 Suspending Programming Processing

Figure 20.30 shows the operation for suspending erasure processing in suspension-priority mode (the ESUSPMD bit in FCPSR is 0). Upon accepting an erasing command, the FCU clears the FRDY bit to 0 and starts erasure processing. If the FCU enters a state in which acceptance of a P/E suspend command is possible after erasure processing is started, the SUSRDY bit is set to 1. When a P/E suspend command is issued, the FCU accepts it and clears the SUSRDY bit. If the FCU accepts a P/E suspend command during erasing processing, the FCU starts the operation for suspending erasure, even if it is applying an erasing pulse, and sets the ERSSPD bit to 1. Upon completion of the suspending processing, the FCU sets the FRDY bit to 1 and enters the erasure-suspended state. When the FCU accepts a P/E resume command while it is in the erasure-suspended state, the FCU clears the FRDY and ERSPD bits to 0 and resumes erasure processing. The operations of the FRDY, SUSRDY, and ERSSPD bits are the same for all erasure-suspended modes.

The setting of erasure-suspended mode affects the control of erasing pulses. In suspension-priority mode, If the FCU accepts a P/E suspend command while applying an erasing pulse A that has not yet been suspended, the FCU suspends the pulse A application and enters the erasure-suspended state. If the FCU accepts a P/E suspend command while reapplying pulse A that was suspended and then resumed by a P/E resume command, the FCU continues applying pulse A. After the specified pulse application period has passed, the FCU stops applying the pulse and enters the erasure-suspended state. Subsequently, when the FCU accepts a P/E resume command and starts application of a new erasing pulse B, and then the FCU accepts a P/E suspend command again, the

application of erasing pulse B is suspended. In suspension-priority mode, priority is given to suspension processing over erasure processing by suspending application of the erasing pulse once per pulse, which may reduce delay in suspension processing in some cases.

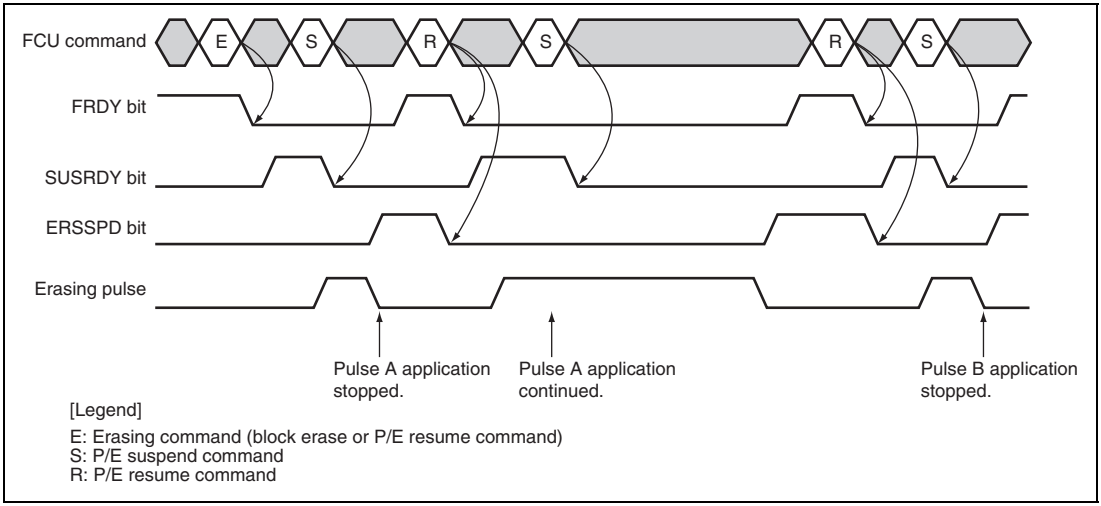


Figure 20.30 Suspending Erasure Processing (Suspension-Priority Mode)

Figure 20.31 shows the operation for suspending erasure processing in erasure-priority mode (the ESUSPMD bit in FCPSR is 1). In this mode, the erasing pulse is controlled in the same way as the programming pulse is controlled in programming suspending processing.

If the FCU accepts a P/E suspend command while applying an erasing pulse, the FCU always continues applying the pulse. As processing to reapply an erasing pulse never takes place in this mode, the total time required for erasure processing is shorter than in suspension-priority mode.

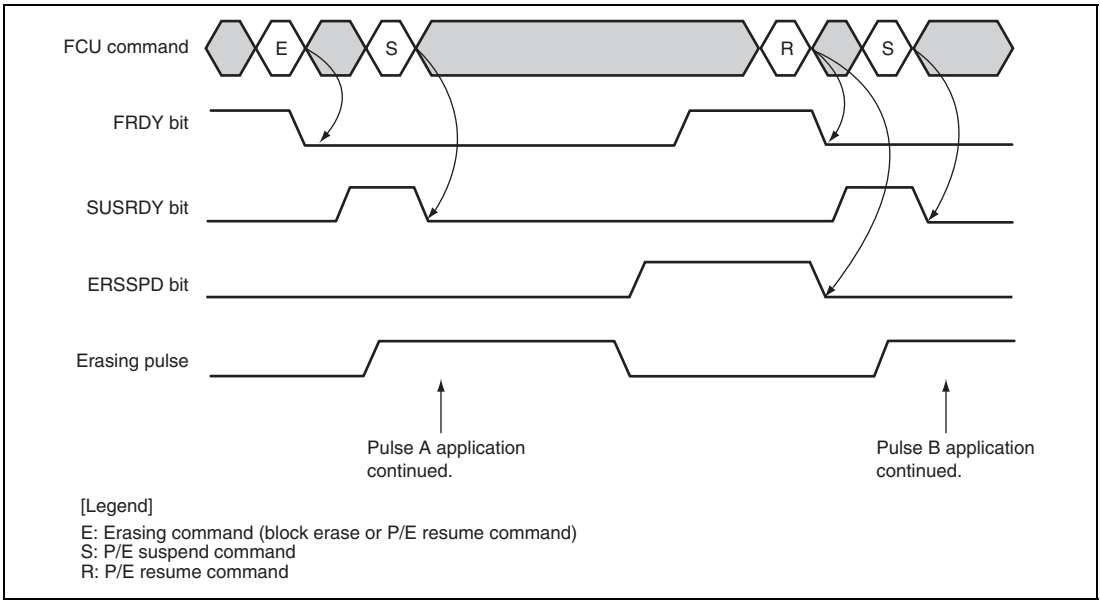


Figure 20.31 Suspending Erasure Processing (Erasure-Priority Mode)

20.7 Programmer Mode

In programmer mode, the on-chip ROM can be programmed with a general-purpose PROM programmer that supports the devices in table 20.13.

Table 20.13 Device Type Supported in Programmer Mode

Target MAT	Capacity	Device Type
User MAT	512 Kbytes	FZTAT512V5A
User boot MAT*	16 Kbytes	FZTAT USBT16V5A

Note: * This LSI does not have user boot MAT.

20.8 Protection

There are three types of flash memory programming/erasure protection: hardware, software, and error protection.

20.8.1 Hardware Protection

The hardware protection function disables flash memory programming and erasure according to the LSI pin settings.

(1) Protection through Mode Pins

While the on-chip ROM is disabled, flash memory programming, erasing, and reading are disabled (this LSI does not support on-chip ROM disabled mode). For the operating modes set through the mode pins of this LSI, refer to section 3, MCU Operating Modes.

20.8.2 Software Protection

The software protection function disables flash memory programming and erasure according to the control register settings or the lock bit settings in the user MAT. If an attempt is made to issue a programming or erasing command to the flash memory against software protection, the FCU detects an error and enters command-locked state.

(1) Protection through FENTRYR

When the FENTRY0 bit in FENTRYR is 0, the flash memory (read addresses: H'000000 to H'07FFFF; program/erase addresses: H'D00000 to H'D7FFFF) is placed in flash memory read mode. In flash memory read mode, the FCU does not accept commands, so flash memory programming and erasure are disabled. If an attempt is made to issue an FCU command in flash memory read mode, the FCU detects an illegal command error and enters command-locked state (see section 20.8.3, Error Protection).

(2) Protection through Lock Bits

Each erasure block in the user MAT has a lock bit. When the FPROTCN bit in FPROTR is 0, the erasure block whose lock bit is set to 0 cannot be programmed or erased. To program or erase the erasure block whose lock bit is 0, set the FPROTCN bit to 1. If an attempt is made to issue a command that programs or erases flash memory in violation of protection by lock bits, the FCU detects a programming/erasure error and enters command-locked state (see section 20.8.3, Error Protection).

20.8.3 Error Protection

The error protection function detects an illegal FCU command issued, an illegal access, or an FCU malfunction, and disables FCU command acceptance (command-locked state). While the FCU is in command-locked state, the flash memory cannot be programmed or erased. To cancel command-locked state, issue a status register clear command while FASTAT is H'10.

While the CMDLKIE bit in FAEINT is 1, a flash interface error (FIFE) interrupt is generated if the FCU enters command-locked state (the CMDLK bit in FASTAT becomes 1). While the ROMAEIE bit in FAEINT is 1, an FIFE interrupt is generated if the ROMAE bit in FASTAT becomes 1.

Table 20.14 shows the error protection types dedicated for the flash memory, those used in common by the flash memory and the EEPROM, and the status bit values (the ILGLERR, ERSERR, and PRGERR bits in FSTATR0, the FCUERR, FRDTCT, and FRCRCT bits in FSTATR1, and the ROMAE bit in FASTST) after each error detection. If the FCU enters command-locked state due to a command other than a suspend command issued during programming or erasure processing, the FCU continues programming or erasing the flash memory. In this state, the P/E suspend command cannot suspend programming or erasure. If a command is issued in command-locked state, the ILGLERR bit becomes 1 and the other bits retain the values set due to the previous error detection.

Table 20.14 Error Protection Types

Error	Description	ILGLERR	ERSERR	PRGERR	FCUERR	FRDTCT	FRCRCT	ROMAE
FENTRYR setting error	The value set in FENTRYR is not H'0001 or H'0080.	1	0	0	0	0	0	0
	The FENTRYR setting for resuming operation does not match that for suspending operation.	1	0	0	0	0	0	0
Illegal command error	An undefined code has been specified in the first cycle of an FCU command.	1	0	0	0	0	0	0
	The value specified in the last of the multiple cycles of an FCU command is not H'D0.	1	0	0	0	0	0	0
	The command issued during programming or erasure is not a suspend command.	1	0	0	0	0	0	0
	A suspend command has been issued during operation that is neither programming nor erasure.	1	0	0	0	0	0	0
	A suspend command has been issued in suspended state.	1	0	0	0	0	0	0
	A resume command has been issued in a state that is not a suspended state.	1	0	0	0	0	0	0
	A programming or erasing command (program, lock bit program, or block erase command (see section 20.6, User Program Mode)) has been issued in programming-suspended state.	1	0	0	0	0	0	0
	A block erase command has been issued in erasure-suspended state.	1	0	0	0	0	0	0
	A program or lock bit program command has been issued for an erasure-suspended area in erasure-suspended state.	1	0	0	0	0	0	0
	The value specified in the second cycle of a program command is not H'40.	1	0	0	0	0	0	0
	A command has been issued in command-locked state.	1	0/1	0/1	0/1	0/1	0/1	0/1

Error	Description	ILGLERR	ERSERR	PRGERR	FCUERR	FRDTCT	ECCERR	ROMAE
Erasure error	An error has occurred during erasure processing.	0	1	0	0	0	0	0
	A block erase command has been issued for the erasure block whose lock bit is set to 0 while the FPROTCN bit in FPROTR is 0.	0	1	0	0	0	0	0
Programming error	An error has occurred during programming processing.	0	0	1	0	0	0	0
	A program or lock bit program command has been issued for the erasure block whose lock bit is set to 0 while the FPROTCN bit in FPROTR is 0.	0	0	1	0	0	0	0
FCU error	An error has occurred during CPU processing in the FCU.	0	0	0	1	0	0	0
FCU RAM ECC error	A 1-bit error has been corrected during FCU RAM reading.	0	0	0	0	0	1	0
	A 2-bit error has been detected during FCU RAM reading.	0	0	0	0	1	0	0
Flash memory access error	A read access command has been issued to addresses H'D00000 to H'D7FFFF while FENTRY0 = 1 in flash memory P/E normal mode.	1	0	0	0	0	0	1
	An access command has been issued to addresses H'D00000 to H'D7FFFF while FENTRY0 = 0.	1	0	0	0	0	0	1
	A read access command has been issued to addresses H'000000 to H'07FFFF while FENTRY0 = 1.	1	0	0	0	0	0	1

20.9 Usage Notes

20.9.1 Other Notes

(1) State in which Interrupts are Ignored

In the following states, the NMI or maskable interrupt requests are ignored.

- Boot mode
- Programmer mode

(2) Programming-/Erasure-Suspended Area

The data stored in the programming-suspended or erasure-suspended area is undefined. To avoid malfunction due to undefined read data, ensure that no instruction is executed or no data is read from the programming-suspended or erasure-suspended area.

(3) Compatibility with Programming/Erasing Program of Conventional F-ZTAT H8SX Microcomputers

The flash memory programming/erasing program used for conventional F-ZTAT H8SX microcomputers does not work with this LSI.

(4) Reset during Programming or Erasure

When resetting the FCU by setting the FRESET bit in FRESETR during programming or erasure, hold the reset state for a period of t_{RESW2}. As a high voltage is applied to the flash memory during programming or erasure, the FCU must be kept in the reset state over a long enough time to secure falling of the applied voltage. Do not attempt to read from flash memory while the FCU is being reset.

When this LSI is reset by asserting the $\overline{\text{RES}}$ pin during programming or erasure, hold the reset state for a period of t_{RESW2}. In this reset, not only the time required for the voltage applied to flash memory to drop but also the time required for initialization of flash memory power supply and its internal circuitry must be secured.

Ensure that an internal reset is not caused by overflow of the WDT counter during programming or erasure. This is because, in the reset caused by the WDT, the time required for the voltage on memory to fall and for initialization of the flash memory power supply and within the flash memory cannot be secured.

When a reset is initiated by assertion of the signal on the $\overline{\text{RES}}$ pin or the FCU is reset by setting of the FRESET bit in FRESETR during processing for programming or erasure, data in the target region for programming or erasure become undefined.

(5) Prohibition of Additional Programming

One area cannot be programmed twice in succession. To program an area that has already been programmed, be sure to erase the area before reprogramming.

(6) Suspending Programming or Erasure

If processing for programming or erasure is suspended by issuing the programming/erasure suspend command, be sure to issue the resume command so that the operation is completed.

Section 21 Data Flash (EEPROM)

The H8SX/1727S and H8SX/1725S incorporate 32 Kbytes and 16 Kbytes of flash memory (EEPROM), respectively, for storing data. This data flash (hereafter, referred to as EEPROM) has the following features.

21.1 Features

- Flash-memory MATs

  The EEPROM has two types of memory areas (hereafter referred to as memory MATs) in the same address space. These two MATs can be switched by bank switching through the control register. For addresses H'E00080 to H'E07FFF, the data MAT contents will always be read even when the product information MAT is selected. The product information MAT cannot be programmed or erased.

  Data MAT: 32 K bytes (H8SX/1727S), 16 Kbytes (H8SX/1725S)

  Product information MAT: 128 bytes

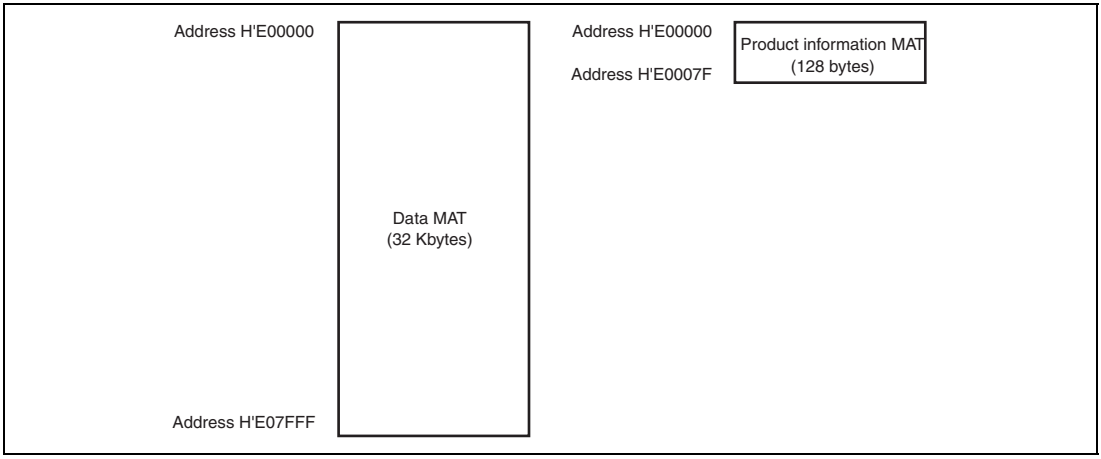


Figure 21.1 Memory MAT Configuration in EEPROM

Notes: 1. Lock bits cannot be used in this LSI.
2. This section describes only the 32-Kbyte EEPROM.
3. For the 16-Kbyte memory MAT configuration, refer to section 3.3.4, Address Map.

- Reading through the peripheral bus

  Both the data MAT and product information MAT can be read through the peripheral bus in three peripheral clock (Fϕ) cycles in byte or word units or in six Fϕ cycles in longword units.

  When the Fϕ clock is 20 MHz or slower, however, reading from the MATs through the peripheral bus can be done in two Fϕ cycles in byte or word units and in four Fϕ cycles in longword units by clearing the EEPWT bit (initial value is 1) in SYSCR1 to 0.

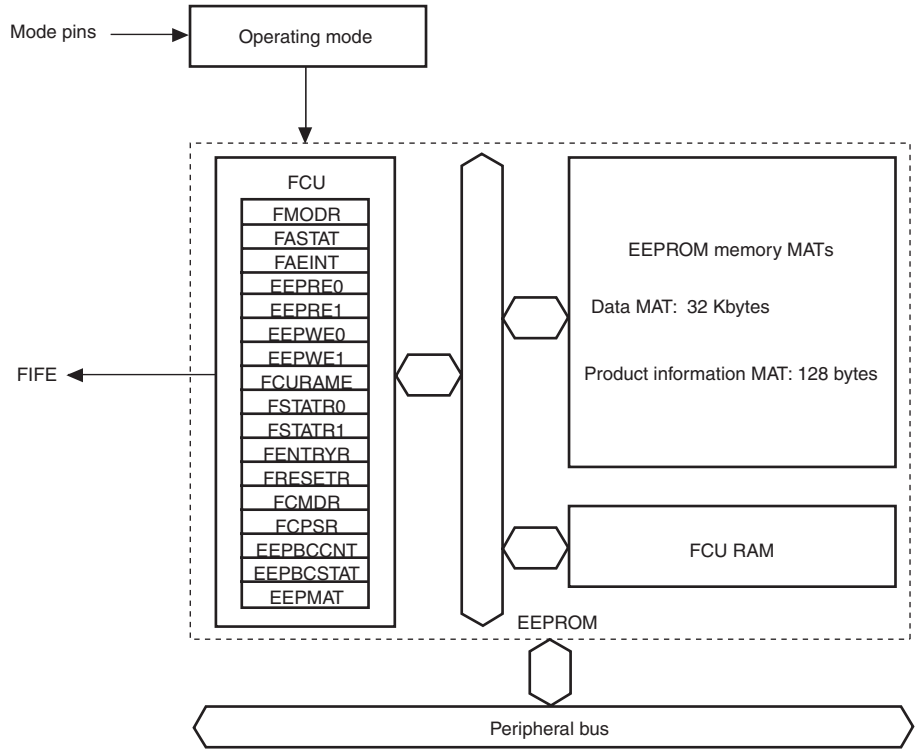
Table 21.1 Read Cycle Settings for Reading through the Peripheral Bus

XTAL	Iϕ Multiplication Factor	Iϕ	Fϕ Multiplication Factor ×2	Fϕ Multiplication Factor ×4
10 MHz	×4	40 MHz	Fϕ = 20 MHz Also available with EEPWT = 0	Fϕ = 40 MHz EEPWT = 1
	×8	80 MHz	Fϕ = 20 MHz Also available with EEPWT = 0	Fϕ = 40 MHz EEPWT = 1
8 MHz	×4	32 MHz	Fϕ = 16 MHz Also available with EEPWT = 0	Fϕ = 32 MHz EEPWT = 1
	×8	64 MHz	Fϕ = 16 MHz Also available with EEPWT = 0	Fϕ = 32 MHz EEPWT = 1

- Programming and erasing methods

  The data MAT can be programmed and erased by commands issued through the peripheral bus to the flash memory/EEPROM-dedicated sequencer (FCU).

  While the FCU is programming or erasing the data MAT, the CPU can execute a program located in the flash memory, RAM, or external address space. While the FCU is programming or erasing the flash memory or data MAT, data cannot be read form the data MAT. When the FCU suspends programming or erasure of the data MAT, the CPU can read data from the data MAT, and then the FCU can resume programming or erasure of the data MAT. While the FCU suspends erasure, areas other than the erasure-suspended area can be programmed.



[Legend]

FMODR:	Flash mode register
FASTAT:	Flash access status register
FAEINT:	Flash access error interrupt enable register
EEPRE0, EEPRE1:	EEPROM read enable registers 0 and 1
EEPWE0, EEPWE1:	EEPROM program/erase enable registers 0 and 1
FCURAME:	FCU RAM enable register
FSTATR0, FSTATR1:	Flash status registers 0 and 1
FENTRYR:	Flash P/E mode entry register
FRESETR:	Flash reset register
FCMDR:	FCU command register
FCPSR:	FCU processing switch register
EEPBCCNT:	EEPROM blank check control register
EEPBCSTAT:	EEPROM blank check status register
EEPMAT:	EEPROM MAT select register
FIFE:	Flash interface error interrupt

Figure 21.2 Block Diagram of EEPROM

- Programming/erasing unit

  The data MAT is programmed in 8-byte or 128-byte units and erased in block units (2 Kbytes) in user mode and user program mode. In boot mode, the data MAT is programmed in 128-byte units and erased in block units (2 Kbytes). The product information MAT is read-only memory and cannot be programmed or erased.

  Figure 21.3 shows the block configuration of the data MAT of this LSI. The data MAT is divided into sixteen 2-Kbyte blocks (DB00 to DB15) in the H8SX/1727S or eight 2-Kbyte blocks (DB00 to DB07) in the H8SX/1725S.

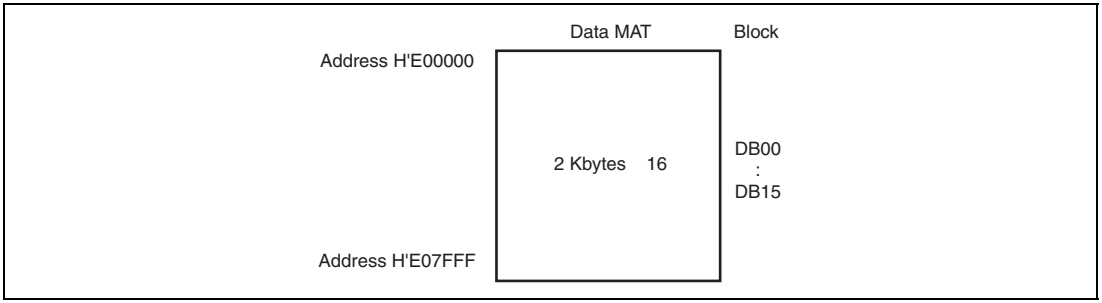


Figure 21.3 Block Configuration of Data MAT

- Blank check function

  If the CPU reads data from erased EEPROM, undefined values are read. Whether EEPROM is erased (blank) or not can be found out by using the blank check command of the FCU. The size of the area that a single blank check command can cover is 2 Kbytes (one erasure block) or eight bytes.

- Two types of on-board programming modes
  - — Boot mode

    The data MAT can be programmed using the SCI. The bit rate for SCI communication between the host and this LSI can be automatically adjusted.
  - — User mode/user program mode

    The data MAT can be programmed with a desired interface.

- Protection modes

  This LSI supports two modes to protect memory against programming or erasure: hardware protection by the levels on the mode pins and software protection by the setting of the FENTRYD bit, EEPRE0/ EEPRE1 registers, or EEPWE0/EEPWE1 registers. The FENTRYD bit enables or disables data MAT programming or erasure by the FCU. EEPRE0/EEPRE1 registers control protection of each data MAT block against reading, and EEPWE0/EEPWE1 registers control protection against programming and erasure. When the EEPWE0/EEPWE1 setting is modified, the data MAT contents are retained.

  The LSI also provides a function to suspend programming or erasure when abnormal operation is detected during programming or erasure. If the CPU attempts to fetch an instruction from EEPROM, address error exception handling will result. For details, see section 4, Exception Handling.

- Programming and erasing time and count

  Refer to section 25, Electrical Characteristics.

21.2 Input/Output Pins

Table 21.2 shows the input/output pins used for the EEPROM. The combination of MD1 and MD0 pin levels determines the EEPROM programming mode (see section 21.4, Overview of EEPROM-Related Modes).

Table 21.2 Pin Configuration

Pin Name	Abbreviation	I/O	Function
Reset	$\overline{\text{RES}}$	Input	This LSI enters the reset state when this signal goes low.
Mode	MD1, MD0	Input	These pins specify the operating mode.
Receive data in SCI	RxD4	Input	Receives data through SCI (channel 4)
Transmit data in SCI	TxD4	Output	Transmits data through SCI (channel 4)

21.3 Register Descriptions

Table 21.3 shows the EEPROM-related registers. Some of these registers have flash memory-related bits, but this section only describes the EEPROM-related bits. For the registers consisting of bits used by the flash memory and EEPROM in common (FMODR, FCURAME, FSTATR0, FSTATR1, FRESETR, FCMDR, and FCPSR) and the flash memory-dedicated bits, refer to section 20.3, Register Descriptions.

Table 21.3 Register Configuration

Register Name	Symbol	R/W*1	Initial Value	Address	Access Size
Flash mode register	FMODR	R/W	H'00	H'FFE002	8
Flash access status register	FASTAT	R/(W)*2	H'00	H'FFE010	8
Flash access error interrupt enable register	FAEINT	R/W	H'9F	H'FFE011	8
EEPROM read enable register 0	EEPRE0	R/(W)*3	H'0000	H'FFE040	8, 16
EEPROM read enable register 1	EEPRE1	R/(W)*3	H'0000	H'FFE042	8, 16
EEPROM program/erase enable register 0	EEPWE0	R/(W)*3	H'0000	H'FFE050	8, 16
EEPROM program/erase enable register 1	EEPWE1	R/(W)*3	H'0000	H'FFE052	8, 16
FCU RAM enable register	FCURAME	R/(W)*3	H'0000	H'FFE054	8, 16
Flash status register 0	FSTATR0	R*5	H'80	H'FFE100	8, 16
Flash status register 1	FSTATR1	R*5	H'00	H'FFE101	8, 16
Flash P/E mode entry register	FENTRYR	R/(W)*4*5	H'0000	H'FFE102	8, 16
Flash reset register	FRESETR	R/(W)*3	H'0000	H'FFE106	8, 16
FCU command register	FCMDR	R*5	H'FFFF	H'FFE10A	8, 16
FCU processing switch register	FCPSR	R/W*5	H'0000	H'FFE118	8, 16
EEPROM blank check control register	EEPBCCNT	R/W*5	H'0000	H'FFE11A	8, 16
EEPROM blank check status register	EEPBCSTAT	R*5	H'0000	H'FFE11B	8, 16
Flash clock notification register	FCKAR	R/W*5	H'0000	H'FFE138	16
EEPROM MAT select register	EEPMAT	R/(W)*3	H'0000	H'FFE380	8, 16

Notes: 1. In on-chip ROM disabled mode, the flash memory-related registers are always read as 0 and writing to them is ignored. (This LSI does not support on-chip ROM disabled mode.)
2. This register consists of the bits where only 0 can be written to clear the flags and the read-only bits.
3. This register can be written to only when a specified value is written to the upper byte in word access. The data written to the upper byte is not stored in the register.
4. This register can be written to only when a specified value is written to the upper byte in word access; the register is initialized when a value not allowed for the register is written to the upper byte. The data written to the upper byte is not stored in the register.
5. This register can be initialized by setting the FRESET bit in FRESETR to 1 as well as applying a reset through the $\overline{\text{RES}}$ pin.

21.3.1 Flash Mode Register (FMODR)

FMODR specifies the operating mode of the FCU.

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	FRDMD	—	—	—	—
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R/W	R	R	R	R

Bit	Bit Name	Initial Value	R/W	Description
7 to 5	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
4	FRDMD	0	R/W	FCU Read Mode Select Selects a mode of reading from flash memory and EEPROM using the FCU. For EEPROM, this bit is used to select between the processing for entering EEPROM lock bit read mode and blank check processing. (See section 21.6.1, FCU Command List, and section 21.6.3, FCU Command Usage.) For flash memory, this bit is used to select the mode for reading the lock bit value (see section 20, Flash Memory). 0: Memory area read mode Select this mode to enter EEPROM lock bit mode. However, since there are no lock bits in EEPROM, if reading from an EEPROM area is performed after entering lock bit mode, undefined data will be read. 1: Register read mode Select this mode when using the blank check command.
3 to 0	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.

21.3.2 Flash Access Status Register (FASTAT)

FASTAT indicates the access error status for the flash memory and EEPROM. If any bit in FASTAT is set to 1, the FCU enters command-locked state (see section 21.7.3, Error Protection). To cancel command-locked state, set FASTAT to H'10, and then issue a status-clear command to the FCU.

Bit	7	6	5	4	3	2	1	0
Bit Name	ROMAE	—	—	CMDLK	EEPAE	—	EEPRPE	EEPWPE
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R	R	R	R/(W)*	R	R/(W)*	R/(W)*

Note: * Only 0 can be written to clear the flag after 1 is read.

Bit	Bit Name	Initial Value	R/W	Description
7	ROMAE	0	R/(W)*	Flash Memory Access Error Refer to section 20, Flash Memory.
6, 5	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
4	CMDLK	0	R	FCU Command Lock Indicates whether the FCU is in command-locked state (see section 21.7.3, Error Protection). 0: The FCU is not in command-locked state 1: The FCU is in command-locked state [Setting condition] • The FCU detects an error and enters command-locked state. [Clearing condition] • The FCU completes the status-clear command processing while the FASTAT register holds H'10.

Bit	Bit Name	Initial Value	R/W	Description
3	EEPAE	0	R/(W)*	EEPROM Access Error Indicates whether an access error has been generated for the EEPROM. If this bit becomes 1, the ILGLERR bit in FSTATR0 is set to 1 and the FCU enters command-locked state. 0: No EEPROM access error has occurred 1: An EEPROM access error has occurred [Setting conditions] • A read access command is issued to the EEPROM area while the FENTRYD bit in FENTRYR is 1 in EEPROM P/E normal mode. • A write access command is issued to the EEPROM area while the FENTRYD bit in FENTRYR is 0. • An access command is issued to the EEPROM area while one of the FENTRY0 bit in FENTRYR is 1. See section 20, Flash Memory. [Clearing condition] • 0 is written to this bit after reading EEPAE = 1.
2	—	0	R	Reserved This bit is always read as 0. The write value should always be 0.

Bit	Bit Name	Initial Value	R/W	Description
1	EEPRPE	0	R/(W)*	EEPROM Read Protect Error Indicates whether an error has been generated against the EEPROM read protection provided by the EEPRE0 and EEPRE1 settings. 0: The EEPROM has not been read against the EEPRE0 and EEPRE1 settings 1: An attempt has been made to read data from the EEPROM against the EEPRE0 and EEPRE1 settings [Setting condition] • An attempt is made to read data from the EEPROM area specified read-protected through the EEPRE0 and EEPRE1 settings. [Clearing condition] • 0 is written to this bit after reading EEPRPE = 1.
0	EEPWPE	0	R/(W)*	EEPROM Program/Erase Protect Error Indicates whether an error has been generated against the EEPROM program/erasure protection provided by the EEPWE0 and EEPWE1 settings. 0: No programming or erasing command has been issued to the EEPROM against the EEPWE0 and EEPWE1 settings 1: A programming or erasing command has been issued to the EEPROM against the EEPWE0 and EEPWE1 settings [Setting condition] • A programming or erasing command is issued to the EEPROM area that has been program/erase-protected through the EEPWE0 and EEPWE1 settings. [Clearing condition] • 0 is written to this bit after reading EEPWPE = 1.

Note: * Only 0 can be written to clear the flag after 1 is read.

21.3.3 Flash Access Error Interrupt Enable Register (FAEINT)

FAEINT enables or disables output of flash interface error (FIFE) interrupt requests.

Bit	7	6	5	4	3	2	1	0
Bit Name	ROMAEIE	—	—	CMDLKIE	EEPAEIE	—	EEPRPEIE	EEPWPEIE
Initial Value:	1	0	0	1	1	1	1	1
R/W:	R/W	R	R	R/W	R/W	R	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
7	ROMAEIE	1	R/W	Flash Memory Access Error Interrupt Enable Refer to section 20, Flash Memory.
6, 5	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
4	CMDLKIE	1	R/W	FCU Command Lock Interrupt Enable Enables or disables an FIFE interrupt request when FCU command-locked state is entered and the CMDLK bit in FASTAT becomes 1. 0: Does not generate an FIFE interrupt request when CMDLK = 1 1: Generates an FIFE interrupt request when CMDLK = 1
3	EEPAEIE	1	R/W	EEPROM Access Error Interrupt Enable Enables or disables an FIFE interrupt request when an EEPROM access error occurs and the EEPAE bit in FASTAT becomes 1. 0: Does not generate an FIFE interrupt request when EEPAE = 1 1: Generates an FIFE interrupt request when EEPAE = 1

Bit	Bit Name	Initial Value	R/W	Description
2	—	1	R	Reserved This bit is always read as 1. The write value should always be 1.
1	EEPRPEIE	1	R/W	EEPROM Read Protect Error Interrupt Enable Enables or disables an FIFE interrupt request when an EEPROM read protect error occurs and the EEPRPE bit in FASTAT becomes 1. 0: Does not generate an FIFE interrupt request when EEPRPE = 1 1: Generates an FIFE interrupt request when EEPRPE = 1
0	EEPWPEIE	1	R/W	EEPROM Program/Erase Protect Error Interrupt Enable Enables or disables an FIFE interrupt request when an EEPROM program/erase protect error occurs and the EEPWPE bit in FASTAT becomes 1. 0: Does not generate an FIFE interrupt request when EEPWPE = 1 1: Generates an FIFE interrupt request when EEPWPE = 1

21.3.4 EEPROM Read Enable Register 0 (EEPRE0)

EEPRE0 enables or disables read access to blocks DB00 to DB07 (see figure 21.3) in the data MAT.

Bit	15	14	13	12	11	10	9	8
Bit Name	REKEY07	REKEY06	REKEY05	REKEY04	REKEY03	REKEY02	REKEY01	REKEY00
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*

Bit	7	6	5	4	3	2	1	0
Bit Name	DBRE07	DBRE06	DBRE05	DBRE04	DBRE03	DBRE02	DBRE01	DBRE00
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: * Written data is not stored in these bits.

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	REKEY07 to REKEY00	All 0	R/(W)*	Key Code These bits enable or disable DBRE07 to DBRE00 bit modification. The data written to these bits are not stored.
7 to 0	DBRE07 to DBRE00	0	R/W	DB07 to DB00 Block Read Enable Enables or disables read access to blocks DB07 to DB00 in the data MAT. The DBREi bit (i = 07 to 00) controls read access to block DBi. Writing to these bits is enabled only when this register is accessed in word size and H'2D is written to the KEY bits. 0: Disables read access 1: Enables read access

Note: * Written data is not stored in these bits.

21.3.5 EEPROM Read Enable Register 1 (EEPRE1)

EEPRE1 enables or disables read access to blocks DB08 to DB15 (see figure 21.3) in the data MAT.

Bit	15	14	13	12	11	10	9	8
Bit Name	REKEY15	REKEY14	REKEY13	REKEY12	REKEY11	REKEY10	REKEY09	REKEY08
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*

Bit	7	6	5	4	3	2	1	0
Bit Name	DBRE15	DBRE14	DBRE13	DBRE12	DBRE11	DBRE10	DBRE09	DBRE08
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: * Written data is not stored in these bits.

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	REKEY15 to REKEY08	All 0	R/(W)*	Key Code These bits enable or disable DBRE15 to DBRE08 bit modification. The data written to these bits are not stored.
7 to 0	DBRE15 to DBRE08	0	R/W	DB15 to DB08 Block Read Enable Enables or disables read access to blocks DB15 to DB08 in the data MAT. The DBREi bit (i = 15 to 08) controls read access to block DBi. Writing to these bits is enabled only when this register is accessed in word size and H'D2 is written to the KEY bits. 0: Disables read access 1: Enables read access

Note: * Written data is not stored in these bits.

21.3.6 EEPROM Program/Erase Enable Register 0 (EEPWE0)

EEPWE0 enables or disables programming and erasure of blocks DB00 to DB07 (see figure 21.3) in the data MAT. In on-chip ROM disabled mode, EEPWE0 is read as H'0000 and writing to it is ignored.

Bit	15	14	13	12	11	10	9	8
Bit Name	WEKEY07	WEKEY06	WEKEY05	WEKEY04	WEKEY03	WEKEY02	WEKEY01	WEKEY00
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*

Bit	7	6	5	4	3	2	1	0
Bit Name	DBWE07	DBWE06	DBWE05	DBWE04	DBWE03	DBWE02	DBWE01	DBWE00
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: * Written data is not stored in these bits.

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	WEKEY07 to WEKEY00	All 0	R/(W)*	Key Code These bits enable or disable DBWE07 to DBWE00 bit modification. The data written to these bits are not stored.
7 to 0	DBWE07 to DBWE00	0	R/W	DB07 to DB00 Block Program/Erase Enable Enables or disables programming and erasure of blocks DB07 to DB00 in the data MAT. The DBWEi bit (i = 07 to 00) controls programming and erasure of block DBi. Writing to these bits is enabled only when this register is accessed in word size and H'1E is written to the KEY bits. 0: Disables programming and erasure 1: Enables programming and erasure

Note: * Written data is not stored in these bits.

21.3.7 EEPROM Program/Erase Enable Register 1 (EEPWE1)

EEPWE1 enables or disables programming and erasure of blocks DB08 to DB15 (see figure 21.3) in the data MAT.

Bit	15	14	13	12	11	10	9	8
Bit Name	WEKEY15	WEKEY14	WEKEY13	WEKEY12	WEKEY11	WEKEY10	WEKEY09	WEKEY08
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*

Bit	7	6	5	4	3	2	1	0
Bit Name	DBWE15	DBWE14	DBWE13	DBWE12	DBWE11	DBWE10	DBWE09	DBWE08
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: * Written data is not stored in these bits.

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	WEKEY15 to WEKEY08	All 0	R/(W)*	Key Code These bits enable or disable DBWE15 to DBWE08 bit modification. The data written to these bits are not stored.
7 to 0	DBWE15 to DBWE08	0	R/W	DB15 to DB08 Block Program/Erase Enable Enables or disables programming and erasure of blocks DB15 to DB08 in the data MAT. The DBWEi bit (i = 15 to 08) controls read/erase access to block DBi. Writing to these bits is enabled only when this register is accessed in word size and H'E1 is written to the KEY bits. 0: Disables programming and erasure 1: Enables programming and erasure

Note: * Written data is not stored in these bits.

21.3.8 Flash P/E Mode Entry Register (FENTRYR)

FENTRYR specifies the P/E mode for the flash memory or EEPROM. To specify the P/E mode for the flash memory or EEPROM so that the FCU can accept commands, set FENTRYD to 1.

Bit	15	14	13	12	11	10	9	8
Bit Name	FEKEY7	FEKEY6	FEKEY5	FEKEY4	FEKEY3	FEKEY2	FEKEY1	FEKEY0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*

Bit	7	6	5	4	3	2	1	0
Bit Name	FENTRYD	—	—	—	—	—	—	FENTRY0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R	R	R	R	R	R	R/W

Note: * Written data is not stored in these bits.

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	FEKEY7 to FEKEY0	All 0	R/(W)*	Key Code These bits enable or disable modification of the FENTRYD and FENTRY0 bits. The data written to these bits are not stored.

Bit	Bit Name	Initial Value	R/W	Description
7	FENTRYD	0	R/W	EEPROM P/E Mode Entry This bit specifies the P/E mode for the EEPROM. 0: The EEPROM is in read mode 1: The EEPROM is in P/E mode [Write enabling conditions] When the following conditions are all satisfied: • The LSI is in on-chip ROM enabled mode. • The FRDY bit in FSTATR0 is 1. • H'AA is written to FEKEY in word access. [Setting condition] • 1 is written to FENTRYD while the write enabling conditions are satisfied and FENTRYR is H'0000. [Clearing conditions] • This register is written to in byte access. • A value other than H'AA is written to FEKEY in word access. • 0 is written to FENTRYD while the write enabling conditions are satisfied. • FENTRYR register is written while the write enabling conditions are satisfied and FENTRYR is not H'0000.
6 to 1	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
0	FENTRY0	0	R/W	Flash Memory P/E Mode Entry 0 Refer to section 20, Flash Memory.

Note: * Written data is not stored in these bits.

21.3.9 EEPROM Blank Check Control Register (EEPBCCNT)

EEPBCCNT specifies the address and size of the EEPROM area for blank check command. This register is initialized by setting the FRESET bit in FRESETR to 1 as well as applying a reset through the $\overline{\text{RES}}$ pin.

Bit	15	14	13	12	11	10	9	8
Bit Name	—	—	BCADR	BCADR	BCADR	BCADR	BCADR	BCADR
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	BCADR	BCADR	BCADR	BCADR	BCADR	—	—	BCSIZE
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R	R	R/W

Bit	Bit Name	Initial Value	R/W	Description
15, 14	—	All 0	R	Reserved These bits are always read as 0. Otherwise, operation cannot be guaranteed.
13 to 3	BCADR	All 0	R/W	Blank Check Address Setting These bits specify the address for blank checking by the blank check command when the size of the area for blank check is eight bytes (BCSIZE bit = 0). When BCSIZE is 0, the start address for blank checking is the setting value of EEPBCCNT (the value set in BCADR shifted toward MSB by 3 bits) added to the erasure block start address specified when a blank check command is issued.
2, 1	—	All 0	R	Reserved These bits are always read as 0. Otherwise, operation cannot be guaranteed.
0	BCSIZE	0	R/W	Blank Check Size Setting Specifies the size of the area for blank checking by the blank check command. 0: 8 bytes 1: 2 Kbytes

21.3.10 EEPROM Blank Check Status Register (EEPBCSTAT)

EEPBCSTAT is a register for storing the result of blank check command processing. This register is initialized by setting the FRESET bit in FRESETR to 1 as well as applying a reset through the $\overline{\text{RES}}$ pin.

Bit	15	14	13	12	11	10	9	8
Bit Name	—	—	—	—	—	—	—	—
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	BCST
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R

Bit	Bit Name	Initial Value	R/W	Description
15 to 1	—	All 0	R	Reserved These bits are always read as 0. Otherwise, operation cannot be guaranteed.
0	BCST	0	R	Blank Check Status Indicates the result of blank check. 0: Area specified for blank check is erased (blank) 1: 0 or 1 valued data have been written to the area specified for blank check

21.3.11 EEPROM MAT Select Register (EEPMAT)

EEPMAT switches memory MATs in the EEPROM.

Bit	15	14	13	12	11	10	9	8
Bit Name	EMKEY7	EMKEY6	EMKEY5	EMKEY4	EMKEY3	EMKEY2	EMKEY1	EMKEY0
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/(W)*1	R/(W)*1	R/(W)*1	R/(W)*1	R/(W)*1	R/(W)*1	R/(W)*1	R/(W)*1

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	EEPSEL
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R	R	R	R	R	R	R	R/W

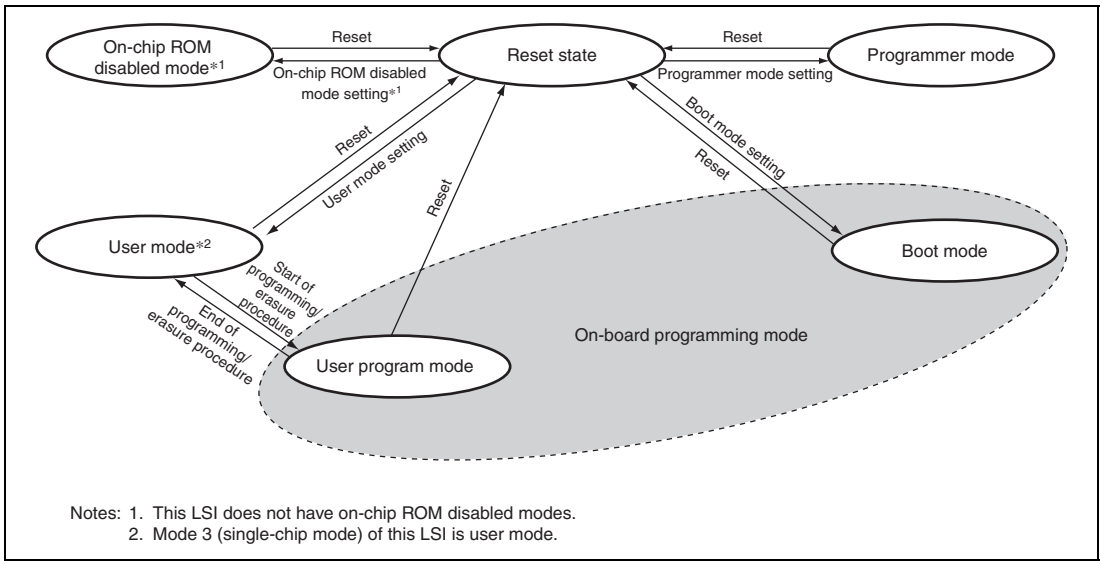
Note: * Written data is not stored in these bits.

Bit	Bit Name	Initial Value	R/W	Description
15 to 8	EMKEY7 to EMKEY0	All 0	R/(W)*1	Key Code These bits enable or disable EEPSEL bit modification. The data written to these bits are not stored.
7 to 1	—	All 0	R	Reserved These bits are always read as 0. The write value should always be 0.
0	EEPSEL	0	R/W	EEPROM MAT Select Selects a memory MAT in the EEPROM. Writing to this bit is enabled only when this register is accessed in word size and H'B3 is written to the EMKEY bits. 0: Selects the data MAT 1: Selects the product information MAT*2

Notes: 1. Written data is not stored in these bits.
2. The product information MAT is read-only memory and cannot be programmed or erased.

21.4 Overview of EEPROM-Related Modes

Figure 21.4 shows the EEPROM-related mode transition in this LSI. For the relationship between the operating modes of this LSI and the MD1 and MD0 pin settings, refer to section 3, MCU Operating Modes.



Notes: 1. This LSI does not have on-chip ROM disabled modes.
2. Mode 3 (single-chip mode) of this LSI is user mode.

Figure 21.4 EEPROM-Related Mode Transition

- The EEPROM cannot be read, programmed, or erased in on-chip ROM disabled mode or programmer mode. (This LSI does not support on-chip ROM disabled mode.)
- The data MAT can be read, programmed, and erased on the board in user mode, user program mode, and boot mode.
- In user mode, the flash memory cannot be programmed or erased but the EEPROM can be programmed and erased. While the EEPROM is being programmed or erased, the flash memory can be read. Therefore, the user can program the EEPROM while executing an application program in the flash memory protected against programming and erasure.

Table 21.4 compares programming- and erasure-related items for the boot mode, user mode, and user program mode.

Table 21.4 Comparison of Programming Modes

Item	Boot Mode	User Mode	User Program Mode
Programming/erasure environment	On-board programming	On-board programming	On-board programming*2
Programming/erasure enabled MAT	Data MAT	Data MAT	Data MAT
Programming/erasure control	Host	FCU	FCU
Entire area erasure	Available (automatic)	Available	Available
Block erasure	Available*1	Available	Available
Programming data transfer	From host via SCI	From any device via RAM	From any device via RAM
Reset-start MAT	Embedded program stored MAT	User MAT	User MAT

Notes: 1. The entire area is erased when the LSI is started. After that, a specified block can be erased.
2. In this LSI, user program mode is defined as a period between the start of the defined programming/erasure procedure in user mode and the end of the procedure. For the procedure of programming/erasure, see section 21.6, User Mode and User Program Mode.

- In boot mode, the user MAT in flash memory and the data MAT are all erased immediately after the LSI is started. The data MAT can then be programmed from the host via the SCI. The data MAT can also be read after this entire area erasure.
- In boot mode and programmer mode, on-chip RAM is used by the boot program. Accordingly, if the LSI is reset with RAM enable register (RAMEN) set for disabling on-chip RAM and then started in boot mode or programmer mode, the data in on-chip RAM before reset will be lost (see section 19, RAM).

21.5 Boot Mode

To program or erase the data MAT in boot mode, send control commands and programming data from the host. For the system configuration and settings in boot mode, refer to section 20, Flash Memory. This section describes only the commands dedicated for the EEPROM.

21.5.1 Inquiry/Selection Host Commands

Table 21.5 shows the inquiry/selection host commands dedicated to the EEPROM. The data MAT inquiry and data MAT information inquiry commands are used in the step for inquiry regarding the MAT programming information shown in figure 20.9 in section 20.5.4, Inquiry/Selection Host Command Wait State.

Table 21.5 Inquiry/Selection Host Commands (for EEPROM)

Host Command Name	Function
Data MAT inquiry	Inquires regarding the availability of data MAT
Data MAT information inquiry	Inquires regarding the number of data MATs and the start and end addresses

Each host command is described in detail below. The "command" in the description indicates a command sent from the host to this LSI and the "response" indicates a response sent from this LSI to the host. The "checksum" is byte-size data calculated so that the sum of all bytes to be sent by this LSI becomes H'00.

(1) Data MAT Inquiry

In response to a data MAT inquiry command sent from the host, this LSI returns the information concerning the availability of data MATs.

Command	H'2A			
Response	H'3A	Size	Availability	SUM

[Legend]

Size (1 byte): Total number of characters in the availability field (fixed at 1)

Availability (1 byte): Availability of data MATs (fixed at H'01)
H'00: No data MAT is available
H'01: Data MAT is available

SUM (1 byte): Checksum

(2) Data MAT Information Inquiry

In response to a data MAT information inquiry command sent from the host, this LSI returns the number of data MATs and their addresses.

Command

H'2B

Response

H'3B	Size	MAT count
MAT start address		
MAT end address		
MAT start address		
MAT end address		
:		
MAT start address		
MAT end address		
SUM		

[Legend]

Size (1 byte): Total number of bytes in the MAT count, MAT start address, and MAT end address fields

MAT count (1 byte): Number of data MATs (consecutive areas are counted as one MAT)

MAT start address (4 bytes): Start address of a data MAT

MAT end address (4 bytes): End address of a data MAT

SUM (1 byte): Checksum

The information concerning the block configuration in the data MAT is included in the response to the erasure block information inquiry command (refer to section 20.5.4, Inquiry/Selection Host Command Wait State).

21.5.2 Programming/Erasing Host Commands

Table 21.6 shows the programming/erasing host commands dedicated to the EEPROM. EEPROM-dedicated host commands are provided only for checksum and blank check; the programming, erasing, and reading commands are used in common for the flash memory and EEPROM.

To program the data MAT, issue from the host a user MAT programming selection command and then a 128-byte programming command specifying a data MAT address as the programming address. To erase the data MAT, issue an erasure selection command and then a block erasure command specifying an erasure block in the data MAT. The information concerning the erasure block configuration in the data MAT is included in the response to the erasure block information inquiry command. To read data from the data MAT, select the user MAT through a memory read command specifying a data MAT address as the read address.

For the user MAT programming selection, 128-byte programming, erasure selection, block erasure, and memory read commands, refer to section 20.5.5, Programming/Erasing Host Command Wait State. For the erasure block information inquiry command, refer to section 20.5.4, Inquiry/Selection Host Command Wait State.

Table 21.6 Programming/Erasure Host Commands (for EEPROM)

Host Command Name	Function
Data MAT checksum	Performs checksum verification for the data MAT
Data MAT blank check	Checks whether the data MAT is blank

Each host command is described in detail below. The "command" in the description indicates a command sent from the host to this LSI and the "response" indicates a response sent from this LSI to the host. The "checksum" is byte-size data calculated so that the sum of all bytes to be sent by this LSI becomes H'00.

(1) Data MAT Checksum

In response to a data MAT checksum command sent from the host, this LSI sums the data MAT data in byte units and returns the result (checksum).

Command	H'61			
Response	H'71	Size	MAT checksum	SUM

[Legend]

Size (1 byte): Number of bytes in the MAT checksum field (fixed at 4)

MAT checksum (4 bytes): Checksum of the data MAT data

SUM (4 bytes): Checksum (for the response data)

(2) Data MAT Blank Check

In response to a data MAT blank check command sent from the host, this LSI checks whether the data MAT is completely erased. When the data MAT is completely erased, this LSI returns a response (H'06). If the user MAT has an unerased area, this LSI returns an error response (sends H'E2 and H'52 in that order).

Command	H'62	
Response	H'06	
Error response	H'E2	H'52

21.6 User Mode and User Program Mode

21.6.1 FCU Command List

To program or erase the data MAT in user mode or user program mode, issue FCU commands to the FCU. Table 21.7 is a list of FCU commands for EEPROM programming and erasure.

Table 21.7 FCU Command List (EEPROM-Related Commands)

Command	Function
Normal mode transition	Moves to the normal mode (see section 21.6.2, Conditions for FCU Command Acceptance)
Status read mode transition	Moves to the status read mode (see section 21.6.2, Conditions for FCU Command Acceptance)
Lock bit read mode transition (lock bit read 1)	Moves to the lock bit read mode (see section 21.6.2, Conditions for FCU Command Acceptance)
Flash clock notification	Specifies the frequency of the flash clock
Program	Programs EEPROM (in 8-byte or 128-byte units)
Block erase	Erases EEPROM (in block units)
P/E suspend	Suspends programming or erasure
P/E resume	Resumes programming or erasure
Status register clear	Clears the ILGLERR, ERSERR, and PRGERR bits in FSTATR0 and cancels the command-locked state
Blank check	Finds out whether the specified area is erased (blank)

The FCU commands in table 21.7 other than the blank check command are also used in flash memory programming/erasure. If a blank check command is issued to flash memory, lock bit reading from flash memory is executed.

Issuance of a command to the FCU takes place as a sequence of write access to an EEPROM area via the peripheral bus. Table 21.8 shows the FCU command formats for the program commands and blank check command. For the formats of FCU commands other than the blank check command, refer to section 21.6.1, FCU Command List. Performing peripheral bus accesses as shown in table 21.8 under specified conditions causes the FCU to start processing corresponding to the command. For the conditions for program command acceptance, refer to section 21.6.2, Conditions for FCU Command Acceptance. For details of command usage, refer to section 21.6.3, FCU Command Usage.

When H'71 is sent in the first cycle of an FCU command while the FRDMD bit is 0 (memory area read mode), the FCU accepts the lock bit read mode transition command (lock bit read 1). Since there are no lock bits in EEPROM, undefined data will be read if an EEPROM area is read through the peripheral bus after transition to the lock bit read mode. The FCU does not detect this reading of undefined data as an error.

When H'71 is sent in the first cycle of the FCU command while the FRDMD bit is 1 (register read mode), the FCU waits for the second-cycle data (H'D0) of the blank check command. When H'D0 is written to an EEPROM area through the peripheral bus in this state, the FCU executes a blank check according to the setting of the EEPBCCNT register and places the result in the EEPBCSTAT register upon completion of the blank check.

There are two suspending modes to be initiated by the P/E suspend command; the suspension-priority mode and erasure-priority mode. For details of each mode, refer to section 20.6.4, Suspending Operation.

Table 21.8 Program Command and Blank Check Command Formats

Command	Number of Bus Cycles	First Cycle		Second Cycle		Third Cycle		Fourth Cycle to Cycle N + 2		Cycle N + 3	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Program (8-byte programming: N = 4)	7	EA	H'E8	EA	H'04	WA	WD1	EA	WDn	EA	H'D0
Program (128-byte programming: N = 64)	67	EA	H'E8	EA	H'40	WA	WD1	EA	WDn	EA	H'D0
Blank check	2	EA	H'71	BA	H'D0	—	—	—	—	—	—

[Legend]

EA: EEPROM area address
Any address from H'E00000 to H'E07FFF

WA: EEPROM program start address
8-byte programming: Start address of 8-byte programming data
128-byte programming: Start address of 128-byte programming data

BA: EEPROM erasure block address
Any address in the erasure block to be erased (specified as a programming/erasure address)

WDn: n-th word of programming data (n = 1 to N)

21.6.2 Conditions for FCU Command Acceptance

The FCU determines whether to accept a command depending on the FCU mode or status. Figure 21.5 is an FCU mode transition diagram.

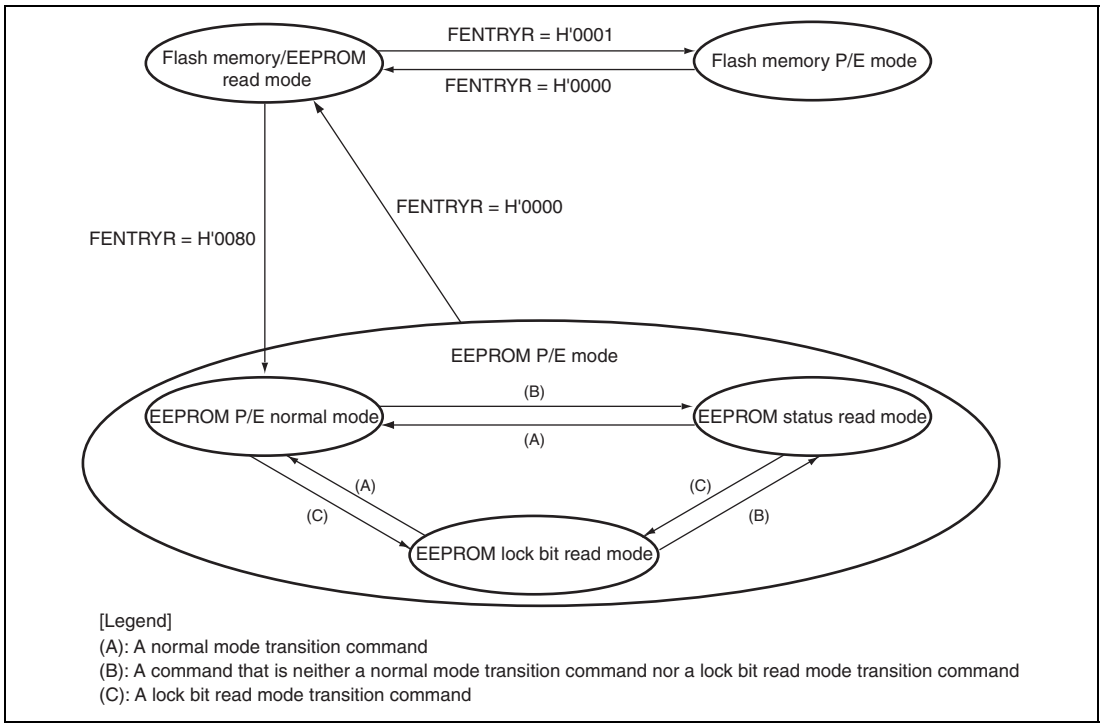


Figure 21.5 FCU Mode Transition Diagram (EEPROM-Related Modes)

(1) Flash Memory P/E Mode

The FCU can accept flash memory programming and erasing commands in this mode. The EEPROM cannot be read. The FCU enters this mode when the FENTRYD bit is cleared to 0 and the FENTRY0 bit is set to 1 in FENTRYR. For details of this mode, refer to section 21.6.2, Conditions for FCU Command Acceptance.

(2) Flash Memory/EEPROM Read Mode

The EEPROM can be read through the peripheral bus, and the flash memory can be read through the internal bus at a high speed. The FCU does not accept commands. The FCU enters this mode when the FENTRYD and FENTRY0 bits are both cleared to 0.

(3) EEPROM P/E Mode

- EEPROM P/E normal mode

  The FCU enters this mode when the FENTRYD bit is set to 1 and the FENTRY0 bit is cleared to 0 in flash memory/EEPROM read mode or flash memory P/E mode, or when a normal mode transition command is accepted in EEPROM P/E mode. Table 21.9 shows the commands that can be accepted in this mode. If the EEPROM area is read through the peripheral bus, an EEPROM access error occurs and the FCU enters the command-locked state. The flash memory can be read at a high speed.
- EEPROM status read mode

  The FCU enters this mode when the FCU accepts a command that is neither a normal mode transition command nor a lock bit read mode transition command in EEPROM P/E mode. The EEPROM status read mode includes the state in which the FRDY bit in FSTATR0 is 0 and the command-locked state after an error has occurred. Table 21.9 shows the commands that can be accepted in this mode. If the EEPROM area is read through the peripheral bus, the FSTATR0 value is read. The flash memory can be read at a high speed.
- EEPROM lock bit read mode

  The FCU enters this mode when the FCU accepts a lock bit read mode transition command in EEPROM P/E mode. Table 21.9 shows the commands that can be accepted in this mode. Since EEPROM does not contain any lock bits, undefined data will be read if the EEPROM area is read through the peripheral bus; but this does not result in an EEPROM access error. The flash memory can be read at a high speed.

Table 21.9 shows the acceptable commands in each FCU mode/state. When a command that cannot be accepted is issued, the FCU enters the command-locked state (see section 21.7.3, Error Protection).

To make sure that the FCU accepts a command, enter the mode in which the FCU can accept the target command, check the values of bits FRDY, ILGLERR, ERSERR, and PRGERR in FSTATR0 and bits FCUERR, FRDTCT, and FRCRCT in FSTATR1, and then issue the target FCU command. The CMDLK bit in FASTAT holds a value obtained by ORing the IRGLERR, ERSERR, and PRGERR bits in FSTATR0 and bits FCUERR, FRDTCT, and FRCRCT in FSTATR1. Accordingly, the state of the FCU can be checked from the value of the CMDLK bit.

In table 21.9, whether an error has occurred or not is determined from the CMDLK bit.

While the programming/erasure processing, programming/erasure suspension processing, or blank check processing is in progress, the FRDY bit in FSTATR0 is 0. While the FRDY bit is 0, the P/E suspend command can be accepted only when the SUSRDY bit in FSTATR0 is 1.

In table 21.9, the values of bits ERSSPD, PRGSPD, and FRDY are indicated as “0/1” to make the table simpler. The ERSSPD bit is 1 during erasure-suspension processing and 0 during programming-suspension processing. The PRGSPD bit is 1 during programming-suspension processing and 0 during erasure-suspension processing. In a command-locked state, the FRDY bit retains the value before the command-locked state is entered.

Table 21.9 FCU Modes/States and Acceptable Commands

Item	P/E Normal Mode			Status Read Mode									Lock Bit Read Mode		
	Programming-Suspended	Erasure-Suspended	Other State	Programming/Erasure Processing	Programming Processing while Erasure-Suspended	Programming/Erasure Suspension Processing	Blank Check Processing	Programming-Suspended	Erasure-Suspended	Command Locked (FRDY = 0)	Command Locked (FRDY = 1)	Other State	Programming-Suspended	Erasure-Suspended	Other State
FRDY bit in FSTATR0	1	1	1	0	0	0	0	1	1	0	1	1	1	1	1
SUSRDY bit in FSTATR0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
ERSSPD bit in FSTATR0	0	1	0	0	1	0/1	0/1	0	1	0/1	0/1	0	0	1	0
PRGSPD bit in FSTATR0	1	0	0	0	0	0/1	0/1	1	0	0/1	0/1	0	1	0	0
CMDLK bit in FASTAT	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
Normal mode transition	A	A	A	×	×	×	×	A	A	×	×	A	A	A	A
Status read mode transition	A	A	A	×	×	×	×	A	A	×	×	A	A	A	A
Lock bit read mode transition (lock bit read 1)	A	A	A	×	×	×	×	A	A	×	×	A	A	A	A
Flash clock notification	×	×	A	×	×	×	×	×	×	×	×	A	×	×	A
Program	×	*	A	×	×	×	×	×	*	×	×	A	×	*	A
Block erase	×	×	A	×	×	×	×	×	×	×	×	A	×	×	A
P/E suspend	×	×	×	A	×	×	×	×	×	×	×	×	×	×	×
P/E resume	A	A	×	×	×	×	×	A	A	×	×	×	A	A	×
Status register clear	A	A	A	×	×	×	×	A	A	×	A	A	A	A	A
Blank check	A	A	A	×	×	×	×	A	A	×	×	A	A	A	A

[Legend]

A: Acceptable

*: Only programming is acceptable for the areas other than the erasure-suspended block

×: Not acceptable

21.6.3 FCU Command Usage

This section shows how to program and erase EEPROM using the program command and block erase command, respectively and how to check if the EEPROM has been erased using the blank check command. For the firmware transfer to the FCU RAM and the other FCU command usage, refer to section 20.6.3, FCU Command Usage.

If the FCUERR, FRDTCT, or FRCRCT bit is set to 1 while the FCU is processing a command and the FCU is thus placed in the command-locked state, the FRDY bit holds 0. Since the FCU stops processing in the command-locked state, the FRDY bit will not be changed from 0 to 1. If the FRDY bit stays 0 for a period longer than the programming/erasure time or suspend delay time (see section 25, Electrical Characteristics), an abnormal condition may have occurred such that FCU processing has been stopped in the command-locked state. In such cases, reset the FCU for initialization. When command processing is completed with the FRDY bit set to 1, the FCUERR, FRDTCT, and FRCRCT bits are always both 0. Accordingly, using bits ILGLERR, ERSERR, and PRGERR is enough for checking the error state after completion of command processing.

(1) Using Flash Clock Notification Command

Before data is written to or erased from flash memory, the current flash clock should be specified. For details, refer to section 20.6.3, FCU Command Usage. Set the FENTRYD bit in FENTRYR to 1 and specify an address in the data flash area in the command.

(2) Programming

To program the EEPROM, use the program command. Write byte H'E8 to an EEPROM area address in the first cycle of the program command and the number of words (N)* to be programmed through byte access in the second cycle. Access the peripheral bus in words from the third cycle to cycle N + 2 of the command. In the third cycle, write the programming data to the start address for EEPROM programming. Here, the start address must be an 8-byte boundary address for 8-byte programming or a 128-byte boundary address for 128-byte programming. After writing words to EEPROM area addresses N − 1 times, write byte H'D0 to an EEPROM area address in cycle N + 3; the FCU then starts EEPROM programming. Read the FRDY bit in FSTATR0 to confirm that EEPROM programming is completed.

If the area accessed in the third cycle to cycle N + 2 includes addresses that do not need to be programmed, write H'FFFF as the programming data for those addresses. To ignore the programming and erasure protection provided by the EEPWE0 and EEPWE1 settings, set the program/erase enable bit for the target block to 1 before starting programming.

Figure 21.6 shows the procedure for EEPROM programming

Note: * N = H'04 for 8-byte programming or N = H'40 for 128-byte programming.

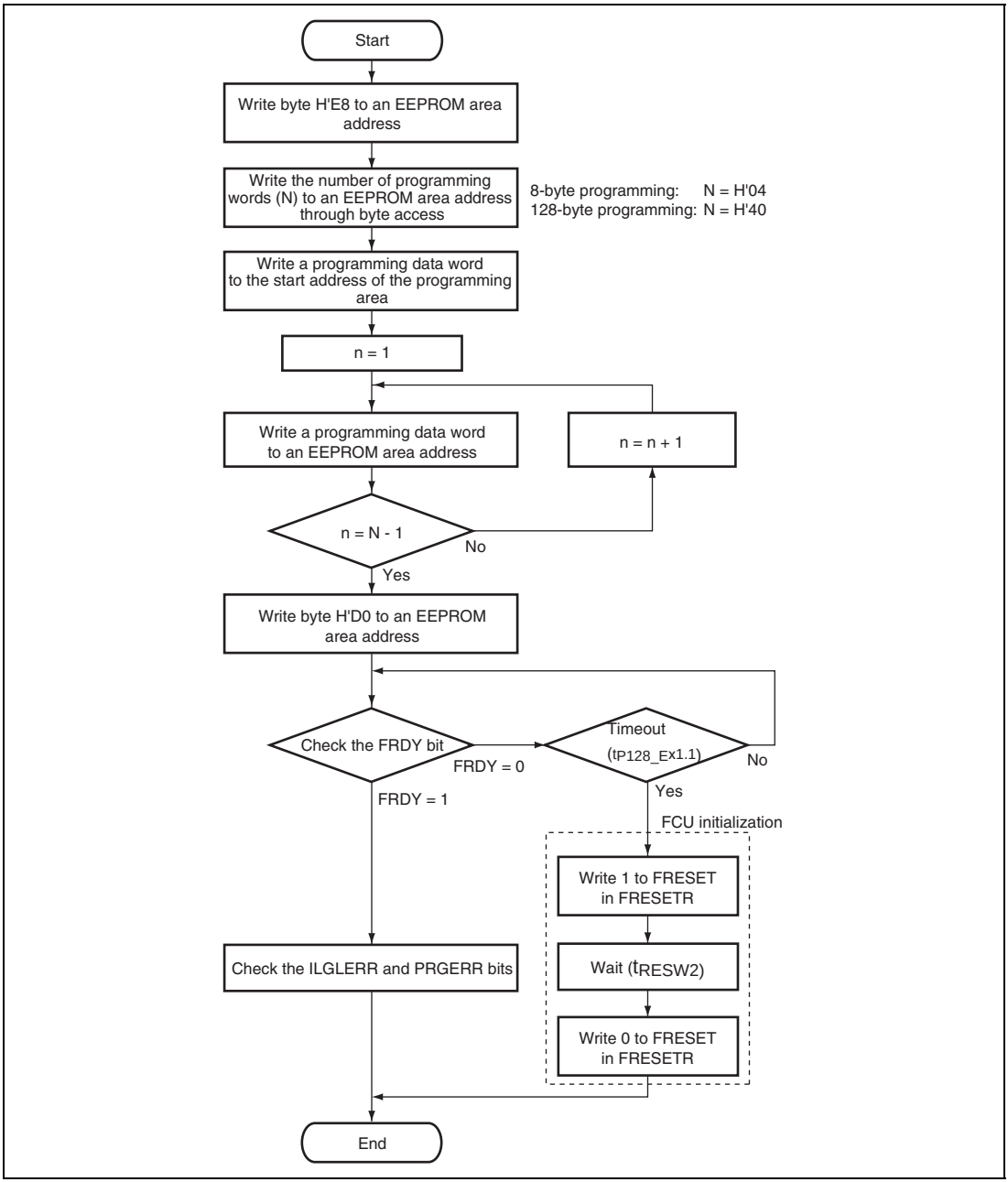


Figure 21.6 Procedure for EEPROM Programming

(3) Erasure

To erase the EEPROM, use the block erase command. The EEPROM can be erased in the same way as flash memory erasure (refer to section 20, Flash Memory). Note that the EEPROM has a programming and erasure protection function through EEPWE0 and EEPWE1. To ignore the programming and erasure protection provided by the EEPWE0 and EEPWE1 settings, set the program/erase enable bit for the target block to 1 before starting erasure.

(4) Blank Check

Since undefined values are read if the CPU reads from erased EEPROM, use the blank check command to find out whether the EEPROM is erased or not. Before using the blank check command, set the FRDMD bit in FMODR to 1 to allow the use of the blank check command and set the size and address of the area for blank check in the EEPBCCNT register. Setting the BCSIZE bit in EEPBCCNT to 1 enables blank checking of the entire erasure block (2 Kbytes) specified in the second cycle of the blank check command. Clearing the BCSIZE bit to 0 selects blank checking of the eight bytes from the address obtained by adding the address value set in the EEPBCCNT register to the start address of the erasure block specified in the second cycle of the blank check command.

In the first cycle of the blank check command, write H'71 in a byte access to an EEPROM area address. The FCU starts EEPROM blank check processing when H'D0 is written to any address in the erasure block containing the area to be checked in the second cycle of the command. Completion of the blank check is indicated in the FRDY bit of the FSTATR0 register. After the blank check is completed, read the BCST bit in the EEPBCSTAT register to see if the area that has been checked is erased or either 0 or 1 valued data is programmed. Figure 21.7 shows the procedure for performing blank checking of the EEPROM.

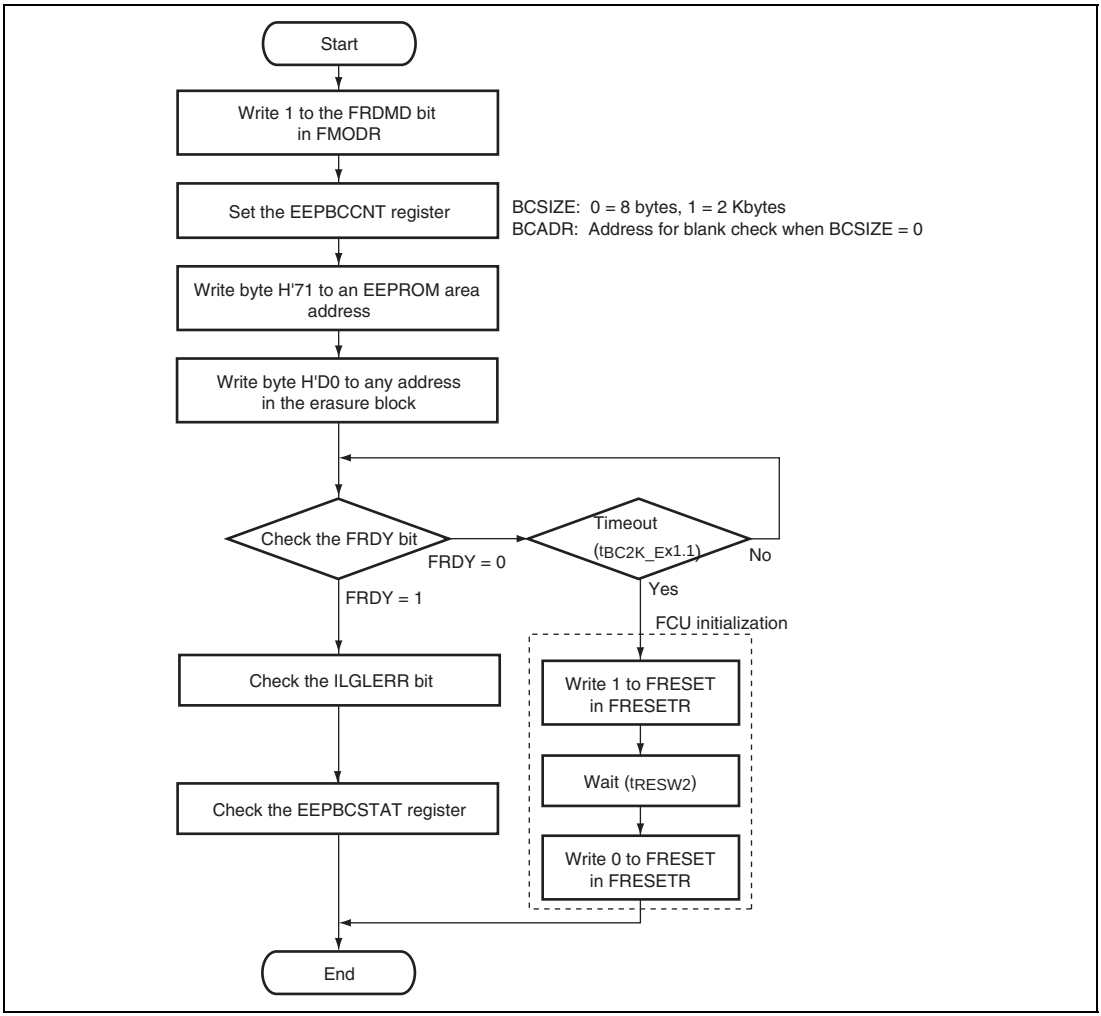


Figure 21.7 Procedure for EEPROM Blank Check

21.7 Protection

There are three types of EEPROM programming/erasure protection: hardware, software, and error protection.

21.7.1 Hardware Protection

The hardware protection function disables EEPROM programming and erasure according to the mode pin settings in this LSI.

While the on-chip ROM is disabled, EEPROM programming, erasing, and reading are disabled. (This LSI does not support on-chip ROM disabled mode.) For the operating modes set through the mode pins of this LSI, refer to section 3, MCU Operating Modes.

21.7.2 Software Protection

The software protection function disables EEPROM programming and erasure according to the control register settings. If an attempt is made to issue a programming or erasing command to the EEPROM against software protection, the FCU detects an error and enters command-locked state.

(1) Protection through FENTRYR

When the FENTRYD bit in FENTRYR is 0, the FCU does not accept commands for the EEPROM, so EEPROM programming and erasure are disabled. If an attempt is made to issue an FCU command for the EEPROM while the FENTRYD bit is 0, the FCU detects an illegal command error and enters command-locked state (see section 21.7.3, Error Protection).

(2) Protection through EEPWE0 and EEPWE1

When the DBWEi (i = 00 to 15) bit in EEPWE0 or EEPWE1 is 0, programming and erasure of block DBi in the data MAT are disabled. If an attempt is made to program or erase block DBi while the DBWEi bit is 0, the FCU detects a program/erase protect error and enters command-locked state (see section 21.7.3, Error Protection).

21.7.3 Error Protection

The error protection function detects an illegal FCU command issued, an illegal access, or an FCU malfunction, and disables FCU command acceptance (command-locked state). While the FCU is in command-locked state, the EEPROM cannot be programmed or erased. To cancel command-locked state, issue a status register clear command while FASTAT is H'10.

While the CMDLKIE bit in FAEINT is 1, a flash interface error (FIFE) interrupt is generated if the FCU enters command-locked state (the CMDLK bit in FASTAT becomes 1). While an EEPROM-related interrupt enable bit (EEPAEIE, EEPRPEIE, or EEPWPEIE) in FAEINT is 1, an FIFE interrupt is generated if the corresponding status bit (EEPAE, EEPRPE, or EEPWPE) in FASTAT becomes 1.

Table 21.10 shows the error protection types dedicated for the EEPROM and the status bit values (the ILGLERR, ERSERR, and PRGERR bits in FSTATR0 and the EEPAE, EEPRPE, and EEPWPE bits in FASTAT) after each error detection. For the error protection types used in common by the flash memory and EEPROM (FENTRYR setting error, most of illegal command errors, erasing error, programming error, FCU error, and FCU RAM ECC error), refer to section 21.7.3, Error Protection.

If the FCU enters command-locked state due to a command other than a suspend command issued during programming or erasure processing, the FCU continues programming or erasing the EEPROM. In this state, the P/E suspend command cannot suspend programming or erasure. If a command is issued in command-locked state, the ILGLERR bit becomes 1 and the other bits retain the values set due to the previous error detection.

Table 21.10 Error Protection Types

Error	Description	ILGLERR	ERSERR	PRGERR	EEPAE	EEPRPE	EEPWPE
Illegal command error	The value specified in the second cycle of a program command is neither H'04 nor H'40.	1	0	0	0	0	0
	A lock bit program command has been issued to the EEPROM area while FENTRYD = 1 in FENTRYR	1	0	0	0	0	0
EEPROM access error	A read access command has been issued to the EEPROM area while FENTRYD = 1 in FENTRYR in EEPROM P/E normal mode.	1	0	0	1	0	0
	A write access command has been issued to the EEPROM area while FENTRYD = 0.	1	0	0	1	0	0
	An access command has been issued to the EEPROM area while FENTRY0 in FENTRYR is 1.	1	0	0	1	0	0
EEPROM read protect error	A read access command has been issued to the EEPROM area protected against reading through EEPRE0 and EEPRE1.	1	0	0	0	1	0
EEPROM program/erase protect error	A program command or block erase command has been issued to the EEPROM area protected against programming or erasure through EEPWE0 and EEPWE1.	1	0	0	0	0	1

21.8 Product Information MAT

The product information MAT stores the device name, device revision number, and embedded program revision number information in ASCII code. The embedded program is stored in the reset-start MAT used in boot mode and programmer mode (refer to section 20.4, Overview of Flash Memory-Related Modes). Tables 21.11 and 21.12 show the addresses to store the information and examples of information data. In the product information MAT (H'E0,0000 to H'E0,007F), the addresses not shown in this table are reserved areas. Undefined data will be read from the reserved areas.

Table 21.11 Data Stored in Product Information MAT (H8SX/1727S)

Information	Address	Example of Data
Device name	H'E0,0000 to H'E0,0008	H'523546363137323753 = R5F61727S
Device revision number	H'E0,0010 to H'E0,0011	H'3031 = 01
Embedded program revision number	H'E0,0020 to H'E0,0022	H'313030 = 100 (1.00)

Table 21.12 Data Stored in Product Information MAT (H8SX/1725S)

Information	Address	Example of Data
Device name	H'E0,0000 to H'E0,0008	H'523546363137323553 = R5F61725S
Device revision number	H'E0,0010 to H'E0,0011	H'3031 = 01
Embedded program revision number	H'E0,0020 to H'E0,0022	H'313030 = 100 (1.00)

21.9 Usage Notes

(1) Protection of Data MAT Immediately after a Reset

As the initial value of EEPRE0, EEPRE1, EEPWE0, and EEPWE1 is H'0000, data MAT programming, erasure, and reading are disabled immediately after a reset. To read data from the data MAT, set EEPRE0 and EEPRE1 appropriately before accessing the data MAT. To program or erase the data MAT, set EEPWE0 and EEPWE1 appropriately before issuing an FCU command for programming or erasure. If an attempt is made to read, program, or erase the data MAT without setting the registers, the FCU detects an error and enters command-locked state.

(2) State in which Interrupts are Ignored

In the following state, NMI and maskable interrupt requests are ignored.

- Boot mode in operation

(3) Programming-/Erasure-Suspended Area

The data stored in the programming-suspended or erasure-suspended area is undetermined. To avoid malfunction due to undefined read data, ensure that no data is read from the programming-suspended or erasure-suspended area.

(4) Compatibility with Programming/Erasing Program of Conventional F-ZTAT H8SX Microcomputers

The flash memory programming/erasing program used for conventional F-ZTAT H8SX microcomputers does not work with this LSI.

(5) Reset during Programming or Erasure

When resetting the FCU by setting the FRESET bit in FRESETR during programming or erasure, hold the reset state for t_{RESW2}. As a high voltage is applied to the EEPROM during programming or erasure, the FCU must be kept in the reset state over a long enough time to secure falling of the applied voltage. Do not attempt to read from EEPROM while the FCU is being reset.

When this LSI is reset by asserting the $\overline{\text{RES}}$ pin during programming or erasure, hold the reset state for t_{RESW2}. In this reset, not only the time required for the voltage applied to EEPROM to drop but also the time required for initialization of EEPROM power supply and its internal circuitry must be secured.

Ensure that an internal reset is not caused by overflow of the WDT counter during programming or erasure. This is because, in the reset caused by the WDT, the time required for the voltage on memory to fall and for initialization of the EEPROM power supply and within the EEPROM cannot be secured.

When a reset is initiated by assertion of the signal on the $\overline{\text{RES}}$ pin or the FCU is reset by setting of the FRESET bit in FRESETR during processing for programming or erasure, data in the target region for programming or erasure become undefined.

(6) Prohibition of Additional Programming

One area cannot be programmed twice in succession. To program an area that has already been programmed, be sure to erase the area before reprogramming.

(7) Programming to the Product Information MAT

The product information MAT is read-only memory and cannot be programmed or erased. If an attempt is made to issue a programming or an erasing command immediately after the EEPSEL bit in the EEPMAT register is set to 1, programming and erasing to the data MAT is executed. No EEPROM access error occurs. Do not attempt to write to or erase the product information MAT.

(8) Suspending Programming or Erasure

If processing for programming or erasure is suspended by issuing the programming/erasure suspend command, be sure to issue the resume command so that the operation is completed.

Section 22 Clock Pulse Generator

This LSI has a clock pulse generator (CPG) that generates the system clock (Iϕ), peripheral module clock (Pϕ), external bus clock (Bϕ), FLASH clock (Fϕ), A/D clock (Aϕ) and RSPI clock (Rϕ).

The clock pulse generator consists of an oscillator, oscillation stoppage detection circuit, internal oscillation circuit, main clock divider, PLL (phase locked loop) circuit, and selector circuit. Figure 22.1 shows a block diagram of the clock pulse generator.

Clock frequencies can be changed by the PLL circuit and main clock divider in the CPG. Changing the system clock control register 0 (SCKCR0) and system clock control register 1 (SCKCR1) settings by software can change the clock frequencies.

This LSI supports a system clock provided to the CPU and bus masters, a peripheral module clock provided to the peripheral modules, an external bus clock provided to the external bus, a FLASH clock provided to the FLASH memory controller, an A/D clock provided to the A/D, and an RSPI clock provided to the RSPI.

The system clock, peripheral module clock, external bus clock, FLASH clock, A/D clock, and RSPI clock can be specified independently. Note, however, that the frequencies of the peripheral module clock, external bus clock, FLASH clock, A/D clock, and RSPI clock operate only when these clocks are lower than that of the system clock.

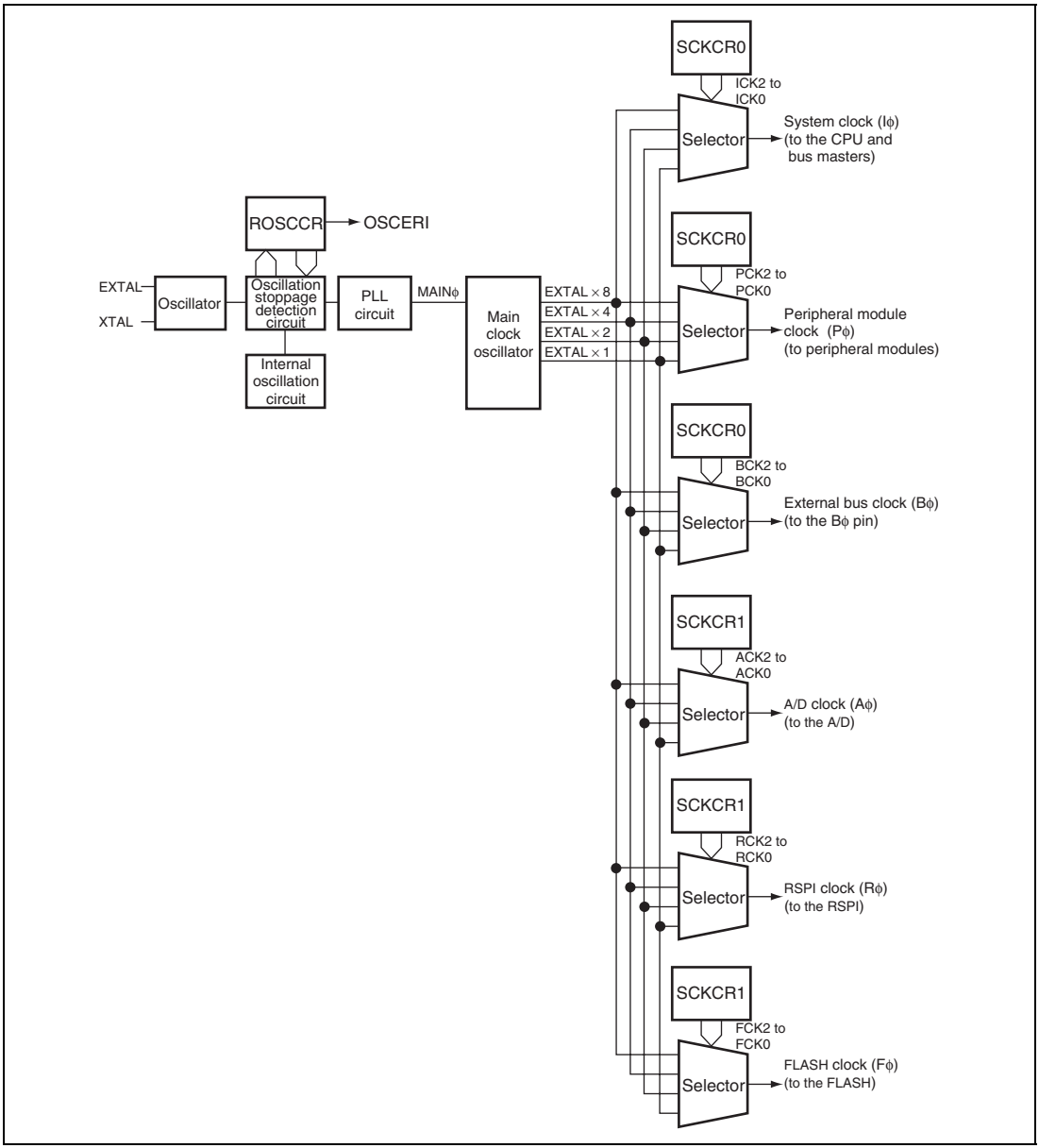


Figure 22.1 Block Diagram of Clock Pulse Generator

Table 22.1 Selection of Clock Pulse Generator (Recommended Settings)

EXTAL Input Clock Frequency	Iϕ	Pϕ	Bϕ	Fϕ	Aϕ	Rϕ
8 MHz	64 MHz (x8)	32 MHz (x4)	16 MHz (x2)	32 MHz (x4)	32 MHz (x4)	32 MHz (x4)
		16 MHz (x2)	16 MHz (x2)	16 MHz (x2)	16 MHz (x2)	16 MHz (x2)
10 MHz	80 MHz (x8)	40 MHz (x4)	20 MHz (x2)	40 MHz (x4)	40 MHz (x4)	40 MHz (x4)
		20 MHz (x2)	20 MHz (x2)	20 MHz (x2)	20 MHz (x2)	20 MHz (x2)
	40 MHz (x4)	40 MHz (x4)	20 MHz (x2)	40 MHz (x4)	40 MHz (x4)	40 MHz (x4)
		20 MHz (x2)	20 MHz (x2)	20 MHz (x2)	20 MHz (x2)	20 MHz (x2)

22.1 Register Description

The clock pulse generator has the following registers.

- System clock control register 0 (SCKCR0)
- System clock control register 1 (SCKCR1)
- Recovery oscillator control register (ROSCCR)

22.1.1 System Clock Control Register 0 (SCKCR0)

SCKCR0 controls Bϕ clock output and frequencies of the system, peripheral module, and external bus clocks, and selects the Bϕ clock to be output.

Bit	15	14	13	12	11	10	9	8
Bit Name	PSTOP1	—	POSEL1	—	—	ICK2	ICK1	ICK0
Initial Value:	0	0	0	0	0	0	1	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	—	PCK2	PCK1	PCK0	—	BCK2	BCK1	BCK0
Initial Value:	0	0	1	0	0	0	1	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
15	PSTOP1	0	R/W	Bϕ Output Select Enable Controls ϕ output on PA7. • Normal operation 0: Bϕ output 1: Fixed high • Software standby mode X: Fixed high • Hardware standby mode* X: Hi-Z Note: This LSI does not have hardware standby mode.
14	—	0	R/W	Reserved This bit is always read as 0. The write value should always be 0.
13	POSEL1	0	R/W	B ϕ Output Select 1 Controls the B ϕ output on PA7. 0: External bus clock (Bϕ) 1: Setting prohibited
12, 11	—	All 0	R/W	Reserved These bits are always read as 0. The write value should always be 0.
10	ICK2	0	R/W	System Clock (Iϕ) Select
9	ICK1	1	R/W	These bits select the frequency of the system clock provided to the CPU, DMAC, and DTC modules. The ratio to the input clock is as follows: 000: ×8 001: ×4 010: ×2 011: ×1 1XX: Setting prohibited The frequency of the peripheral module clock changes to the same frequency as the system clock if the frequency of the system clock is lower than that of the peripheral module clock, external clock, A/D clock, and RSPI clock.
8	ICK0	0	R/W	

Bit	Bit Name	Initial Value	R/W	Description
7	—	0	R/W	Reserved This bit is always read as 0. The write value should always be 0.
6	PCK2	0	R/W	Peripheral Module Clock (Pϕ) Select
5	PCK1	1	R/W	These bits select the frequency of the peripheral module clock. The ratio to the input clock is as follows:
4	PCK0	0	R/W	000: ×8 001: ×4 010: ×2 011: ×1 1XX: Setting prohibited The frequency of the peripheral module clock should be lower than that of the system clock. Though these bits can be set so as to make the frequency of the peripheral module clock higher than that of the system clock, the clock will have the same frequency in reality.
3	—	0	R/W	Reserved This bit is always read as 0. The write value should always be 0.
2	BCK2	0	R/W	External Bus Clock (Bϕ) Select
1	BCK1	1	R/W	These bits select the frequency of the external bus clock. The ratio to the input clock is as follows:
0	BCK0	0	R/W	000: ×8 001: ×4 010: ×2 011: ×1 1XX: Setting prohibited The frequency of the external bus clock should be lower than that of the system clock. Though these bits can be set so as to make the frequency of the external bus clock higher than that of the system clock, the clock will have the same frequency.

Note: X: Don't care

22.1.2 System Clock Control Register 1 (SCKCR1)

SCKCR1 controls frequencies of the A/D clock and RSPI clock.

Bit	15	14	13	12	11	10	9	8
Bit Name	—	—	—	—	—	FCK2	FCK1	FCK0
Initial Value:	0	0	0	0	0	0	1	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	—	ACK2	ACK1	ACK0	—	RCK2	RCK1	RCK0
Initial Value:	0	0	1	0	0	0	1	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
15 to 11	—	All 0	R/W	Reserved These bits are always read as 0. The write value should always be 0.
10	FCK2	0	R/W	FLASH Clock (Fϕ) Select
9	FCK1	1	R/W	These bits select the frequency of the FLASH clock. The ratio to the input clock is as follows:
8	FCK0	0	R/W	000: ×8 001: ×4 010: ×2 011: ×1 1XX: Setting prohibited The frequency of the FLASH clock should be lower than that of the system clock. Though these bits can be set so as to make the frequency of the FLASH clock higher than that of the system clock, the clock will have the same frequency.
7	—	0	R/W	Reserved This bit is always read as 0. The write value should always be 0.

Bit	Bit Name	Initial Value	R/W	Description
6	ACK2	0	R/W	A/D Clock (Aϕ) Select
5	ACK1	1	R/W	These bits select the frequency of the A/D clock. The ratio to the input clock is as follows:
4	ACK0	0	R/W	000: ×8 001: ×4 010: ×2 011: ×1 1XX: Setting prohibited The frequency of the A/D clock should be lower than that of the system clock. Though these bits can be set so as to make the frequency of the A/D clock higher than that of the system clock, the clock will have the same frequency.
3	—	0	R/W	Reserved This bit is always read as 0. The write value should always be 0.
2	RCK2	0	R/W	RSPI Clock (Rϕ) Select
1	RCK1	1	R/W	These bits select the frequency of the RSPI clock. The ratio to the input clock is as follows:
0	RCK0	0	R/W	000: ×8 001: ×4 010: ×2 011: ×1 1XX: Setting prohibited The frequency of the RSPI clock should be lower than that of the system clock. Though these bits can be set so as to make the frequency of the RSPI clock higher than that of the system clock, the clock will have the same frequency.

Note: X: Don't care

22.1.3 Recovery Oscillator Control Register (ROSCCR)

ROSCCR controls the external oscillation stoppage detection function.

Bit	15	14	13	12	11	10	9	8
Bit Name	INOSCE	OSCERR	OSCIE	—	—	—	—	ERRTEST
Initial Value:	1	0	0	0	0	0	0	0
R/W:	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	—
Initial Value:	0	0	0	0	0	0	0	0
R/W:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
15	INOSCE	1	R/W	External Oscillation Stoppage Detection Function Enable Controls the operation of the external oscillation stoppage detection function. 0: Disables the external oscillation stoppage detection function 1: Enables the external oscillation stoppage detection function
14	OSCERR	0	R	External Oscillation Status Flag Status flag that indicates the state of external oscillation. When INOSCE = 1 0: The external oscillator is operating normally 1: The external oscillator is stuck at 0 or 1. When INOSCE = 0 This bit is always read as 0. This bit is a read-only bit and cannot be modified.
13	OSCIE	0	R/W	External Oscillation Stoppage Detection Interrupt Enable Enables or disables the generation of interrupt requests when the OSCERR flag is set to 1. 0: Disables the generation of interrupts by OSCERR 1: Enables the generation of interrupts by OSCERR

Bit	Bit Name	Initial Value	R/W	Description
12 to 9	—	All 0	R/W	Reserved These bits are always read as 0. The write value should always be 0.
8	ERRTEST	0	R/W	Oscillation Stoppage Detection Test Forcibly sets the external oscillation detection flag (OSCERR) to 1. 0: The value of the OSCERR flag is retained. 1: The OSCERR flag is set to 1.
7 to 0	—	All 0	R/W	Reserved These bits are always read as 0. The write value should always be 0.

22.2 Oscillator

Clock pulses can be supplied by connecting a crystal resonator or by input of an external clock.

22.2.1 Connecting Crystal Resonator

A crystal resonator can be connected as shown in the example in figure 22.2. Select the damping resistance R_d according to table 22.2.

When the clock is provided by connecting a crystal resonator, a crystal resonator having a frequency of 8 to 10 MHz should be connected.

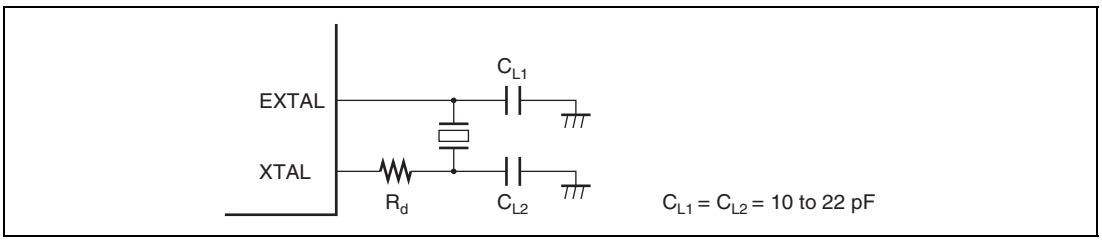


Figure 22.2 Connection of Crystal Resonator (Example)

Table 22.2 Damping Resistance Value

Frequency (MHz)	8	10
R_d (Ω)	200	0

Figure 22.3 shows an equivalent circuit of the crystal resonator. Use a crystal resonator that has the characteristics shown in table 22.3.

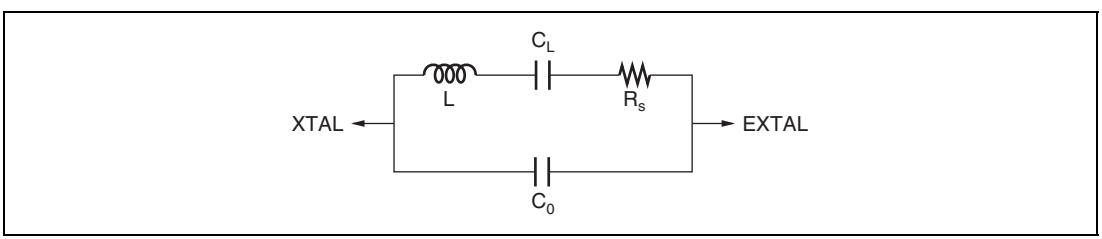


Figure 22.3 Crystal Resonator Equivalent Circuit

Table 22.3 Crystal Resonator Characteristics

Frequency (MHz)	8	10
R_S Max. (Ω)	80	70
C_0 Max. (pF)	7	

22.2.2 External Clock Input

An external clock signal can be input as shown in the examples in figure 22.4. If the XTAL pin is left open, be sure that parasitic capacitance is no more than 10 pF. When the counter clock is input to the XTAL pin, be sure that the external clock is held high in standby mode.

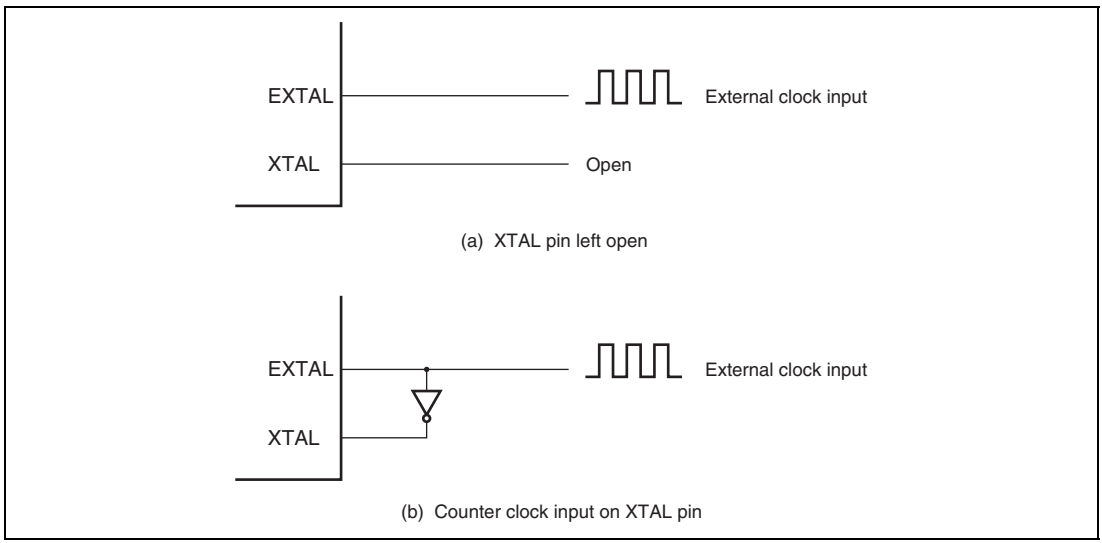


Figure 22.4 External Clock Input (Examples)

For the input conditions of the external clock, refer to section 25.3.1, Clock Timing. The input external clock should be from 8 to 10 MHz.

22.3 PLL Circuit

The PLL circuit has the function of multiplying the frequency of the clock from the oscillator by a factor of 8. The frequency multiplication factor is fixed. In this case, the phase of the rising edge of the internal clock is controlled to match the phase of the rising edge of EXTAL.

22.4 Main Clock Divider

The main clock divider divides the PLL clock to generate a 1/2, 1/4, or 1/8 clock. After bits ICK2 to ICK0, PCK2 to PCK0, BCK2 to BCK0, FCK2 to FCK0, ACK2 to ACK0, and RCK2 to RCK0 are modified, this LSI operates at the modified frequency.

22.5 External Oscillation Stoppage Detection Function

22.5.1 Overview

The external oscillation stoppage detection circuit sets the OSCERR bit in ROSCCR to 1 (the initial value is 0) in cases of abnormal operation where external oscillation has stopped for some reason. Furthermore, the operating clock for the LSI is switched from the external oscillator to the internal oscillation circuit. Once abnormal operation of the external oscillator has been detected, the state is retained until the next start from reset.

In this LSI, the clock from the internal oscillator is supplied to the bus masters, peripheral modules, and external bus on a reset start or cancellation of software standby. If the clock from the external oscillator has become stable enough for normal operation within the oscillation settling time, the external oscillator provides the clock. When the external oscillation stoppage detection function is disabled (the INOSCE bit in ROSCCR is 0), abnormal operation of the external oscillator will not be detected. Furthermore, the operating clock will not switch to the internal oscillation circuit.

22.5.2 Setting of External Oscillation Stoppage Detection Function

External oscillation stoppage detection function is enabled immediately after a reset start. This function is disabled by clearing the INOSCE bit in ROSCCR to 0 (the initial value is 1) after release from the reset state. The INOSCE bit in ROSCCR is set to 1 by a reset or in hardware standby mode*[1].

When the external oscillation stoppage detection function has been enabled and no edges are received from the clock source over a certain period (see table 25.12) due to an abnormality of the frequency of the external oscillator, this function judges that the external oscillator is operating abnormally and switches the operating clock to the clock from the internal oscillator (see table 22.4). At this time, the LSI sets the OSCERR bit*[2] in ROSCCR to 1 (the initial value is 0). Once the operating clock has been switched to the internal oscillator, operation continues with the internal oscillator even if the external oscillator returns to normal operation. To return to operation with the external oscillator, perform another reset start.

When the OSCIE bit in ROSCCR is set to 1 (the initial value is 0), an external oscillation stoppage detection interrupt (OSCERI)*[2] will be generated on detection of an abnormality in the external oscillation. When this interrupt is generated, processing by the LSI to handle the abnormality should proceed during the interrupt processing. Use a reset start to return from the external-oscillation stoppage-detection interrupt processing.

Notes: 1. This LSI does not have hardware standby mode.
2. Setting the ERRTEST bit in ROSCCR to 1 allows software setting of the OSCERR bit to 1 without requiring the detection of an abnormality in the external oscillation. Furthermore, setting the OSCIE bit in ROSCCR to 1 (initial value: 0) enables the external oscillation stoppage detection interrupt (OSCERI).

Table 22.4 Frequency of Internal Oscillation Circuit (Frequency of System Clock (Iϕ))

	Min.	Typ.	Max.
System clock (Iϕ) frequency (when multiplied by 8)	$8 \times f_{MAIN}\phi$ (min.)	$8 \times f_{MAIN}\phi$ (typ.)	$8 \times f_{MAIN}\phi$ (max.)
System clock (Iϕ) frequency (when multiplied by 4)	$4 \times f_{MAIN}\phi$ (min.)	$4 \times f_{MAIN}\phi$ (typ.)	$4 \times f_{MAIN}\phi$ (max.)
System clock (Iϕ) frequency (when multiplied by 2)	$2 \times f_{MAIN}\phi$ (min.)	$2 \times f_{MAIN}\phi$ (typ.)	$2 \times f_{MAIN}\phi$ (max.)
System clock (Iϕ) frequency (when multiplied by 1)	$f_{MAIN}\phi$ (min.)	$f_{MAIN}\phi$ (typ.)	$f_{MAIN}\phi$ (max.)

Note: For $f_{MAIN}\phi$, see section 25.6.3, Internal Oscillation Frequency.

22.5.3 Note when Using External Oscillation Stoppage Detection Function

When the internal oscillator is used as the operating clock because the external oscillator is not operating normally, the LSI must perform processing to handle the abnormality. The following operations are prohibited during this processing. Correct operation is not guaranteed if any of the following is performed.

- Clearing of the INOSCE bit
- Return from interrupt processing
- Transition to software standby mode

22.6 Usage Notes

22.6.1 Notes on Clock Pulse Generator

1. The frequency of φ (Iφ: system clock, Pφ: peripheral module clock, Bφ: external bus clock, Fφ: FLASH clock, Aφ: A/D clock, and Rφ: RSPI clock) supplied to each module changes according to the settings of SCKCR0 and SCKCR1. Set these registers so that the clock frequencies fall within the ranges shown in table 22.5.

Table 22.5 Frequency Range of Each Clock

		Frequency [MHz]								
		↓	8	16	20	32	40	64	80	↑
Clock name	Iφ	Setting prohibited	min	—	—	—	—	—	max	Setting prohibited
	Pφ		min	—	—	—	max			
	Bφ		min	—	max					
	Fφ		min	—	—	—	max			
	Aφ		min	—	—	—	max			
	Rφ		min	—	—	—	max			

Notes: 1. Do not set frequency higher than Iφ.
2. Set so as to satisfy Rφ ≥ Pφ ≥ Fφ.
3. When the RCAN is used, set so that Pφmin = 16 MHz.

2. All the on-chip peripheral modules (except for the DMAC, DTC, A/D, and RSPI) operate on the Pφ. Therefore, note that the time processing of modules such as a timer and SCI differs before and after changing the frequency.

   In addition, wait time for clearing software standby mode is modified by changing the frequency. For details, see section 23.7.3, Setting Oscillation Settling Time after Clearing Software Standby Mode.
3. The relationship between the system clock, peripheral module clock, external bus clock, A/D clock, and RSPI clock is Iφ ≥ Pφ, Iφ ≥ Bφ, Iφ ≥ Fφ, Iφ ≥ Aφ, and Iφ ≥ Rφ. In addition, the system clock setting has priority. Accordingly, Pφ, Bφ, Fφ, Aφ, and Rφ may have the frequency set by bits ICK2 to ICK0 regardless of the settings of bits PCK2 to PCK0, BCK2 to BCK0, FCK2 to FCK0, ACK2 to ACK0, and RCK2 to RCK0.

4. Figure 22.6 shows the clock modification timing. After a value is written to SCKCR0 or SCKCR1, this LSI waits for the current bus cycle to complete. After the current bus cycle completes, each clock frequency will be modified within one cycle (worst case) of the external clock.
5. When Iϕ > Pϕ, Iϕ> Fϕ, Iϕ > Aϕ, or Iϕ > Rϕ is specified by SCKCR0 and SCKCR1, signals from the peripheral modules must be synchronized with the system clock. When CPU instructions are used to clear the interrupt source flag of a peripheral module, the flag must be read after being cleared to 0.
6. When modifying the clock frequency of the system clock, peripheral module clock, external bus clock, A/D clock, and RSPI clock while Iϕ ≥ 40 MHz, execute NOP instruction eight times immediately after setting SCKCR0 and SCKCR1.

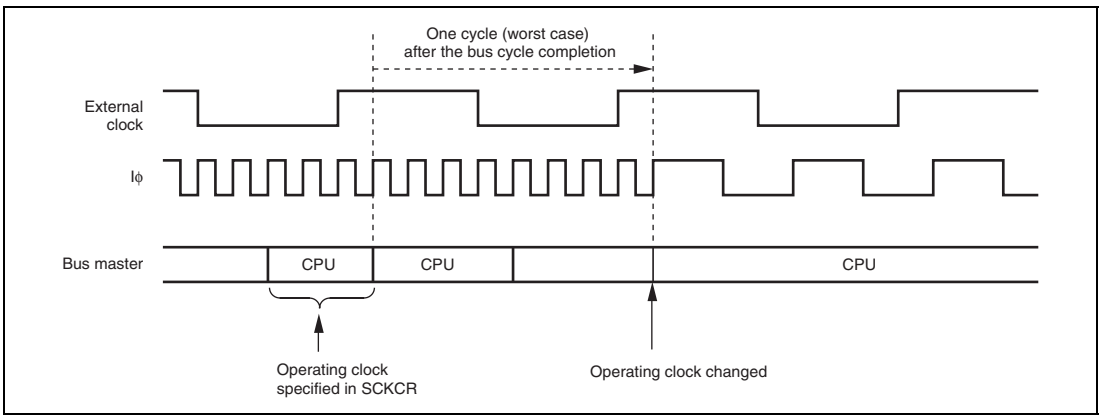


Figure 22.5 Clock Modification Timing

22.6.2 Notes on Resonator

Since various characteristics related to the resonator are closely linked to the user's board design, thorough evaluation is necessary on the user's part, using the resonator connection examples shown in this section as a reference. As the parameters for the resonator will depend on the floating capacitance of the resonator and the mounting circuit, the parameters should be determined in consultation with the resonator manufacturer. The design must ensure that a voltage exceeding the maximum rating is not applied to the resonator pin.

22.6.3 Notes on Board Design

When using the crystal resonator, place the crystal resonator and its load capacitors as close as possible to the XTAL and EXTAL pins. Other signal lines should be routed away from the oscillation circuit as shown in figure 22.6 to prevent induction from interfering with correct oscillation.

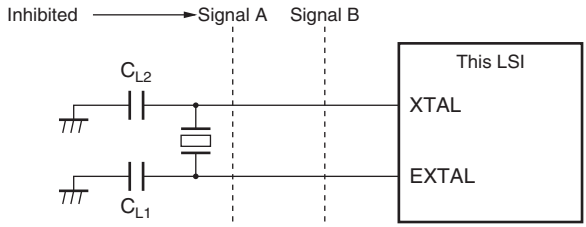
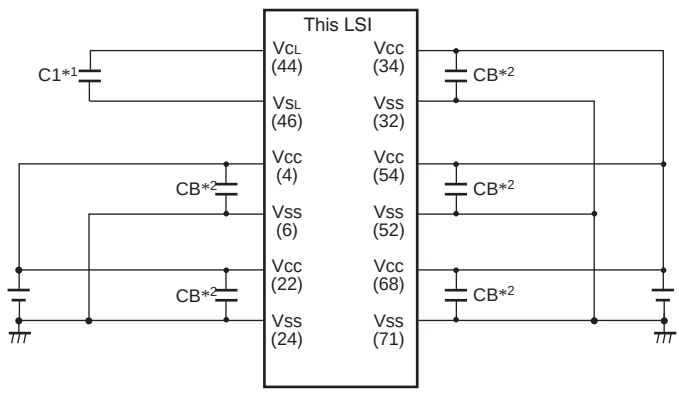


Figure 22.6 Note on Board Design for Oscillation Circuit

Figure 22.7 shows a connection example of bypass capacitor. Separate Vcc and Vss from the other Vcc and Vss lines at the board power supply source, and be sure to insert bypass capacitor (CB) close to the pins and its capacitance meets the characteristics of the user system board.



Notes: Numbers in parenthesis are pin numbers.
1. A 0.47- F capacitor should be used here.
2. CB is a laminated ceramic capacitor.

Figure 22.7 Connection Example of Bypass Capacitor

22.6.4 Notes on Input Clock Frequency

The frequency of the input clock is multiplied in the PLL circuit by a factor of 8. To reduce noises, a lower frequency ranging from 8 to 10 MHz is recommended.

Section 23 Power-Down Modes

This LSI has power consumption reduction functions, such as the multi-clock function, module stop function, and transition function to a power-down state.

23.1 Features

- Multi-clock function
  The frequency division ratio is settable independently for the system clock, peripheral module clock, external bus clock, FLASH clock, A/D clock, and RSPI clock.
- Module stop function
  The functions for each peripheral module can be stopped to make a transition to a power-down state.
- Transition function to power-down state
  Transition to a power-down state is possible to stop the CPU, peripheral modules, and oscillator.
- Three power-down modes
  Sleep mode
  All-module-clock-stop mode
  Software standby mode

Table 23.1 shows conditions for making a transition to a power-down state, states of the CPU and peripheral modules, and clearing method for each mode.

After a reset, since this LSI operates in normal program execution state, modules other than the DMAC and DTC are stopped.

Table 23.1 Operating States

Operating State	Sleep Mode	All-Module-Clock-Stop Mode	Software Standby Mode
Transition condition	Control register + instruction	Control register + instruction	Control register + instruction
Cancellation method	Interrupt	Interrupt*1	External interrupt
Oscillator	Functioning	Functioning	Halted
CPU	Halted (retained)	Halted (retained)	Halted (retained)
Watchdog timer	Functioning	Functioning	Halted (retained)
Peripheral modules	Functioning	Halted*2	Halted*2
I/O port	Functioning	Retained	Retained

Notes: "Halted (retained)" in the table means that the internal register values are retained and internal operations are suspended.

1. External interrupts and some internal interrupts (watchdog timer)
2. The RCAN enters the reset state, and other peripheral modules retain their states.

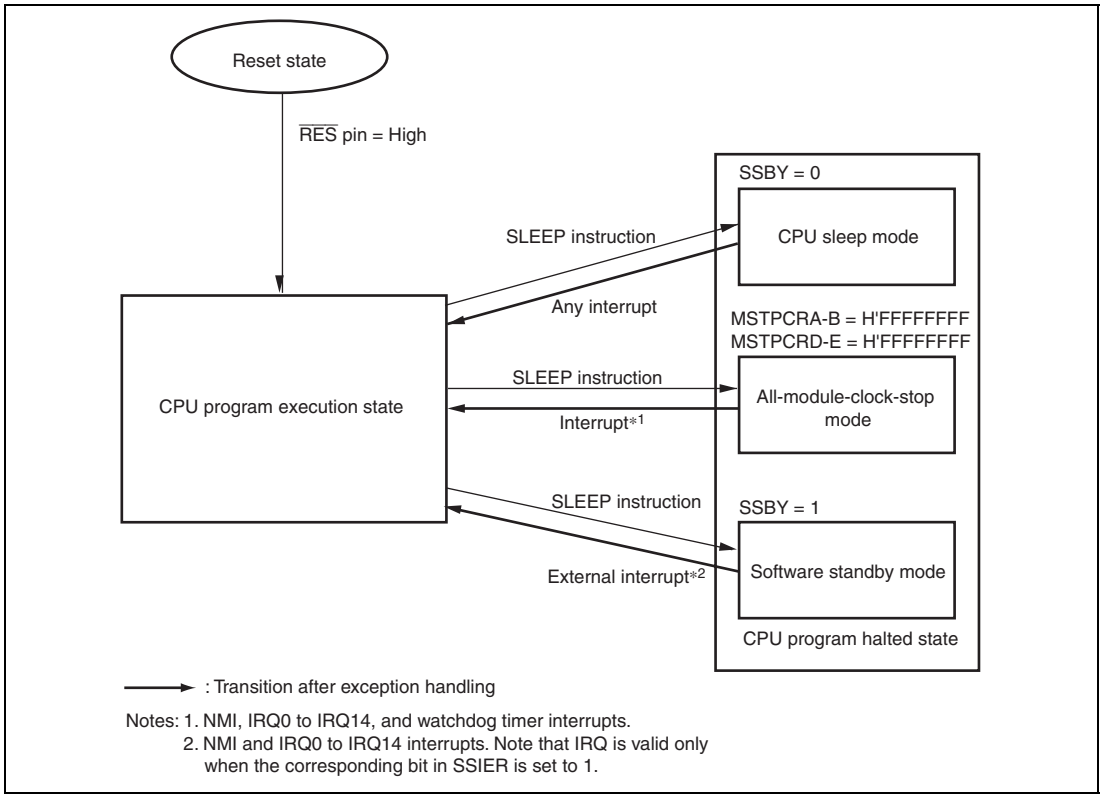


Figure 23.1 Mode Transitions

23.2 Register Descriptions

The registers related to the power-down modes are shown below. For details on the system clock control register 0 (SCKCR0) and the system clock control register 1 (SCKCR1), see section 22.1.1, System Clock Control Register 0 (SCKCR0) and section 22.1.2, System Clock Control Register 1 (SCKCR1).

- Standby control register (SBYCR)
- Module stop control register A (MSTPCRA)
- Module stop control register B (MSTPCRB)
- Module stop control register C (MSTPCRC)
- Module stop control register D (MSTPCRD)
- Module stop control register E (MSTPCRE)

23.2.1 Standby Control Register (SBYCR)

SBYCR controls software standby mode.

Bit	15	14	13	12	11	10	9	8
Bit Name	SSBY	OPE	SSBYF	STS4	STS3	STS2	STS1	STS0
Initial Value	0	1	0	0	1	1	1	1
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	—	—	—	—	—	—	—	—
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Bit Name	Initial Value	R/W	Description
15	SSBY	0	R/W	Software Standby Specifies the transition mode after executing the SLEEP instruction 0: Shifts to sleep mode after the SLEEP instruction is executed 1: Shifts to software standby mode after the SLEEP instruction is executed This bit does not change when clearing the software standby mode by using external interrupts and shifting to normal operation. For clearing, write 0 to this bit. When the WDT is used as the watchdog timer, the setting of this bit is disabled. In this case, a transition is always made to sleep mode or all-module-clock-stop mode after the SLEEP instruction is executed.
14	OPE	1	R/W	Output Port Enable Specifies whether the output of the address bus and bus control signals ($\overline{CS0}$ to $\overline{CS7}$, $\overline{AS}$, $\overline{RD}$, $\overline{HWR}$, $\overline{LWR}$, $\overline{UCAS}$, and $\overline{LCAS}$) is retained or set to the high-impedance state in software standby mode. 0: In software standby mode, address bus and bus control signals are high-impedance 1: In software standby mode, address bus and bus control signals retain output state
13	SSBYF	0	R	Software Standby Flag This flag is set to 1 when a transition is made to software standby mode. This flag is cleared to 0 by a reset by the $\overline{RES}$ pin. If software standby mode is cleared by the input of the rising edge on the NMI pin, this flag remains set to 1 and is not cleared to 0. 0: Software standby mode is not entered, or this flag has been cleared by a rest caused by the $\overline{RES}$ pin after software standby mode is entered. 1: The reset state is not entered by the $\overline{RES}$ pin after recovery from software standby mode.

Bit	Bit Name	Initial Value	R/W	Description
12	STS4	0	R/W	Standby Timer Select 4 to 0
11	STS3	1	R/W	These bits select the time the MCU waits for the clock to settle when software standby mode is cleared by an external interrupt. With a crystal resonator, see table 23.2 and make a selection according to the operating frequency so that the standby time is at least equal to the oscillation settling time. With an external clock, a PLL circuit settling time is necessary. See table 23.2 to set the standby time. While oscillation is being settled, the timer is counted on the Pϕ clock frequency. Careful consideration is required in multi-clock mode. 00000: Reserved 00001: Reserved 00010: Reserved 00011: Reserved 00100: Reserved 00101: Standby time = 64 states 00110: Standby time = 512 states 00111: Standby time = 1024 states 01000: Standby time = 2048 states 01001: Standby time = 4096 states 01010: Standby time = 16384 states 01011: Standby time = 32768 states 01100: Standby time = 65536 states 01101: Standby time = 131072 states 01110: Standby time = 262144 states 01111: Standby time = 524288 states 10000: Reserved 10001: Reserved 1001*: Reserved 101**: Reserved 11***: Reserved
10	STS2	1	R/W	
9	STS1	1	R/W	
8	STS0	1	R/W	
7 to 0	—	0	R/W	Reserved These bits are always read as 0. The write value should always be 0.

Note: * Don't care

23.2.2 Module Stop Control Registers A, B, C, D, and E (MSTPCRA, MSTPCRB, MSTPCRC, MSTPCRD, and MSTPCRE)

MSTPCRA, MSTPCRB, MSTPCRC, MSTPCRD, and MSTPCRE control module stop mode. Setting a bit to 1 makes the corresponding module enter module stop mode, while clearing the bit to 0 clears module stop mode.

- MSTPCRA

Bit	15	14	13	12	11	10	9	8
Bit Name	ACSE	MSTPA14	MSTPA13	MSTPA12	MSTPA11	MSTPA10	MSTPA9	MSTPA8
Initial Value	0	0	0	0	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	MSTPA7	MSTPA6	MSTPA5	MSTPA4	MSTPA3	MSTPA2	MSTPA1	MSTPA0
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- MSTPCRB

Bit	15	14	13	12	11	10	9	8
Bit Name	MSTPB15	MSTPB14	MSTPB13	MSTPB12	MSTPB11	MSTPB10	MSTPB9	MSTPB8
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	MSTPB7	MSTPB6	MSTPB5	MSTPB4	MSTPB3	MSTPB2	MSTPB1	MSTPB0
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- MSTPCRC

Bit	15	14	13	12	11	10	9	8
Bit Name	MSTPC15	MSTPC14	MSTPC13	MSTPC12	MSTPC11	MSTPC10	MSTPC9	MSTPC8
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	MSTPC7	MSTPC6	MSTPC5	MSTPC4	MSTPC3	MSTPC2	MSTPC1	MSTPC0
Initial Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- MSTPCRD

Bit	15	14	13	12	11	10	9	8
Bit Name	MSTPD15	MSTPD14	MSTPD13	MSTPD12	MSTPD11	MSTPD10	MSTPD9	MSTPD8
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	MSTPD7	MSTPD6	MSTPD5	MSTPD4	MSTPD3	MSTPD2	MSTPD1	MSTPD0
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- MSTPCRE

Bit	15	14	13	12	11	10	9	8
Bit Name	MSTPE15	MSTPE14	MSTPE13	MSTPE12	MSTPE11	MSTPE10	MSTPE9	MSTPE8
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Bit Name	MSTPE7	MSTPE6	MSTPE5	MSTPE4	MSTPE3	MSTPE2	MSTPE1	MSTPE0
Initial Value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- MSTPCRA

Bit	Bit Name	Initial Value	R/W	Module
15	ACSE	0	R/W	All-Module-Clock-Stop Mode Enable Enables/disables all-module-clock-stop mode for reducing current consumption by stopping the bus controller and I/O ports operations when the CPU executes the SLEEP instruction after module stop mode has been set for all of the on-chip peripheral modules controlled by MSTPCR. 0: All-module-clock-stop mode disabled 1: All-module-clock-stop mode enabled
14	MSTPA14	0	R/W	Reserved
13	MSTPA13	0	R/W	DMA controller (DMAC)
12	MSTPA12	0	R/W	Data transfer controller (DTC)
11 to 5	MSTPA11 to MSTPA5	All 1	R/W	Reserved These bits are always read as 1. The write value should always be 1.
4	MSTPA4	1	R/W	A/D converter (Unit 1)
3	MSTPA3	1	R/W	A/D converter (Unit 0)
2	MSTPA2	1	R/W	Reserved This bit is always read as 1. The write value should always be 1.
1	MSTPA1	1	R/W	16-bit timer pulse unit (TPU channels 11 to 6)
0	MSTPA0	1	R/W	16-bit timer pulse unit (TPU channels 5 to 0)

- MSTPCRB

Bit	Bit Name	Initial Value	R/W	Module
15	MSTPB15	1	R/W	Programmable pulse generator (PPG)
14, 13	MSTPB14 MSTPB13	All 1	R/W	Reserved These bits are always read as 1. The write value should always be 1.
12	MSTPB12	1	R/W	Serial communication interface_4 (SCI_4)
11	MSTPB11	1	R/W	Serial communication interface_3 (SCI_3)
10 to 0	MSTPB10 to MSTPB0	All 1	R/W	Reserved These bits are always read as 1. The write value should always be 1.

- MSTPCRC

Bit	Bit Name	Initial Value	R/W	Module
15 to 8	MSTPC15 to MSTPC8	All 1	R/W	Reserved These bits are always read as 1. The write value should always be 1.
7	MSTPC7	0	R/W	Reserved
6	MSTPC6	0	R/W	These bits are always read as 0. The write value should always be 0.
5	MSTPC5	0	R/W	On-chip RAM_4 (H'FFF2000 to H'FFF3FFF)
4	MSTPC4	0	R/W	Always set the same value in MSTPC5 and MSTPC4. While these bits are set, the value read from the RAM is not guaranteed.
3	MSTPC3	0	R/W	On-chip RAM_3, 2 (H'FFF4000 to H'FFF7FFF)
2	MSTPC2	0	R/W	Always set the same value in MSTPC3 and MSTPC2. While these bits are set, the value read from the RAM is not guaranteed.
1	MSTPC1	0	R/W	On-chip RAM_1, 0 (H'FFF8000 to H'FFFBFFF)
0	MSTPC0	0	R/W	Always set the same value in MSTPC1 and MSTPC0. While these bits are set, the value read from the RAM is not guaranteed.

- MSTPCRD

Bit	Bit Name	Initial Value	R/W	Module
15 to 2	MSTPD15 to MSTPD2	All 1	R/W	Reserved These bits are always read as 1. The write value should always be 1.
1	MSTPD1	1	R/W	Error correction (CRC)
0	MSTPD0	1	R/W	Reserved This bit is always read as 1. The write value should always be 1.

- MSTPCRE

Bit	Bit Name	Initial Value	R/W	Module
15	MSTPE15	1	R/W	Control area network_1 (RCAN-TL1_1)
14	MSTPE14	1	R/W	Control area network_0 (RCAN-TL1_0)
13	MSTPE13	1	R/W	Hardware local interconnect network (HW-LIN)
12	MSTPE12	1	R/W	Reserved This bit is always read as 1. The write value should always be 1.
11	MSTPE11	1	R/W	Renesas serial peripheral interface_D (RSPI_D)
10	MSTPE10	1	R/W	Renesas serial peripheral interface_C (RSPI_C)
9	MSTPE9	1	R/W	Renesas serial peripheral interface_B (RSPI_B)
8	MSTPE8	1	R/W	Renesas serial peripheral interface_A (RSPI_A)
7 to 0	MSTPE7 to MSTP0	All 1	R/W	Reserved These bits are always read as 1. The write value should always be 1.

23.3 Multi-Clock Function

When bits ICK2 to ICK0, PCK2 to PCK0, and BCK2 to BCK0 in SCKCR0 and FCK2 to FCK0, ACK2 to ACK0, and RCK2 to RCK0 in SCKCR1 are set, a transition is made to the multi-clock function at the end of the bus cycle. In the multi-clock function, the CPU and bus masters operate on the operating clock specified by bits ICK2 to ICK0. The peripheral modules operate on the operating clock specified by bits PCK2 to PCK0, FCK2 to FCK0, ACK2 to ACK0, and RCK2 to RCK0. The external clock operates on the operating clock specified by bits BCK2 to BCK0.

Even if the frequencies specified by bits PCK2 to PCK0, BCK2 to BCK0, FCK2 to FCK0, ACK2 to ACK0, and RCK2 to RCK0 are higher than the frequency specified by bits ICK2 to ICK0, the specified values are not reflected in the peripheral module and external bus clocks. The peripheral module and external bus clocks are restricted to the operating clock specified by bits ICK2 to ICK0.

The multi-clock function is cleared by clearing all of bits ICK2 to ICK0, PCK2 to PCK0, BCK2 to BCK0, FCK2 to FCK0, ACK2 to ACK0, and RCK2 to RCK0 to 0. A transition is made to the normal state at the end of the bus cycle, and the multi-clock function is cleared.

If a SLEEP instruction is executed while the SSBY bit in SBYCR is cleared to 0, this LSI enters sleep mode. The multi-clock function is enabled even in sleep mode.

If a SLEEP instruction is executed while the SSBY bit in SBYCR is set to 1, this LSI enters software standby mode. The multi-clock function stops in software standby mode. When software standby mode is cleared by an external interrupt, the multi-clock function is restored.

When the $\overline{\text{RES}}$ pin is driven low, the reset state is entered and the multi-clock function is cleared. The same applies to a reset caused by a watchdog timer overflow.

23.4 Module Stop Function

The module stop function can be set for individual on-chip peripheral modules.

When the corresponding MSTP bit in MSTPCRA, MSTPCRB, MSTPCRC, MSTPCRD, or MSTPCRE is set to 1, module operation stops at the end of the bus cycle and a transition is made to module stop mode. The CPU continues operation independently.

When the corresponding MSTP bit is cleared to 0, module stop mode is cleared and the module starts operation at the end of the bus cycle. In module stop mode, the internal states of modules are retained.

After the reset state is cleared, all modules other than the DMAC and DTC are in module stop mode.

The registers of the module for which module stop mode is selected cannot be read from or written to.

23.5 Sleep Mode

23.5.1 Transition to Sleep Mode

When the SLEEP instruction is executed when the SSBY bit in SBYCR is 0, the CPU enters sleep mode. In sleep mode, CPU operation stops but the contents of the CPU's internal registers are retained. Other peripheral functions do not stop.

23.5.2 Clearing Sleep Mode

Sleep mode is exited by any interrupt, signals on the $\overline{\text{RES}}$ or $\overline{\text{STBY}}$ pin, and a reset caused by a watchdog timer overflow.

1. Clearing by interrupt
   When an interrupt occurs, sleep mode is exited and interrupt exception processing starts. Sleep mode is not exited if the interrupt is disabled, or interrupts other than NMI are masked by the CPU.
2. Clearing by $\overline{\text{RES}}$ pin
   Driving the $\overline{\text{RES}}$ pin low selects the reset state. After the stipulated reset input duration, driving the $\overline{\text{RES}}$ pin high makes the CPU start the reset exception processing.
3. Clearing by reset caused by watchdog timer overflow
   Sleep mode is exited by an internal reset caused by a watchdog timer overflow.

23.6 All-Module-Clock-Stop Mode

When the ACSE bit is set to 1 and all modules controlled by MSTPCR are stopped (MSTPCRA, MSTPCRB, MSTPCRD, and MSTPCRE = H'FFFFFFFFFFFFFFFF), executing a SLEEP instruction with the SSBY bit in SBYCR cleared to 0 will cause all modules (except watchdog timer), the bus controller, and the I/O ports to stop operating, and to make a transition to all-module-clock-stop mode at the end of the bus cycle.

All-module-clock-stop mode is cleared by an external interrupt (NMI or $\overline{\text{IRQ0}}$ to $\overline{\text{IRQ14}}$ pins), $\overline{\text{RES}}$ pin input, or an internal interrupt (watchdog timer), and the CPU returns to the normal program execution state via the exception handling state. All-module-clock-stop mode is not cleared if interrupts are disabled, interrupts other than NMI are masked on the CPU side, or the interrupt is specified for DTC activation.

23.7 Software Standby Mode

23.7.1 Transition to Software Standby Mode

If a SLEEP instruction is executed when the SSBY bit in SBYCR is set to 1, software standby mode is entered. In this mode, the CPU, on-chip peripheral modules, and oscillator all stop. However, the contents of the CPU's internal registers, on-chip RAM data, the states of on-chip peripheral functions other than the SCI, and the states of the I/O ports are retained. Whether the address bus and bus control signals are placed in the high-impedance state or retain the output state can be specified by the OPE bit in SBYCR. In this mode the oscillator stops, allowing power consumption to be significantly reduced.

If the WDT is used as a watchdog timer, it is impossible to make a transition to software standby mode. The WDT should be stopped before the SLEEP instruction execution.

23.7.2 Clearing Software Standby Mode

Software standby mode is cleared by an external interrupt (NMI pin, or pins $\overline{\text{IRQ0}}$ to $\overline{\text{IRQ14}}$*), and $\overline{\text{RES}}$ pin.

1. Clearing by interrupt

   When an NMI or IRQ0 to IRQ14* interrupt request signal is input, clock oscillation starts, and after the elapse of the time set in bits STS4 to STS0 in SBYCR, stable clocks are supplied to the entire LSI, software standby mode is cleared, and interrupt exception handling is started.

   When clearing software standby mode with an IRQ0 to IRQ14* interrupt, set the corresponding enable bit to 1 and ensure that no interrupt with a higher priority than interrupts IRQ0 to IRQ14* is generated. Software standby mode cannot be cleared if the interrupt has been masked on the CPU side or has been designated as a DTC activation source.

   Note that the vector address for the WAT interrupt exception handling differs between the cases when the WAT interrupt cancels the software standby mode and when the WAT interrupt occurs during normal operation.

   Note: * By setting the SSIn bit in SSIER to 1, $\overline{\text{IRQ0}}$ to $\overline{\text{IRQ14}}$ can be used as a software standby mode clearing source.

2. Clearing by $\overline{\text{RES}}$ pin

   When the $\overline{\text{RES}}$ pin is driven low, clock oscillation is started. At the same time as clock oscillation starts, clocks are supplied to the entire LSI. Note that the $\overline{\text{RES}}$ pin must be held low until clock oscillation settles. When the $\overline{\text{RES}}$ pin goes high, the CPU begins reset exception handling.

23.7.3 Setting Oscillation Settling Time after Clearing Software Standby Mode

Bits STS4 to STS0 in SBYCR should be set as described below.

1. Using a crystal resonator
   Set bits STS4 to STS0 so that the standby time is at least equal to the oscillation settling time.
   Table 23.2 shows the standby times for operating frequencies and settings of bits STS4 to STS0.
2. Using an external clock
   Oscillation settling time is necessary as with the crystal resonator. See table 23.2 to set the standby time.

Table 23.2 Oscillation Settling Time Settings

						Pϕ* [MHz]							
STS4	STS3	STS2	STS1	STS0	Standby Time	40	33	25	20	13	10	8	Unit
0	0	0	0	0	Reserved	—	—	—	—	—	—	—	µs
				1	Reserved	—	—	—	—	—	—	—	
			1	0	Reserved	—	—	—	—	—	—	—	
				1	Reserved	—	—	—	—	—	—	—	
		1	0	0	Reserved	—	—	—	—	—	—	—	
				1	64	1.6	1.9	2.6	3.2	4.9	6.4	8.0	
			1	0	512	12.8	15.5	20.5	25.6	39.4	51.2	64.0	
				1	1024	25.6	31.0	41.0	51.2	78.8	102.4	128.0	
	1	0	0	0	2048	51.2	62.1	81.9	102.4	157.5	204.8	256.0	
				1	4096	0.10	0.12	0.16	0.20	0.32	0.41	0.51	ms
			1	0	16384	0.41	0.50	0.66	0.82	1.26	1.64	2.05	
				1	32768	0.82	0.99	1.31	1.64	2.52	3.28	4.10	
		1	0	0	65536	1.64	1.99	2.62	3.28	5.04	6.55	8.19	
				1	131072	3.28	3.97	5.24	6.55	10.08	13.11	16.38	
			1	0	262144	6.55	7.94	10.49	13.11	20.16	26.21	32.77	
				1	524288	13.11	15.89	20.97	26.21	40.33	52.43	65.54	
1	0	0	0	0	1048576	26.21	31.78	41.94	52.43	80.66	104.86	131.07	

: Recommended time setting when using the external clock.

: Recommended time setting when using the crystal resonator.

Note: * Pϕ is the output from the peripheral module frequency divider.

23.7.4 Software Standby Mode Application Example

Figure 23.2 shows an example in which a transition is made to software standby mode at the falling edge on the NMI pin, and software standby mode is cleared at the rising edge on the NMI pin.

In this example, an NMI interrupt is accepted with the NMIEG bit in INTCR cleared to 0 (falling edge specification), then the NMIEG bit is set to 1 (rising edge specification), the SSBY bit is set to 1, and a SLEEP instruction is executed, causing a transition to software standby mode.

Software standby mode is then cleared at the rising edge on the NMI pin.

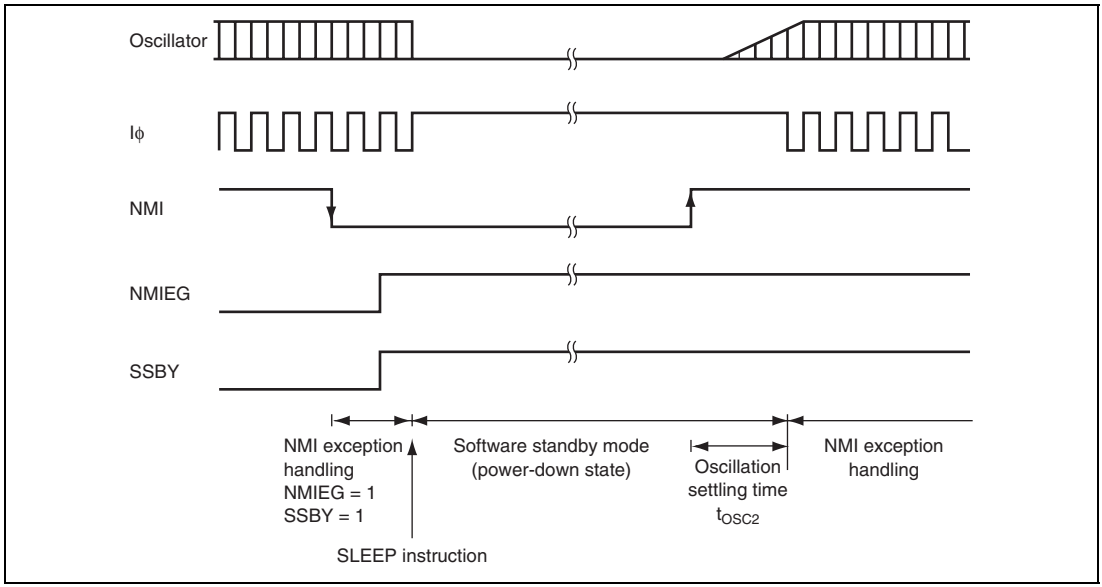


Figure 23.2 Software Standby Mode Application Example

23.8 φ Clock Output Control

Output of the φ clock (Bφ) can be controlled by bits PSTOP1 and POSEL1 in SCKCR0, and DDR for the corresponding PA7 pin. Clearing both bits PSTOP1 and POSEL1 to 0 enables the Bφ clock output on the PA7 pin. When bit PSTOP1 is set to 1, the Bφ clock output stops at the end of the bus cycle, and the Bφ clock output goes high. When DDR for the PA7 pin is cleared to 0, the Bφ clock output is disabled and the pin becomes an input port.

Table 23.3 shows the states of the φ pin in each processing state.

Table 23.3 φ Pin (PA7) State in Each Processing State

Register Setting Value			Normal Operating State	Sleep Mode	All-Module-Clock-Stop Mode	Software Standby Mode	
DDR	PSTOP1	POSEL1				OPE = 0	OPE = 1
0	×	×	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z
1	0	0	Bφ output	Bφ output	Bφ output	High	High
1	0	1	Setting prohibited	Setting prohibited	Setting prohibited	Setting prohibited	Setting prohibited
1	1	×	High	High	High	Hi-Z	High

23.9 Usage Notes

23.9.1 I/O Port Status

In software standby mode, the I/O port states are retained. Therefore, there is no reduction in current consumption for the output current when a high-level signal is output.

23.9.2 Current Consumption during Oscillation Settling Standby Period

Current consumption increases during the oscillation settling standby period.

23.9.3 DMAC and DTC Module Stop

Depending on the operating state of the DMAC and DTC, bits MSTPA13 and MSTPA12 may not be set to 1. Setting of the DMAC or DTC module stop mode should be performed only when each module is not activated.

For details, see section 7, DMA Controller (DMAC), and section 8, Data Transfer Controller (DTC).

23.9.4 On-Chip Peripheral Module Interrupts

Relevant interrupt operations cannot be performed in module stop mode. Consequently, if module stop mode is entered when an interrupt has been requested, it will not be possible to clear the CPU interrupt source or the activation sources of DMAC and DTC. Interrupts should therefore be disabled before entering module stop mode.

23.9.5 Writing to MSTPCRA, MSTPCRB, MSTPCRC, MSTPCRD, and MSTPCRE

MSTPCRA, MSTPCRB, MSTPCRC, MSTPCRD, and MSTPCRE should only be written to by the CPU.

Section 24 List of Registers

The register list gives information on the on-chip I/O register addresses, the bit configurations of the registers, and the register states in each operating mode. The information is given as shown below.

1. Register Addresses (address order)

- Registers are listed from the lower allocation addresses.
- Registers are classified according to functional modules.
- The number of Access Cycles indicates the number of states based on the specified reference clock.
- Undefined and reserved addresses cannot be accessed. Do not access these addresses; otherwise, the operation when accessing these bits and subsequent operations cannot be guaranteed.

2. Register Bits

- Bit configurations of the registers are listed in the same order as the Register Addresses.
- Reserved bits are indicated by "—" in the bit name column.
- Space in the bit name field indicates that the entire register is allocated to either the counter or data.
- For the registers of 16 or 32 bits, the MSB is listed first. Byte configuration description order is subject to big endian.

3. Register States in Each Operating Mode

- Register states are listed in the same order as the Register Addresses.
- For the initialized state of each bit, refer to the register description in the corresponding section.
- The register states described here are for the basic operating modes. If there is a specific reset for an on-chip peripheral module, refer to the section on that on-chip peripheral module.

24.1 Register Addresses (Address Order)

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
Master control register_0	MCR_0	16	H'FFC000	RCAN_0	16	2Pϕ/3Pϕ
General status register_0	GSR_0	16	H'FFC002	RCAN_0	16	2Pϕ/3Pϕ
Bit configuration register 1_0	BCR1_0	16	H'FFC004	RCAN_0	16	2Pϕ/3Pϕ
Bit configuration register 0_0	BCR0_0	16	H'FFC006	RCAN_0	16	2Pϕ/3Pϕ
Interrupt request register_0	IRR_0	16	H'FFC008	RCAN_0	16	2Pϕ/3Pϕ
Interrupt mask register_0	IMR_0	16	H'FFC00A	RCAN_0	16	2Pϕ/3Pϕ
Transmit error counter_0	TEC_0	16	H'FFC00C	RCAN_0	16	2Pϕ/3Pϕ
Receive error counter_0	REC_0					
Transmission pending register 1_0	TXPR1_0	16	H'FFC020	RCAN_0	32	4Pϕ/6Pϕ
Transmission pending register 0_0	TXPR0_0	16				
Transmit cancel register 1_0	TXCR1_0	16	H'FFC028	RCAN_0	16	2Pϕ/3Pϕ
Transmit cancel register 0_0	TXCR0_0	16	H'FFC02A	RCAN_0	16	2Pϕ/3Pϕ
Transmit acknowledge register 1_0	TXACK1_0	16	H'FFC030	RCAN_0	16	2Pϕ/3Pϕ
Transmit acknowledge register 0_0	TXACK0_0	16	H'FFC032	RCAN_0	16	2Pϕ/3Pϕ
Abort acknowledgement register 1_0	ABACK1_0	16	H'FFC038	RCAN_0	16	2Pϕ/3Pϕ
Abort acknowledgement register 0_0	ABACK0_0	16	H'FFC03A	RCAN_0	16	2Pϕ/3Pϕ
Data frame receive pending register 1_0	RXPR1_0	16	H'FFC040	RCAN_0	16	2Pϕ/3Pϕ
Data frame receive pending register 0_0	RXPR0_0	16	H'FFC042	RCAN_0	16	2Pϕ/3Pϕ
Remote frame receive pending register 1_0	RFPR1_0	16	H'FFC048	RCAN_0	16	2Pϕ/3Pϕ
Remote frame receive pending register 0_0	RFPR0_0	16	H'FFC04A	RCAN_0	16	2Pϕ/3Pϕ
Mailbox Interrupt mask register1_0	MBIMR1_0	16	H'FFC050	RCAN_0	16	2Pϕ/3Pϕ
Mailbox Interrupt mask register0_0	MBIMR0_0	16	H'FFC052	RCAN_0	16	2Pϕ/3Pϕ
Unread message status register 1_0	UMSR1_0	16	H'FFC058	RCAN_0	16	2Pϕ/3Pϕ
Unread message status register 0_0	UMSR0_0	16	H'FFC05A	RCAN_0	16	2Pϕ/3Pϕ
Time trigger control register 0_0	TTCR0_0	16	H'FFC080	RCAN_0	16	2Pϕ/3Pϕ
Cycle maximum/Tx-Enable Window register_0	CMAX_TEW_0	16	H'FFC084	RCAN_0	16	2Pϕ/3Pϕ
Reference trigger offset register_0	RFTROFF_0	16	H'FFC086	RCAN_0	16	2Pϕ/3Pϕ
Timer status register_0	TSR_0	16	H'FFC088	RCAN_0	16	2Pϕ/3Pϕ
Cycle counter register_0	CCR_0	16	H'FFC08A	RCAN_0	16	2Pϕ/3Pϕ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
Timer counter register_0		TCNTR_0	16	H'FFC08C	RCAN_0	16	2Pϕ/3Pϕ
Cycle time register_0		CYCTR_0	16	H'FFC090	RCAN_0	16	2Pϕ/3Pϕ
Reference mark register_0		RFMK_0	16	H'FFC094	RCAN_0	16	2Pϕ/3Pϕ
Timer compare match register 0_0		TCMR0_0	16	H'FFC098	RCAN_0	16	2Pϕ/3Pϕ
Timer compare match register 1_0		TCMR1_0	16	H'FFC09C	RCAN_0	16	2Pϕ/3Pϕ
Timer compare match register 2_0		TCMR2_0	16	H'FFC0A0	RCAN_0	16	2Pϕ/3Pϕ
Tx-trigger time selection register_0		TTTSEL_0	16	H'FFC0A4	RCAN_0	16	2Pϕ/3Pϕ
MB_0[0].	CONTROL0H	—	16	H'FFC100	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFC102	RCAN_0	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFC104	RCAN_0	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFC106	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFC108	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFC109	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFC10A	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFC10B	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFC10C	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFC10D	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFC10E	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFC10F	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFC110	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFC111	RCAN_0	16	2+nPϕ/3+nPϕ
	TIMESTAMP	—	16	H'FFC112	RCAN_0	16	2Pϕ/3Pϕ
MB_0[1].	CONTROL0H	—	16	H'FFC120	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFC122	RCAN_0	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFC124	RCAN_0	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFC126	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFC128	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFC129	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFC12A	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFC12B	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFC12C	RCAN_0	16	2+nPϕ/3+nPϕ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[1].	MSG_DATA[5]	—	8	H'FFC12D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC12E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC12F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC130	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC131	RCAN_0	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFC132	RCAN_0	16	2Pφ/3Pφ
MB_0[2].	CONTROL0H	—	16	H'FFC140	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC142	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC144	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC146	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC148	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC149	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC14A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC14B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC14C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC14D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC14E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC14F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC150	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC151	RCAN_0	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFC152	RCAN_0	16	2Pφ/3Pφ
MB_0[3].	CONTROL0H	—	16	H'FFC160	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC162	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC164	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC166	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC168	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC169	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC16A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC16B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC16C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC16D	RCAN_0	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[3].	MSG_DATA[6]	—	8	H'FFC16E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC16F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC170	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC171	RCAN_0	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFC172	RCAN_0	16	2Pφ/3Pφ
MB_0[4].	CONTROL0H	—	16	H'FFC180	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC182	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC184	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC186	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC188	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC189	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC18A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC18B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC18C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC18D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC18E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC18F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC190	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC191	RCAN_0	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFC192	RCAN_0	16	2Pφ/3Pφ
MB_0[5].	CONTROL0H	—	16	H'FFC1A0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC1A2	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC1A4	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC1A6	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC1A8	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC1A9	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC1AA	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC1AB	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC1AC	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC1AD	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC1AE	RCAN_0	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[5].	MSG_DATA[7]	—	8	H'FFC1AF	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC1B0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC1B1	RCAN_0	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFC1B2	RCAN_0	16	2Pφ/3Pφ
MB_0[6].	CONTROL0H	—	16	H'FFC1C0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC1C2	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC1C4	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC1C6	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC1C8	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC1C9	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC1CA	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC1CB	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC1CC	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC1CD	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC1CE	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC1CF	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC1D0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC1D1	RCAN_0	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFC1D2	RCAN_0	16	2Pφ/3Pφ
MB_0[7].	CONTROL0H	—	16	H'FFC1E0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC1E2	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC1E4	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC1E6	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC1E8	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC1E9	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC1EA	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC1EB	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC1EC	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC1ED	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC1EE	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC1EF	RCAN_0	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[7].	CONTROL1H	—	8	H'FFC1F0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC1F1	RCAN_0	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFC1F2	RCAN_0	16	2Pφ/3Pφ
MB_0[8].	CONTROL0H	—	16	H'FFC200	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC202	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC204	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC206	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC208	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC209	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC20A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC20B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC20C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC20D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC20E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC20F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC210	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC211	RCAN_0	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFC212	RCAN_0	16	2Pφ/3Pφ
MB_0[9].	CONTROL0H	—	16	H'FFC220	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC222	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC224	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC226	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC228	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC229	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC22A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC22B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC22C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC22D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC22E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC22F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC230	RCAN_0	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[9].	CONTROL1L	—	8	H'FFC231	RCAN_0	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFC232	RCAN_0	16	2Pφ/3Pφ
MB_0[10].	CONTROL0H	—	16	H'FFC240	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC242	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC244	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC246	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC248	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC249	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC24A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC24B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC24C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC24D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC24E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC24F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC250	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC251	RCAN_0	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFC252	RCAN_0	16	2Pφ/3Pφ
MB_0[11].	CONTROL0H	—	16	H'FFC260	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC262	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC264	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC266	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC268	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC269	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC26A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC26B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC26C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC26D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC26E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC26F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC270	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC271	RCAN_0	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFC272	RCAN_0	16	2Pφ/3Pφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[12].	CONTROL0H	—	16	H'FFC280	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC282	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC284	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC286	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC288	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC289	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC28A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC28B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC28C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC28D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC28E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC28F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC290	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC291	RCAN_0	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFC292	RCAN_0	16	2Pφ/3Pφ
MB_0[13].	CONTROL0H	—	16	H'FFC2A0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC2A2	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC2A4	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC2A6	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC2A8	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC2A9	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC2AA	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC2AB	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC2AC	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC2AD	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC2AE	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC2AF	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC2B0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC2B1	RCAN_0	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFC2B2	RCAN_0	16	2Pφ/3Pφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[14].	CONTROL0H	—	16	H'FFC2C0	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFC2C2	RCAN_0	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFC2C4	RCAN_0	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFC2C6	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFC2C8	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFC2C9	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFC2CA	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFC2CB	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFC2CC	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFC2CD	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFC2CE	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFC2CF	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFC2D0	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFC2D1	RCAN_0	16	2+nPϕ/3+nPϕ
	TIMESTAMP	—	16	H'FFC2D2	RCAN_0	16	2Pϕ/3Pϕ
MB_0[15].	CONTROL0H	—	16	H'FFC2E0	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFC2E2	RCAN_0	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFC2E4	RCAN_0	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFC2E6	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFC2E8	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFC2E9	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFC2EA	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFC2EB	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFC2EC	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFC2ED	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFC2EE	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFC2EF	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFC2F0	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFC2F1	RCAN_0	16	2+nPϕ/3+nPϕ
	TIMESTAMP	—	16	H'FFC2F2	RCAN_0	16	2Pϕ/3Pϕ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[16].	CONTROL0H	—	16	H'FFC300	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC302	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC304	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC306	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC308	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC309	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC30A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC30B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC30C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC30D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC30E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC30F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC310	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC311	RCAN_0	16	2+nPφ/3+nPφ
MB_0[17].	CONTROL0H	—	16	H'FFC320	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC322	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC324	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC326	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC328	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC329	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC32A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC32B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC32C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC32D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC32E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC32F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC330	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC331	RCAN_0	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[18].	CONTROL0H	—	16	H'FFC340	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC342	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC344	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC346	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC348	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC349	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC34A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC34B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC34C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC34D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC34E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC34F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC350	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC351	RCAN_0	16	2+nPφ/3+nPφ
MB_0[19].	CONTROL0H	—	16	H'FFC360	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC362	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC364	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC366	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC368	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC369	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC36A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC36B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC36C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC36D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC36E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC36F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC370	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC371	RCAN_0	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[20].	CONTROL0H	—	16	H'FFC380	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC382	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC384	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC386	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC388	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC389	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC38A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC38B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC38C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC38D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC38E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC38F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC390	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC391	RCAN_0	16	2+nPφ/3+nPφ
MB_0[21].	CONTROL0H	—	16	H'FFC3A0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC3A2	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC3A4	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC3A6	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC3A8	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC3A9	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC3AA	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC3AB	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC3AC	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC3AD	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC3AE	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC3AF	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC3B0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC3B1	RCAN_0	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[22].	CONTROL0H	—	16	H'FFC3C0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC3C2	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC3C4	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC3C6	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC3C8	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC3C9	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC3CA	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC3CB	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC3CC	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC3CD	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC3CE	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC3CF	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC3D0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC3D1	RCAN_0	16	2+nPφ/3+nPφ
MB_0[23].	CONTROL0H	—	16	H'FFC3E0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC3E2	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC3E4	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC3E6	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC3E8	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC3E9	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC3EA	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC3EB	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC3EC	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC3ED	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC3EE	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC3EF	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC3F0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC3F1	RCAN_0	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[24].	CONTROL0H	—	16	H'FFC400	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC402	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC404	RCAN_0	16	2+nPφ3+nPφ
	LAFML	—	16	H'FFC406	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC408	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC409	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC40A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC40B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC40C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC40D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC40E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC40F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC410	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC411	RCAN_0	16	2+nPφ/3+nPφ
	TTT	—	16	H'FFC414	RCAN_0	16	2Pφ/3Pφ
	TTCONTROL	—	16	H'FFC416	RCAN_0	16	2Pφ/3Pφ
MB_0[25].	CONTROL0H	—	16	H'FFC420	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC422	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC424	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC426	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC428	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC429	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC42A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC42B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC42C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC42D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC42E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC42F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC430	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC431	RCAN_0	16	2+nPφ3+nPφ
	TTT	—	16	H'FFC434	RCAN_0	16	2Pφ/3Pφ
	TTCONTROL	—	16	H'FFC436	RCAN_0	16	2Pφ/3Pφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[26].	CONTROL0H	—	16	H'FFC440	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFC442	RCAN_0	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFC444	RCAN_0	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFC446	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFC448	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFC449	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFC44A	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFC44B	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFC44C	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFC44D	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFC44E	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFC44F	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFC450	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFC451	RCAN_0	16	2+nPϕ/3+nPϕ
	TTT	—	16	H'FFC454	RCAN_0	16	2Pϕ/3Pϕ
	TTCONTROL	—	16	H'FFC456	RCAN_0	16	2Pϕ/3Pϕ
MB_0[27].	CONTROL0H	—	16	H'FFC460	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFC462	RCAN_0	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFC464	RCAN_0	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFC466	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFC468	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFC469	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFC46A	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFC46B	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFC46C	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFC46D	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFC46E	RCAN_0	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFC46F	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFC470	RCAN_0	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFC471	RCAN_0	16	2+nPϕ/3+nPϕ
	TTT	—	16	H'FFC474	RCAN_0	16	2Pϕ/3Pϕ
	TTCONTROL	—	16	H'FFC476	RCAN_0	16	2Pϕ/3Pϕ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[28].	CONTROL0H	—	16	H'FFC480	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC482	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC484	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC486	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC488	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC489	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC48A	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC48B	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC48C	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC48D	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC48E	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC48F	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC490	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC491	RCAN_0	16	2+nPφ/3+nPφ
	TTT	—	16	H'FFC494	RCAN_0	16	2Pφ/3Pφ
	TTCONTROL	—	16	H'FFC496	RCAN_0	16	2Pφ/3Pφ
MB_0[29].	CONTROL0H	—	16	H'FFC4A0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC4A2	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC4A4	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC4A6	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC4A8	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC4A9	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC4AA	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC4AB	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC4AC	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC4AD	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC4AE	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC4AF	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC4B0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC4B1	RCAN_0	16	2+nPφ/3+nPφ
	TTT	—	16	H'FFC4B4	RCAN_0	16	2Pφ/3Pφ
	TTCONTROL	—	16	H'FFC4B6	RCAN_0	16	2Pφ/3Pφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_0[30].	CONTROL0H	—	16	H'FFC4C0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC4C2	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC4C4	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC4C6	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC4C8	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC4C9	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC4CA	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC4CB	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC4CC	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC4CD	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC4CE	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC4CF	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC4D0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC4D1	RCAN_0	16	2+nPφ/3+nPφ
	TIMSTAMP	—	16	H'FFC4D2	RCAN_0	16	2Pφ/3Pφ
	TTT	—	16	H'FFC4D4	RCAN_0	16	2Pφ/3Pφ
MB_0[31].	CONTROL0H	—	16	H'FFC4E0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFC4E2	RCAN_0	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFC4E4	RCAN_0	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFC4E6	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFC4E8	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFC4E9	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFC4EA	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFC4EB	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFC4EC	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFC4ED	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFC4EE	RCAN_0	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFC4EF	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFC4F0	RCAN_0	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFC4F1	RCAN_0	16	2+nPφ/3+nPφ
	TIMSTAMP	—	16	H'FFC4F2	RCAN_0	16	2Pφ/3Pφ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
RCAN-TL1 monitor register_0	RCANMON_0	8	H'FFC800	RCAN_0	16	2Pφ/2Pφ
RCAN-TL1 monitor register_1	RCANMON_1	8	H'FFC801	RCAN_1	16	2Pφ/2Pφ
Master control register_1	MCR_1	16	H'FFD000	RCAN_1	16	2Pφ/3Pφ
General status register_1	GSR_1	16	H'FFD002	RCAN_1	16	2Pφ/3Pφ
Bit configuration register 1_1	BCR1_1	16	H'FFD004	RCAN_1	16	2Pφ/3Pφ
Bit configuration register 0_1	BCR0_1	16	H'FFD006	RCAN_1	16	2Pφ/3Pφ
Interrupt request register_1	IRR_1	16	H'FFD008	RCAN_1	16	2Pφ/3Pφ
Interrupt mask register_1	IMR_1	16	H'FFD00A	RCAN_1	16	2Pφ/3Pφ
Transmit error counter _1	TEC_1	16	H'FFD00C	RCAN_1	16	2Pφ/3Pφ
Receive error counter _1	REC_1					
Transmission pending register 1_1	TXPR1_1	16	H'FFD020	RCAN_1	32	4Pφ/6Pφ
Transmission pending register 0_1	TXPR0_1	16				
Transmit cancel register 1_1	TXCR1_1	16	H'FFD028	RCAN_1	16	2Pφ/3Pφ
Transmit cancel register 0_1	TXCR0_1	16	H'FFD02A	RCAN_1	16	2Pφ/3Pφ
Transmit acknowledge register 1_1	TXACK1_1	16	H'FFD030	RCAN_1	16	2Pφ/3Pφ
Transmit acknowledge register 0_1	TXACK0_1	16	H'FFD032	RCAN_1	16	2Pφ/3Pφ
Abort acknowledgement register 1_1	ABACK1_1	16	H'FFD038	RCAN_1	16	2Pφ/3Pφ
Abort acknowledgement register 0_1	ABACK0_1	16	H'FFD03A	RCAN_1	16	2Pφ/3Pφ
Data frame receive pending register 1_1	RXPR1_1	16	H'FFD040	RCAN_1	16	2Pφ/3Pφ
Data frame receive pending register 0_1	RXPR0_1	16	H'FFD042	RCAN_1	16	2Pφ/3Pφ
Remote frame receive pending register 1_1	RFPR1_1	16	H'FFD048	RCAN_1	16	2Pφ/3Pφ
Remote frame receive pending register 0_1	RFPR0_1	16	H'FFD04A	RCAN_1	16	2Pφ/3Pφ
Mailbox interrupt mask register1_1	MBIMR1_1	16	H'FFD050	RCAN_1	16	2Pφ/3Pφ
Mailbox interrupt mask register0_1	MBIMR0_1	16	H'FFD052	RCAN_1	16	2Pφ/3Pφ
Unread message status register 1_1	UMSR1_1	16	H'FFD058	RCAN_1	16	2Pφ/3Pφ
Unread message status register 0_1	UMSR0_1	16	H'FFD05A	RCAN_1	16	2Pφ/3Pφ
Time trigger control register 0_1	TTCR0_1	16	H'FFD080	RCAN_1	16	2Pφ/3Pφ
Cycle maximum/Tx-Enable Window register_1	CMAX_TEW_1	16	H'FFD084	RCAN_1	16	2Pφ/3Pφ
Reference trigger offset register_1	RFTROFF_1	16	H'FFD086	RCAN_1	16	2Pφ/3Pφ
Timer status register_1	TSR_1	16	H'FFD088	RCAN_1	16	2Pφ/3Pφ
Cycle counter register_1	CCR_1	16	H'FFD08A	RCAN_1	16	2Pφ/3Pφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
Timer counter register_1		TCNTR_1	16	H'FFD08C	RCAN_1	16	2Pϕ/3Pϕ
Cycle time register_1		CYCTR_1	16	H'FFD090	RCAN_1	16	2Pϕ/3Pϕ
Reference mark register_1		RFMK_1	16	H'FFD094	RCAN_1	16	2Pϕ/3Pϕ
Timer compare match register 0_1		TCMR0_1	16	H'FFD098	RCAN_1	16	2Pϕ/3Pϕ
Timer compare match register 1_1		TCMR1_1	16	H'FFD09C	RCAN_1	16	2Pϕ/3Pϕ
Timer compare match register 2_1		TCMR2_1	16	H'FFD0A0	RCAN_1	16	2Pϕ/3Pϕ
Tx-trigger time selection register_1		TTTSEL_1	16	H'FFD0A4	RCAN_1	16	2Pϕ/3Pϕ
MB_1[0].	CONTROL0H	—	16	H'FFD100	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFD102	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFD104	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFD106	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFD108	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFD109	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFD10A	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFD10B	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFD10C	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFD10D	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFD10E	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFD10F	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFD110	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFD111	RCAN_1	16	2+nPϕ/3+nPϕ
	TIMESTAMP	—	16	H'FFD112	RCAN_1	16	2Pϕ/3Pϕ
MB_1[1].	CONTROL0H	—	16	H'FFD120	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFD122	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFD124	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFD126	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFD128	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFD129	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFD12A	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFD12B	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFD12C	RCAN_1	16	2+nPϕ/3+nPϕ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[1].	MSG_DATA[5]	—	8	H'FFD12D	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD12E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD12F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD130	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD131	RCAN_1	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFD132	RCAN_1	16	2Pφ/3Pφ
MB_1[2].	CONTROL0H	—	16	H'FFD140	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD142	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD144	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD146	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD148	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD149	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD14A	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD14B	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD14C	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD14D	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD14E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD14F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD150	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD151	RCAN_1	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFD152	RCAN_1	16	2Pφ/3Pφ
MB_1[3].	CONTROL0H	—	16	H'FFD160	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD162	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD164	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD166	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD168	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD169	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD16A	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD16B	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD16C	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD16D	RCAN_1	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[3].	MSG_DATA[6]	—	8	H'FFD16E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD16F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD170	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD171	RCAN_1	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFD172	RCAN_1	16	2Pφ/3Pφ
MB_1[4].	CONTROL0H	—	16	H'FFD180	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD182	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD184	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD186	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD188	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD189	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD18A	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD18B	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD18C	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD18D	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD18E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD18F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD190	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD191	RCAN_1	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFD192	RCAN_1	16	2Pφ/3Pφ
MB_1[5].	CONTROL0H	—	16	H'FFD1A0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD1A2	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD1A4	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD1A6	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD1A8	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD1A9	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD1AA	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD1AB	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD1AC	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD1AD	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD1AE	RCAN_1	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[5].	MSG_DATA[7]	—	8	H'FFD1AF	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD1B0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD1B1	RCAN_1	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFD1B2	RCAN_1	16	2Pφ/3Pφ
MB_1[6].	CONTROL0H	—	16	H'FFD1C0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD1C2	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD1C4	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD1C6	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD1C8	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD1C9	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD1CA	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD1CB	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD1CC	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD1CD	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD1CE	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD1CF	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD1D0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD1D1	RCAN_1	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFD1D2	RCAN_1	16	2Pφ/3Pφ
MB_1[7].	CONTROL0H	—	16	H'FFD1E0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD1E2	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD1E4	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD1E6	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD1E8	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD1E9	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD1EA	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD1EB	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD1EC	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD1ED	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD1EE	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD1EF	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD1F0	RCAN_1	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[7].	CONTROL1L	—	8	H'FFD1F1	RCAN_1	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFD1F2	RCAN_1	16	2Pφ/3Pφ
MB_1[8].	CONTROL0H	—	16	H'FFD200	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD202	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD204	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD206	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD208	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD209	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD20A	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD20B	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD20C	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD20D	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD20E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD20F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD210	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0H	—	8	H'FFD211	RCAN_1	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFD212	RCAN_1	16	2Pφ/3Pφ
MB_1[9].	CONTROL0H	—	16	H'FFD220	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD222	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD224	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD226	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD228	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD229	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD22A	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD22B	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD22C	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD22D	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD22E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD22F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD230	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD231	RCAN_1	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFD232	RCAN_1	16	2Pφ/3Pφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[10].	CONTROL0H	—	16	H'FFD240	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFD242	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFD244	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFD246	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFD248	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFD249	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFD24A	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFD24B	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFD24C	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFD24D	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFD24E	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFD24F	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFD250	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFD251	RCAN_1	16	2+nPϕ/3+nPϕ
	TIMESTAMP	—	16	H'FFD252	RCAN_1	16	2Pϕ/3Pϕ
MB_1[11].	CONTROL0H	—	16	H'FFD260	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFD262	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFD264	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFD266	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFD268	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFD269	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFD26A	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFD26B	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFD26C	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFD26D	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFD26E	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFD26F	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFD270	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFD271	RCAN_1	16	2+nPϕ/3+nPϕ
	TIMESTAMP	—	16	H'FFD272	RCAN_1	16	2Pϕ/3Pϕ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[12].	CONTROL0H	—	16	H'FFD280	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD282	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD284	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD286	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD288	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD289	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD28A	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD28B	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD28C	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD28D	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD28E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD28F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD290	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD291	RCAN_1	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFD292	RCAN_1	16	2Pφ/3Pφ
MB_1[13].	CONTROL0H	—	16	H'FFD2A0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD2A2	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD2A4	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD2A6	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD2A8	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD2A9	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD2AA	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD2AB	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD2AC	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD2AD	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD2AE	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD2AF	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD2B0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD2B1	RCAN_1	16	2+nPφ/3+nPφ
	TIMESTAMP	—	16	H'FFD2B2	RCAN_1	16	2Pφ/3Pφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[14].	CONTROL0H	—	16	H'FFD2C0	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFD2C2	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFD2C4	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFD2C6	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFD2C8	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFD2C9	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFD2CA	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFD2CB	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFD2CC	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFD2CD	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFD2CE	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFD2CF	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFD2D0	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFD2D1	RCAN_1	16	2+nPϕ/3+nPϕ
	TIMESTAMP	—	16	H'FFD2D2	RCAN_1	16	2Pϕ/3Pϕ
MB_1[15].	CONTROL0H	—	16	H'FFD2E0	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFD2E2	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFD2E4	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFD2E6	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFD2E8	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFD2E9	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFD2EA	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFD2EB	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFD2EC	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFD2ED	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFD2EE	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFD2EF	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFD2F0	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFD2F1	RCAN_1	16	2+nPϕ/3+nPϕ
	TIMESTAMP	—	16	H'FFD2F2	RCAN_1	16	2Pϕ/3Pϕ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[16].	CONTROL0H	—	16	H'FFD300	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD302	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD304	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD306	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD308	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD309	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD30A	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD30B	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD30C	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD30D	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD30E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD30F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD310	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD311	RCAN_1	16	2+nPφ/3+nPφ
MB_1[17].	CONTROL0H	—	16	H'FFD320	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD322	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD324	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD326	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD328	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD329	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD32A	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD32B	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD32C	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD32D	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD32E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD32F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD330	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD331	RCAN_1	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[18].	CONTROL0H	—	16	H'FFD340	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD342	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD344	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD346	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD348	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD349	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD34A	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD34B	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD34C	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD34D	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD34E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD34F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD350	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD351	RCAN_1	16	2+nPφ/3+nPφ
MB_1[19].	CONTROL0H	—	16	H'FFD360	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD362	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD364	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD366	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD368	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD369	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD36A	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD36B	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD36C	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD36D	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD36E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD36F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD370	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD371	RCAN_1	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[20].	CONTROL0H	—	16	H'FFD380	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD382	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD384	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD386	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD388	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD389	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD38A	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD38B	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD38C	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD38D	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD38E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD38F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD390	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD391	RCAN_1	16	2+nPφ/3+nPφ
MB_1[21].	CONTROL0H	—	16	H'FFD3A0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD3A2	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD3A4	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD3A6	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD3A8	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD3A9	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD3AA	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD3AB	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD3AC	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD3AD	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD3AE	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD3AF	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD3B0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD3B1	RCAN_1	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[22].	CONTROL0H	—	16	H'FFD3C0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD3C2	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD3C4	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD3C6	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD3C8	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD3C9	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD3CA	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD3CB	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD3CC	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD3CD	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD3CE	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD3CF	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD3D0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD3D1	RCAN_1	16	2+nPφ/3+nPφ
MB_1[23].	CONTROL0H	—	16	H'FFD3E0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD3E2	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD3E4	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD3E6	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD3E8	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD3E9	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD3EA	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD3EB	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD3EC	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD3ED	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD3EE	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD3EF	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD3F0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD3F1	RCAN_1	16	2+nPφ/3+nPφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[24].	CONTROL0H	—	16	H'FFD400	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFD402	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFD404	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFD406	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFD408	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFD409	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFD40A	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFD40B	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFD40C	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFD40D	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFD40E	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFD40F	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFD410	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFD411	RCAN_1	16	2+nPϕ/3+nPϕ
	TTT	—	16	H'FFD414	RCAN_1	16	2Pϕ/3Pϕ
	TTCONTROL	—	16	H'FFD416	RCAN_1	16	2Pϕ/3Pϕ
MB_1[25].	CONTROL0H	—	16	H'FFD420	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFD422	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFD424	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFD426	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFD428	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFD429	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFD42A	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFD42B	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFD42C	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFD42D	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFD42E	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFD42F	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFD430	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFD431	RCAN_1	16	2+nPϕ/3+nPϕ
	TTT	—	16	H'FFD434	RCAN_1	16	2Pϕ/3Pϕ
	TTCONTROL	—	16	H'FFD436	RCAN_1	16	2Pϕ/3Pϕ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[26].	CONTROL0H	—	16	H'FFD440	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD442	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD444	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD446	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD448	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD449	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD44A	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD44B	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD44C	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD44D	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD44E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD44F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD450	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD451	RCAN_1	16	2+nPφ/3+nPφ
	TTT	—	16	H'FFD454	RCAN_1	16	2Pφ/3Pφ
	TTCONTROL	—	16	H'FFD456	RCAN_1	16	2Pφ/3Pφ
MB_1[27].	CONTROL0H	—	16	H'FFD460	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD462	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD464	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD466	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD468	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD469	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD46A	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD46B	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD46C	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD46D	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD46E	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD46F	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD470	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD471	RCAN_1	16	2+nPφ/3+nPφ
	TTT	—	16	H'FFD474	RCAN_1	16	2Pφ/3Pφ
	TTCONTROL	—	16	H'FFD476	RCAN_1	16	2Pφ/3Pφ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[28].	CONTROL0H	—	16	H'FFD480	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFD482	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFD484	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFD486	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFD488	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFD489	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFD48A	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFD48B	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFD48C	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFD48D	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFD48E	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFD48F	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFD490	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFD491	RCAN_1	16	2+nPϕ/3+nPϕ
	TTT	—	16	H'FFD494	RCAN_1	16	2Pϕ/3Pϕ
	TTCONTROL	—	16	H'FFD496	RCAN_1	16	2Pϕ/3Pϕ
MB_1[29].	CONTROL0H	—	16	H'FFD4A0	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL0L	—	16	H'FFD4A2	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFMH	—	16	H'FFD4A4	RCAN_1	16	2+nPϕ/3+nPϕ
	LAFML	—	16	H'FFD4A6	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[0]	—	8	H'FFD4A8	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[1]	—	8	H'FFD4A9	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[2]	—	8	H'FFD4AA	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[3]	—	8	H'FFD4AB	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[4]	—	8	H'FFD4AC	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[5]	—	8	H'FFD4AD	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[6]	—	8	H'FFD4AE	RCAN_1	16	2+nPϕ/3+nPϕ
	MSG_DATA[7]	—	8	H'FFD4AF	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1H	—	8	H'FFD4B0	RCAN_1	16	2+nPϕ/3+nPϕ
	CONTROL1L	—	8	H'FFD4B1	RCAN_1	16	2+nPϕ/3+nPϕ
	TTT	—	16	H'FFD4B4	RCAN_1	16	2Pϕ/3Pϕ
	TTCONTROL	—	16	H'FFD4B6	RCAN_1	16	2Pϕ/3Pϕ

Register Name		Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
MB_1[30].	CONTROL0H	—	16	H'FFD4C0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD4C2	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD4C4	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD4C6	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD4C8	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD4C9	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD4CA	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD4CB	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD4CC	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD4CD	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD4CE	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD4CF	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD4D0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD4D1	RCAN_1	16	2+nPφ/3+nPφ
	TIMSTAMP	—	16	H'FFD4D2	RCAN_1	16	2Pφ/3Pφ
	TTT	—	16	H'FFD4D4	RCAN_1	16	2Pφ/3Pφ
MB_1[31].	CONTROL0H	—	16	H'FFD4E0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL0L	—	16	H'FFD4E2	RCAN_1	16	2+nPφ/3+nPφ
	LAFMH	—	16	H'FFD4E4	RCAN_1	16	2+nPφ/3+nPφ
	LAFML	—	16	H'FFD4E6	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[0]	—	8	H'FFD4E8	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[1]	—	8	H'FFD4E9	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[2]	—	8	H'FFD4EA	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[3]	—	8	H'FFD4EB	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[4]	—	8	H'FFD4EC	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[5]	—	8	H'FFD4ED	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[6]	—	8	H'FFD4EE	RCAN_1	16	2+nPφ/3+nPφ
	MSG_DATA[7]	—	8	H'FFD4EF	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1H	—	8	H'FFD4F0	RCAN_1	16	2+nPφ/3+nPφ
	CONTROL1L	—	8	H'FFD4F1	RCAN_1	16	2+nPφ/3+nPφ
	TIMSTAMP	—	16	H'FFD4F2	RCAN_1	16	2Pφ/3Pφ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
RSPI control register A	SPCRA	8	H'FFD800	RSPI	16	2Rφ/2Rφ
RSPI slave select polarity register A	SSLPA	8	H'FFD801	RSPI	16	2Rφ/2Rφ
RSPI pin control register A	SPPCRA	8	H'FFD802	RSPI	16	2Rφ/2Rφ
RSPI status register A	SPSRA	8	H'FFD803	RSPI	16	2Rφ/2Rφ
RSPI data register A	SPDRA	32	H'FFD804	RSPI	16	2Rφ/2Rφ
RSPI sequence control register A	SPSCRA	8	H'FFD808	RSPI	16	2Rφ/2Rφ
RSPI sequence status register A	SPSSRA	8	H'FFD809	RSPI	16	2Rφ/2Rφ
RSPI bit rate register A	SPBRA	8	H'FFD80A	RSPI	16	2Rφ/2Rφ
RSPI data control register A	SPDCRA	8	H'FFD80B	RSPI	16	2Rφ/2Rφ
RSPI clock delay register A	SPCKDA	8	H'FFD80C	RSPI	16	2Rφ/2Rφ
RSPI slave select negation delay register A	SSLNDA	8	H'FFD80D	RSPI	16	2Rφ/2Rφ
RSPI next-access delay register A	SPNDA	8	H'FFD80E	RSPI	16	2Rφ/2Rφ
RSPI control register 2A	SPCR2A	8	H'FFD80F	RSPI	16	2Rφ/2Rφ
RSPI command register A0	SPCMDA0	16	H'FFD810	RSPI	16	2Rφ/2Rφ
RSPI command register A1	SPCMDA1	16	H'FFD812	RSPI	16	2Rφ/2Rφ
RSPI command register A2	SPCMDA2	16	H'FFD814	RSPI	16	2Rφ/2Rφ
RSPI command register A3	SPCMDA3	16	H'FFD816	RSPI	16	2Rφ/2Rφ
RSPI command register A4	SPCMDA4	16	H'FFD818	RSPI	16	2Rφ/2Rφ
RSPI command register A5	SPCMDA5	16	H'FFD81A	RSPI	16	2Rφ/2Rφ
RSPI command register A6	SPCMDA6	16	H'FFD81C	RSPI	16	2Rφ/2Rφ
RSPI command register A7	SPCMDA7	16	H'FFD81E	RSPI	16	2Rφ/2Rφ
RSPI control register B	SPCRB	8	H'FFD820	RSPI	16	2Rφ/2Rφ
RSPI slave select polarity register B	SSLPB	8	H'FFD821	RSPI	16	2Rφ/2Rφ
RSPI pin control register B	SPPCRB	8	H'FFD822	RSPI	16	2Rφ/2Rφ
RSPI status register B	SPSRB	8	H'FFD823	RSPI	16	2Rφ/2Rφ
RSPI data register B	SPDRB	32	H'FFD824	RSPI	16	2Rφ/2Rφ
RSPI sequence control register B	SPSCRB	8	H'FFD828	RSPI	16	2Rφ/2Rφ
RSPI sequence status register B	SPSSRB	8	H'FFD829	RSPI	16	2Rφ/2Rφ
RSPI bit rate register B	SPBRB	8	H'FFD82A	RSPI	16	2Rφ/2Rφ
RSPI data control register B	SPDCRB	8	H'FFD82B	RSPI	16	2Rφ/2Rφ
RSPI clock delay register B	SPCKDB	8	H'FFD82C	RSPI	16	2Rφ/2Rφ
RSPI slave select negation delay register B	SSLNDB	8	H'FFD82D	RSPI	16	2Rφ/2Rφ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
RSPI next-access delay register B	SPNDB	8	H'FFD82E	RSPI	16	2Rφ/2Rφ
RSPI control register 2B	SPCR2B	8	H'FFD82F	RSPI	16	2Rφ/2Rφ
RSPI command register B0	SPCMDB0	16	H'FFD830	RSPI	16	2Rφ/2Rφ
RSPI command register B1	SPCMDB1	16	H'FFD832	RSPI	16	2Rφ/2Rφ
RSPI command register B2	SPCMDB2	16	H'FFD834	RSPI	16	2Rφ/2Rφ
RSPI command register B3	SPCMDB3	16	H'FFD836	RSPI	16	2Rφ/2Rφ
RSPI command register B4	SPCMDB4	16	H'FFD838	RSPI	16	2Rφ/2Rφ
RSPI command register B5	SPCMDB5	16	H'FFD83A	RSPI	16	2Rφ/2Rφ
RSPI command register B6	SPCMDB6	16	H'FFD83C	RSPI	16	2Rφ/2Rφ
RSPI command register B7	SPCMDB7	16	H'FFD83E	RSPI	16	2Rφ/2Rφ
RSPI control register C	SPCRC	8	H'FFD840	RSPI	16	2Rφ/2Rφ
RSPI slave select polarity register C	SSLPC	8	H'FFD841	RSPI	16	2Rφ/2Rφ
RSPI pin control register C	SPPCRC	8	H'FFD842	RSPI	16	2Rφ/2Rφ
RSPI status register C	SPSRC	8	H'FFD843	RSPI	16	2Rφ/2Rφ
RSPI data register C	SPDRC	32	H'FFD844	RSPI	16	2Rφ/2Rφ
RSPI sequence control register C	SPSCRC	8	H'FFD848	RSPI	16	2Rφ/2Rφ
RSPI sequence status register C	SPSSRC	8	H'FFD849	RSPI	16	2Rφ/2Rφ
RSPI bit rate register C	SPBRC	8	H'FFD84A	RSPI	16	2Rφ/2Rφ
RSPI data control register C	SPDCRC	8	H'FFD84B	RSPI	16	2Rφ/2Rφ
RSPI clock delay register C	SPCKDC	8	H'FFD84C	RSPI	16	2Rφ/2Rφ
RSPI slave select negation delay register C	SSLNDC	8	H'FFD84D	RSPI	16	2Rφ/2Rφ
RSPI next-access delay register C	SPNDC	8	H'FFD84E	RSPI	16	2Rφ/2Rφ
RSPI control register 2C	SPCR2C	8	H'FFD84F	RSPI	16	2Rφ/2Rφ
RSPI command register C0	SPCMDC0	16	H'FFD850	RSPI	16	2Rφ/2Rφ
RSPI command register C1	SPCMDC1	16	H'FFD852	RSPI	16	2Rφ/2Rφ
RSPI command register C2	SPCMDC2	16	H'FFD854	RSPI	16	2Rφ/2Rφ
RSPI command register C3	SPCMDC3	16	H'FFD856	RSPI	16	2Rφ/2Rφ
RSPI command register C4	SPCMDC4	16	H'FFD858	RSPI	16	2Rφ/2Rφ
RSPI command register C5	SPCMDC5	16	H'FFD85A	RSPI	16	2Rφ/2Rφ
RSPI command register C6	SPCMDC6	16	H'FFD85C	RSPI	16	2Rφ/2Rφ
RSPI command register C7	SPCMDC7	16	H'FFD85E	RSPI	16	2Rφ/2Rφ
RSPI control register D	SPCRD	8	H'FFD860	RSPI	16	2Rφ/2Rφ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
RSPI slave select polarity register D	SSLPD	8	H'FFD861	RSPI	16	2Rφ/2Rφ
RSPI pin control register D	SPPCRD	8	H'FFD862	RSPI	16	2Rφ/2Rφ
RSPI status register D	SPSRD	8	H'FFD863	RSPI	16	2Rφ/2Rφ
RSPI data register D	SPDRD	32	H'FFD864	RSPI	16	2Rφ/2Rφ
RSPI sequence control register D	SPSCRD	8	H'FFD868	RSPI	16	2Rφ/2Rφ
RSPI sequence status register D	SPSSRD	8	H'FFD869	RSPI	16	2Rφ/2Rφ
RSPI bit rate register D	SPBRD	8	H'FFD86A	RSPI	16	2Rφ/2Rφ
RSPI data control register D	SPDCRD	8	H'FFD86B	RSPI	16	2Rφ/2Rφ
RSPI clock delay register D	SPCKDD	8	H'FFD86C	RSPI	16	2Rφ/2Rφ
RSPI slave select negation delay register D	SSLNDD	8	H'FFD86D	RSPI	16	2Rφ/2Rφ
RSPI next-access delay register D	SPNDD	8	H'FFD86E	RSPI	16	2Rφ/2Rφ
RSPI control register 2D	SPCR2D	8	H'FFD86F	RSPI	16	2Rφ/2Rφ
RSPI command register D0	SPCMDD0	16	H'FFD870	RSPI	16	2Rφ/2Rφ
RSPI command register D1	SPCMDD1	16	H'FFD872	RSPI	16	2Rφ/2Rφ
RSPI command register D2	SPCMDD2	16	H'FFD874	RSPI	16	2Rφ/2Rφ
RSPI command register D3	SPCMDD3	16	H'FFD876	RSPI	16	2Rφ/2Rφ
RSPI command register D4	SPCMDD4	16	H'FFD878	RSPI	16	2Rφ/2Rφ
RSPI command register D5	SPCMDD5	16	H'FFD87A	RSPI	16	2Rφ/2Rφ
RSPI command register D6	SPCMDD6	16	H'FFD87C	RSPI	16	2Rφ/2Rφ
RSPI command register D7	SPCMDD7	16	H'FFD87E	RSPI	16	2Rφ/2Rφ
LIN control register	LINCR	8	H'FFD900	HWLIN	16	2Pφ/2Pφ
LIN status register	LINSTR	8	H'FFD901	HWLIN	16	2Pφ/2Pφ
LIN timer control register	LINTCR	8	H'FFD902	HWLIN	16	2Pφ/2Pφ
	—		H'FFD903	HWLIN	16	
LIN timer counter	LINTCNT	16	H'FFD904	HWLIN	16	2Pφ/2Pφ
LIN timeout counter	LINTOCNT	16	H'FFD906	HWLIN	16	2Pφ/2Pφ
	—	—	H'FFD908 to H'FFD9FF	HWLIN	16	
CRC control register	CRCCR	8	H'FFDA4C	CRC	16	2Pφ/2Pφ
CRC data input register	CRCDIR	8	H'FFDA4D	CRC	16	2Pφ/2Pφ
CRC data output register	CRCDOR	16	H'FFDA4E	CRC	16	2Pφ/2Pφ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
Flash mode register	FMODR	8	H'FFE002	FLASH/ EEPROM	8	2Fφ/3Fφ
Flash access status register	FASTAT	8	H'FFE010	FLASH/ EEPROM	8	2Fφ/3Fφ
Flash access error interrupt enable register	FAEINT	8	H'FFE011	FLASH/ EEPROM	8	2Fφ/3Fφ
Flash memory MAT select register	ROMMAT	16	H'FFE020	FLASH/ EEPROM	16	2Fφ/3Fφ
EEPROM read enable register 0	EEPRE0	16	H'FFE040	FLASH/ EEPROM	16	2Fφ/3Fφ
EEPROM read enable register 1	EEPRE1	16	H'FFE042	FLASH/ EEPROM	16	2Fφ/3Fφ
EEPROM program/erase enable register 0	EEPWE0	16	H'FFE050	FLASH/ EEPROM	16	2Fφ/3Fφ
EEPROM program/erase enable register 1	EEPWE1	16	H'FFE052	FLASH/ EEPROM	16	2Fφ/3Fφ
FCU RAM enable register	FCURAME	16	H'FFE054	FLASH/ EEPROM	16	2Fφ/3Fφ
Flash status register 0	FSTATR0	8	H'FFE100	FLASH/ EEPROM	8	2Fφ/3Fφ
Flash status register 1	FSTATR1	8	H'FFE101	FLASH/ EEPROM	8	2Fφ/3Fφ
Flash P/E mode entry register	FENTRYR	16	H'FFE102	FLASH/ EEPROM	16	2Fφ/3Fφ
Flash protect register	FPROTR	16	H'FFE104	FLASH/ EEPROM	16	2Fφ/3Fφ
Flash reset register	FRESETR	16	H'FFE106	FLASH/ EEPROM	16	2Fφ/3Fφ
FCU command register	FCMDR	16	H'FFE10A	FLASH/ EEPROM	16	2Fφ/3Fφ
FCU RAM ECC error control register	FRAMECCR	8	H'FFE10C	FLASH/ EEPROM	8	2Fφ/3Fφ
FCU processing switch register	FCPSR	16	H'FFE118	FLASH/ EEPROM	16	2Fφ/3Fφ
EEPROM blank check control register	EEPBCCNT	16	H'FFE11A	FLASH/ EEPROM	16	2Fφ/3Fφ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
Flash P/E status register1	FPESTAT	16	H'FFE11C	FLASH/ EEPROM	16	2Fφ/3Fφ
EEPROM blank check status register	EEPBCSTAT	16	H'FFE11E	FLASH/ EEPROM	16	2Fφ/3Fφ
Flash clock notification register	FCKAR	16	H'FFE138	FLASH/ EEPROM	16	2Fφ/3Fφ
EEPROM MAT select register	EEPMAT	16	H'FFE380	FLASH/ EEPROM	16	2Fφ/2Fφ
RAM enable control register	RAMEN	16	H'FFE400	RAM	16	2Pφ/2Pφ
RAM write enable control register	RAMWEN	16	H'FFE402	RAM	16	2Pφ/2Pφ
RAM_ECC enable control register	RAMECC	16	H'FFE404	RAM	16	2Pφ/2Pφ
RAM error status register	RAMERR	8	H'FFE406	RAM	8	2Pφ/2Pφ
RAM error interrupt control register	RAMINT	8	H'FFE410	RAM	8	2Pφ/2Pφ
RAM access cycle set register	RAMACYC	16	H'FFE412	RAM	16	2Pφ/2Pφ
A/D data register A_0	ADDRA_0	16	H'FFE500	A/D_0	16	2Aφ/2Aφ
A/D data register B_0	ADDRB_0	16	H'FFE502	A/D_0	16	2Aφ/2Aφ
A/D data register C_0	ADDRC_0	16	H'FFE504	A/D_0	16	2Aφ/2Aφ
A/D data register D_0	ADDRD_0	16	H'FFE506	A/D_0	16	2Aφ/2Aφ
A/D data register E_0	ADDRE_0	16	H'FFE508	A/D_0	16	2Aφ/2Aφ
A/D data register F_0	ADDRF_0	16	H'FFE50A	A/D_0	16	2Aφ/2Aφ
A/D data register G_0	ADDRG_0	16	H'FFE50C	A/D_0	16	2Aφ/2Aφ
A/D data register H_0	ADDRH_0	16	H'FFE50E	A/D_0	16	2Aφ/2Aφ
A/D control/status register_0	ADCSR_0	8	H'FFE510	A/D_0	16	2Aφ/2Aφ
A/D control register_0	ADCR_0	8	H'FFE511	A/D_0	16	2Aφ/2Aφ
Analog port pull-down control register_0	APPDCR_0	8	H'FFE51C	A/D_0	16	2Aφ/2Aφ
A/D self-diagnostic register 0	ADDIAGR_0	8	H'FFE51D	A/D_0	16	2Aφ/2Aφ
A/D data register A_1	ADDRA_1	16	H'FFE520	A/D_1	16	2Aφ/2Aφ
A/D data register B_1	ADDRB_1	16	H'FFE522	A/D_1	16	2Aφ/2Aφ
A/D data register C_1	ADDRC_1	16	H'FFE524	A/D_1	16	2Aφ/2Aφ
A/D data register D_1	ADDRD_1	16	H'FFE526	A/D_1	16	2Aφ/2Aφ
A/D data register E_1	ADDRE_1	16	H'FFE528	A/D_1	16	2Aφ/2Aφ
A/D data register F_1	ADDRF_1	16	H'FFE52A	A/D_1	16	2Aφ/2Aφ
A/D data register G_1	ADDRG_1	16	H'FFE52C	A/D_1	16	2Aφ/2Aφ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
A/D data register H_1	ADDRH_1	16	H'FFE52E	A/D_1	16	2Aφ/2Aφ
A/D control/status register_1	ADCSR_1	8	H'FFE530	A/D_1	16	2Aφ/2Aφ
A/D control register_1	ADCR_1	8	H'FFE531	A/D_1	16	2Aφ/2Aφ
Analog port pull-down control register_1	APPDCR_1	8	H'FFE53C	A/D_1	16	2Aφ/2Aφ
A/D self-diagnostic register 1	ADDIAGR_1	8	H'FFE53D	A/D_1	16	2Aφ/2Aφ
Timer start register	TSTRB	8	H'FFFB00	TPU (unit 1)	16	2Pφ/2Pφ
Timer synchronous register	TSYRB	8	H'FFFB01	TPU (unit 1)	16	2Pφ/2Pφ
Timer control register_6	TCR_6	8	H'FFFB10	TPU_6	16	2Pφ/2Pφ
Timer mode register_6	TMDR_6	8	H'FFFB11	TPU_6	16	2Pφ/2Pφ
Timer I/O control register H_6	TIORH_6	8	H'FFFB12	TPU_6	16	2Pφ/2Pφ
Timer I/O control register L_6	TIORL_6	8	H'FFFB13	TPU_6	16	2Pφ/2Pφ
Timer interrupt enable register_6	TIER_6	8	H'FFFB14	TPU_6	16	2Pφ/2Pφ
Timer status register_6	TSR_6	8	H'FFFB15	TPU_6	16	2Pφ/2Pφ
Timer counter_6	TCNT_6	16	H'FFFB16	TPU_6	16	2Pφ/2Pφ
Timer general register A_6	TGRA_6	16	H'FFFB18	TPU_6	16	2Pφ/2Pφ
Timer general register B_6	TGRB_6	16	H'FFFB1A	TPU_6	16	2Pφ/2Pφ
Timer general register C_6	TGRC_6	16	H'FFFB1C	TPU_6	16	2Pφ/2Pφ
Timer general register D_6	TGRD_6	16	H'FFFB1E	TPU_6	16	2Pφ/2Pφ
Timer control register_7	TCR_7	8	H'FFFB20	TPU_7	16	2Pφ/2Pφ
Timer mode register_7	TMDR_7	8	H'FFFB21	TPU_7	16	2Pφ/2Pφ
Timer I/O control register_7	TIOR_7	8	H'FFFB22	TPU_7	16	2Pφ/2Pφ
	—		H'FFFB23	TPU_7	16	
Timer interrupt enable register_7	TIER_7	8	H'FFFB24	TPU_7	16	2Pφ/2Pφ
Timer status register_7	TSR_7	8	H'FFFB25	TPU_7	16	2Pφ/2Pφ
Timer counter_7	TCNT_7	16	H'FFFB26	TPU_7	16	2Pφ/2Pφ
Timer general register A_7	TGRA_7	16	H'FFFB28	TPU_7	16	2Pφ/2Pφ
Timer general register B_7	TGRB_7	16	H'FFFB2A	TPU_7	16	2Pφ/2Pφ
	—		H'FFFB2C to H'FFFB2F	TPU_7	16	
Timer control register_8	TCR_8	8	H'FFFB30	TPU_8	16	2Pφ/2Pφ
Timer mode register_8	TMDR_8	8	H'FFFB31	TPU_8	16	2Pφ/2Pφ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
Timer I/O control register_8	TIOR_8	8	H'FFFB32	TPU_8	16	2Pϕ/2Pϕ
	—		H'FFFB33	TPU_8	16	
Timer interrupt enable register_8	TIER_8	8	H'FFFB34	TPU_8	16	2Pϕ/2Pϕ
Timer status register_8	TSR_8	8	H'FFFB35	TPU_8	16	2Pϕ/2Pϕ
Timer counter_8	TCNT_8	16	H'FFFB36	TPU_8	16	2Pϕ/2Pϕ
Timer general register A_8	TGRA_8	16	H'FFFB38	TPU_8	16	2Pϕ/2Pϕ
Timer general register B_8	TGRB_8	16	H'FFFB3A	TPU_8	16	2Pϕ/2Pϕ
	—		H'FFFB3C to H'FFFB3F	TPU_8	16	
Timer control register_9	TCR_9	8	H'FFFB40	TPU_9	16	2Pϕ/2Pϕ
Timer mode register_9	TMDR_9	8	H'FFFB41	TPU_9	16	2Pϕ/2Pϕ
Timer I/O control register H_9	TIORH_9	8	H'FFFB42	TPU_9	16	2Pϕ/2Pϕ
Timer I/O control register L_9	TIORL_9	8	H'FFFB43	TPU_9	16	2Pϕ/2Pϕ
Timer interrupt enable register_9	TIER_9	8	H'FFFB44	TPU_9	16	2Pϕ/2Pϕ
Timer status register_9	TSR_9	8	H'FFFB45	TPU_9	16	2Pϕ/2Pϕ
Timer counter_9	TCNT_9	16	H'FFFB46	TPU_9	16	2Pϕ/2Pϕ
Timer general register A_9	TGRA_9	16	H'FFFB48	TPU_9	16	2Pϕ/2Pϕ
Timer general register B_9	TGRB_9	16	H'FFFB4A	TPU_9	16	2Pϕ/2Pϕ
Timer general register C_9	TGRC_9	16	H'FFFB4C	TPU_9	16	2Pϕ/2Pϕ
Timer general register D_9	TGRD_9	16	H'FFFB4E	TPU_9	16	2Pϕ/2Pϕ
Timer control register_10	TCR_10	8	H'FFFB50	TPU_10	16	2Pϕ/2Pϕ
Timer mode register_10	TMDR_10	8	H'FFFB51	TPU_10	16	2Pϕ/2Pϕ
Timer I/O control register_10	TIOR_10	8	H'FFFB52	TPU_10	16	2Pϕ/2Pϕ
	—		H'FFFB53	TPU_10	16	
Timer interrupt enable register_10	TIER_10	8	H'FFFB54	TPU_10	16	2Pϕ/2Pϕ
Timer status register_10	TSR_10	8	H'FFFB55	TPU_10	16	2Pϕ/2Pϕ
Timer counter_10	TCNT_10	16	H'FFFB56	TPU_10	16	2Pϕ/2Pϕ
Timer general register A_10	TGRA_10	16	H'FFFB58	TPU_10	16	2Pϕ/2Pϕ
Timer general register B_10	TGRB_10	16	H'FFFB5A	TPU_10	16	2Pϕ/2Pϕ
	—		H'FFFB5C to H'FFFB5F	TPU_10	16	

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
Timer control register_11	TCR_11	8	H'FFFB60	TPU_11	16	2Pφ/2Pφ
Timer mode register_11	TMDR_11	8	H'FFFB61	TPU_11	16	2Pφ/2Pφ
Timer I/O control register_11	TIOR_11	8	H'FFFB62	TPU_11	16	2Pφ/2Pφ
	—		H'FFFB63	TPU_11	16	
Timer interrupt enable register_11	TIER_11	8	H'FFFB64	TPU_11	16	2Pφ/2Pφ
Timer status register_11	TSR_11	8	H'FFFB65	TPU_11	16	2Pφ/2Pφ
Timer counter_11	TCNT_11	16	H'FFFB66	TPU_11	16	2Pφ/2Pφ
Timer general register A_11	TGRA_11	16	H'FFFB68	TPU_11	16	2Pφ/2Pφ
Timer general register B_11	TGRB_11	16	H'FFFB6A	TPU_11	16	2Pφ/2Pφ
	—		H'FFFB6C to H'FFFB6F	TPU_11	16	
Port 1 data direction register	P1DDR	8	H'FFFB80	I/O port	8	2Pφ/2Pφ
Port 3 data direction register	P3DDR	8	H'FFFB82	I/O port	8	2Pφ/2Pφ
Port 6 data direction register	P6DDR	8	H'FFFB85	I/O port	8	2Pφ/2Pφ
Port A data direction register	PADDR	8	H'FFFB89	I/O port	8	2Pφ/2Pφ
Port D data direction register	PDDDR	8	H'FFFB8C	I/O port	8	2Pφ/2Pφ
Port 1 input buffer control register	P1ICR	8	H'FFFB90	I/O port	8	2Pφ/2Pφ
Port 3 input buffer control register	P3ICR	8	H'FFFB92	I/O port	8	2Pφ/2Pφ
Port 4 input buffer control register	P4ICR	8	H'FFFB93	I/O port	8	2Pφ/2Pφ
Port 5 input buffer control register	P5ICR	8	H'FFFB94	I/O port	8	2Pφ/2Pφ
Port 6 input buffer control register	P6ICR	8	H'FFFB95	I/O port	8	2Pφ/2Pφ
Port A input buffer control register	PAICR	8	H'FFFB99	I/O port	8	2Pφ/2Pφ
Port D input buffer control register	PDICR	8	H'FFFB9C	I/O port	8	2Pφ/2Pφ
Port H register	PORTH	8	H'FFFBA0	I/O port	8	2Pφ/2Pφ
Port J register	PORTJ	8	H'FFFBA2	I/O port	8	2Pφ/2Pφ
Port K register	PORTK	8	H'FFFBA3	I/O port	8	2Pφ/2Pφ
Port H data register	PHDR	8	H'FFFBA4	I/O port	8	2Pφ/2Pφ
Port J data register	PJDR	8	H'FFFBA6	I/O port	8	2Pφ/2Pφ
Port K data register	PKDR	8	H'FFFBA7	I/O port	8	2Pφ/2Pφ
Port H data direction register	PHDDR	8	H'FFFBA8	I/O port	8	2Pφ/2Pφ
Port J data direction register	PJDDR	8	H'FFFBAA	I/O port	8	2Pφ/2Pφ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
Port K data direction register	PKDDR	8	H'FFFBAB	I/O port	8	2Pφ/2Pφ
Port H input buffer control register	PHICR	8	H'FFFBAC	I/O port	8	2Pφ/2Pφ
Port J input buffer control register	PJICR	8	H'FFFBAE	I/O port	8	2Pφ/2Pφ
Port K input buffer control register	PKICR	8	H'FFFBAF	I/O port	8	2Pφ/2Pφ
Port D pull-up MOS control register	PDPCR	8	H'FFFBB4	I/O port	8	2Pφ/2Pφ
Port H pull-up MOS control register	PHPCR	8	H'FFFBB8	I/O port	8	2Pφ/2Pφ
Port J pull-up MOS control register	PJPCR	8	H'FFFBBA	I/O port	8	2Pφ/2Pφ
Port K pull-up MOS control register	PKPCR	8	H'FFFBBB	I/O port	8	2Pφ/2Pφ
Port function control register 5	PFCR5	8	H'FFFBC5	I/O port	8	2Pφ/3Pφ
Port function control register 6	PFCR6	8	H'FFFBC6	I/O port	8	2Pφ/3Pφ
Port function control register 8	PFCR8	8	H'FFFBC8	I/O port	8	2Pφ/3Pφ
Port function control register 9	PFCR9	8	H'FFFBC9	I/O port	8	2Pφ/3Pφ
Port function control register A	PFCRA	8	H'FFFBCA	I/O port	8	2Pφ/3Pφ
Port function control register B	PFCRB	8	H'FFFBCB	I/O port	8	2Pφ/3Pφ
Port function control register C	PFCRC	8	H'FFFBCC	I/O port	8	2Pφ/3Pφ
Port function control register D	PFCRD	8	H'FFFBCD	I/O port	8	2Pφ/3Pφ
Software standby release IRQ enable register	SSIER	16	H'FFFBCE	INTC	16	2Iφ/3Iφ
Port 1 driving ability setting register	P1DSR	8	H'FFFBE0	I/O port	8	2Pφ/2Pφ
Port 6 driving ability setting register	P6DSR	8	H'FFFBE2	I/O port	8	2Pφ/2Pφ
Port A driving ability setting register	PADSR	8	H'FFFBE3	I/O port	8	2Pφ/2Pφ
Port D driving ability setting register	PDDSR	8	H'FFFBE4	I/O port	8	2Pφ/2Pφ
Port H driving ability setting register	PHDSR	8	H'FFFBE6	I/O port	8	2Pφ/2Pφ
Port 1 pin state setting register	P1PSR	8	H'FFFBE8	I/O port	8	2Pφ/2Pφ
Port 6 pin state setting register	P6PSR	8	H'FFFBEA	I/O port	8	2Pφ/2Pφ
Port A pin state setting register	PAPSR	8	H'FFFBEB	I/O port	8	2Pφ/2Pφ
Port D pin state setting register	PDPSR	8	H'FFFBEC	I/O port	8	2Pφ/2Pφ
Port H pin state setting register	PHPSR	8	H'FFFBEE	I/O port	8	2Pφ/2Pφ
DMA source address register_0	DSAR_0	32	H'FFFC00	DMAC_0	16	2Iφ/2Iφ
DMA destination address register_0	DDAR_0	32	H'FFFC04	DMAC_0	16	2Iφ/2Iφ
DMA offset register_0	DOFR_0	32	H'FFFC08	DMAC_0	16	2Iφ/2Iφ
DMA transfer count register_0	DTCR_0	32	H'FFFC0C	DMAC_0	16	2Iφ/2Iφ
DMA block size register_0	DBSR_0	32	H'FFFC10	DMAC_0	16	2Iφ/2Iφ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
DMA mode control register_0	DMDR_0	32	H'FFFC14	DMAC_0	16	2Iϕ/2Iϕ
DMA address control register_0	DACR_0	32	H'FFFC18	DMAC_0	16	2Iϕ/2Iϕ
DMA source address register_1	DSAR_1	32	H'FFFC20	DMAC_1	16	2Iϕ/2Iϕ
DMA destination address register_1	DDAR_1	32	H'FFFC24	DMAC_1	16	2Iϕ/2Iϕ
DMA offset register_1	DOFR_1	32	H'FFFC28	DMAC_1	16	2Iϕ/2Iϕ
DMA transfer count register_1	DTCR_1	32	H'FFFC2C	DMAC_1	16	2Iϕ/2Iϕ
DMA block size register_1	DBSR_1	32	H'FFFC30	DMAC_1	16	2Iϕ/2Iϕ
DMA mode control register_1	DMDR_1	32	H'FFFC34	DMAC_1	16	2Iϕ/2Iϕ
DMA address control register_1	DACR_1	32	H'FFFC38	DMAC_1	16	2Iϕ/2Iϕ
DMA source address register_2	DSAR_2	32	H'FFFC40	DMAC_2	16	2Iϕ/2Iϕ
DMA destination address register_2	DDAR_2	32	H'FFFC44	DMAC_2	16	2Iϕ/2Iϕ
DMA offset register_2	DOFR_2	32	H'FFFC48	DMAC_2	16	2Iϕ/2Iϕ
DMA transfer count register_2	DTCR_2	32	H'FFFC4C	DMAC_2	16	2Iϕ/2Iϕ
DMA block size register_2	DBSR_2	32	H'FFFC50	DMAC_2	16	2Iϕ/2Iϕ
DMA mode control register_2	DMDR_2	32	H'FFFC54	DMAC_2	16	2Iϕ/2Iϕ
DMA address control register_2	DACR_2	32	H'FFFC58	DMAC_2	16	2Iϕ/2Iϕ
DMA source address register_3	DSAR_3	32	H'FFFC60	DMAC_3	16	2Iϕ/2Iϕ
DMA destination address register_3	DDAR_3	32	H'FFFC64	DMAC_3	16	2Iϕ/2Iϕ
DMA offset register_3	DOFR_3	32	H'FFFC68	DMAC_3	16	2Iϕ/2Iϕ
DMA transfer count register_3	DTCR_3	32	H'FFFC6C	DMAC_3	16	2Iϕ/2Iϕ
DMA block size register_3	DBSR_3	32	H'FFFC70	DMAC_3	16	2Iϕ/2Iϕ
DMA mode control register_3	DMDR_3	32	H'FFFC74	DMAC_3	16	2Iϕ/2Iϕ
DMA address control register_3	DACR_3	32	H'FFFC78	DMAC_3	16	2Iϕ/2Iϕ
DMA module request select register_0	DMRSR_0	8	H'FFFD20	DMAC_0	16	2Iϕ/2Iϕ
DMA module request select register_1	DMRSR_1	8	H'FFFD21	DMAC_1	16	2Iϕ/2Iϕ
DMA module request select register_2	DMRSR_2	8	H'FFFD22	DMAC_2	16	2Iϕ/2Iϕ
DMA module request select register_3	DMRSR_3	8	H'FFFD23	DMAC_3	16	2Iϕ/2Iϕ
Interrupt priority register A	IPRA	16	H'FFFD40	INTC	16	2Iϕ/3Iϕ
Interrupt priority register B	IPRB	16	H'FFFD42	INTC	16	2Iϕ/3Iϕ
Interrupt priority register C	IPRC	16	H'FFFD44	INTC	16	2Iϕ/3Iϕ
Interrupt priority register D	IPRD	16	H'FFFD46	INTC	16	2Iϕ/3Iϕ
Interrupt priority register E	IPRE	16	H'FFFD48	INTC	16	2Iϕ/3Iϕ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
Interrupt priority register F	IPRF	16	H'FFFD4A	INTC	16	2Iϕ/3Iϕ
Interrupt priority register G	IPRG	16	H'FFFD4C	INTC	16	2Iϕ/3Iϕ
Interrupt priority register I	IPRI	16	H'FFFD50	INTC	16	2Iϕ/3Iϕ
Interrupt priority register J	IPRJ	16	H'FFFD52	INTC	16	2Iϕ/3Iϕ
Interrupt priority register L	IPRL	16	H'FFFD56	INTC	16	2Iϕ/3Iϕ
Interrupt priority register M	IPRM	16	H'FFFD58	INTC	16	2Iϕ/3Iϕ
Interrupt priority register N	IPRN	16	H'FFFD5A	INTC	16	2Iϕ/3Iϕ
Interrupt priority register O	IPRO	16	H'FFFD5C	INTC	16	2Iϕ/3Iϕ
Interrupt priority register P	IPRP	16	H'FFFD5E	INTC	16	2Iϕ/3Iϕ
IRQ sense control register H	ISCRH	16	H'FFFD68	INTC	16	2Iϕ/3Iϕ
IRQ sense control register L	ISCRL	16	H'FFFD6A	INTC	16	2Iϕ/3Iϕ
DTC vector base register	DTCVBR	32	H'FFFD80	BSC	16	2Iϕ/3Iϕ
Bus control register 2	BCR2	8	H'FFFD94	BSC	16	2Iϕ/3Iϕ
Mode control register	MDCR	16	H'FFFDC0	SYSTEM	16	2Iϕ/3Iϕ
System control register	SYSCR0	16	H'FFFDC2	SYSTEM	16	2Iϕ/3Iϕ
System clock control register	SCKCR0	16	H'FFFDC4	SYSTEM	16	2Iϕ/3Iϕ
Standby control register	SBYCR	16	H'FFFDC6	SYSTEM	16	2Iϕ/3Iϕ
Module stop control register A	MSTPCRA	16	H'FFFDC8	SYSTEM	16	2Iϕ/3Iϕ
Module stop control register B	MSTPCRB	16	H'FFFDCA	SYSTEM	16	2Iϕ/3Iϕ
Module stop control register C	MSTPCRC	16	H'FFFDCC	SYSTEM	16	2Iϕ/3Iϕ
Recovery oscillator control register	ROSCCR	16	H'FFFDE0	SYSTEM	16	2Iϕ/3Iϕ
System control register 1	SYSCR1	16	H'FFFDE2	SYSTEM	16	2Iϕ/3Iϕ
System clock control register 1	SCKCR1	16	H'FFFDE4	SYSTEM	16	2Iϕ/3Iϕ
Module stop control register D	MSTPCRD	16	H'FFFDE8	SYSTEM	16	2Iϕ/3Iϕ
Module stop control register E	MSTPCRE	16	H'FFFDEA	SYSTEM	16	2Iϕ/3Iϕ
Serial mode register_3	SMR_3	8	H'FFFE88	SCI_3	8	2Pϕ/2Pϕ
Bit rate register_3	BRR_3	8	H'FFFE89	SCI_3	8	2Pϕ/2Pϕ
Serial control register_3	SCR_3	8	H'FFFE8A	SCI_3	8	2Pϕ/2Pϕ
Transmit data register_3	TDR_3	8	H'FFFE8B	SCI_3	8	2Pϕ/2Pϕ
Serial status register_3	SSR_3	8	H'FFFE8C	SCI_3	8	2Pϕ/2Pϕ
Receive data register_3	RDR_3	8	H'FFFE8D	SCI_3	8	2Pϕ/2Pϕ
Serial mode register_4	SMR_4	8	H'FFFE90	SCI_4	8	2Pϕ/2Pϕ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
Bit rate register_4	BRR_4	8	H'FFFE91	SCI_4	8	2Pϕ/2Pϕ
Serial control register_4	SCR_4	8	H'FFFE92	SCI_4	8	2Pϕ/2Pϕ
Transmit data register_4	TDR_4	8	H'FFFE93	SCI_4	8	2Pϕ/2Pϕ
Serial status register_4	SSR_4	8	H'FFFE94	SCI_4	8	2Pϕ/2Pϕ
Receive data register_4	RDR_4	8	H'FFFE95	SCI_4	8	2Pϕ/2Pϕ
Timer control register_4	TCR_4	8	H'FFFEE0	TPU_4	16	2Pϕ/2Pϕ
Timer mode register_4	TMDR_4	8	H'FFFEE1	TPU_4	16	2Pϕ/2Pϕ
Timer I/O control register_4	TIOR_4	8	H'FFFEE2	TPU_4	16	2Pϕ/2Pϕ
	—		H'FFFEE3	TPU_4	16	
Timer interrupt enable register_4	TIER_4	8	H'FFFEE4	TPU_4	16	2Pϕ/2Pϕ
Timer status register_4	TSR_4	8	H'FFFEE5	TPU_4	16	2Pϕ/2Pϕ
Timer counter_4	TCNT_4	16	H'FFFEE6	TPU_4	16	2Pϕ/2Pϕ
Timer general register A_4	TGRA_4	16	H'FFFEE8	TPU_4	16	2Pϕ/2Pϕ
Timer general register B_4	TGRB_4	16	H'FFFEEA	TPU_4	16	2Pϕ/2Pϕ
	—		H'FFFEEC to H'FFFEEF	TPU_4	16	
Timer control register_5	TCR_5	8	H'FFFEF0	TPU_5	16	2Pϕ/2Pϕ
Timer mode register_5	TMDR_5	8	H'FFFEF1	TPU_5	16	2Pϕ/2Pϕ
Timer I/O control register_5	TIOR_5	8	H'FFFEF2	TPU_5	16	2Pϕ/2Pϕ
	—		H'FFFEF3	TPU_5	16	
Timer interrupt enable register_5	TIER_5	8	H'FFFEF4	TPU_5	16	2Pϕ/2Pϕ
Timer status register_5	TSR_5	8	H'FFFEF5	TPU_5	16	2Pϕ/2Pϕ
Timer counter_5	TCNT_5	16	H'FFFEF6	TPU_5	16	2Pϕ/2Pϕ
Timer general register A_5	TGRA_5	16	H'FFFEF8	TPU_5	16	2Pϕ/2Pϕ
Timer general register B_5	TGRB_5	16	H'FFFEFA	TPU_5	16	2Pϕ/2Pϕ
	—		H'FFFEFC to H'FFFEFF	TPU_5	16	
DTC enable register A	DTCERA	16	H'FFFF20	DTC	16	2Iϕ/3Iϕ
DTC enable register B	DTCERB	16	H'FFFF22	DTC	16	2Iϕ/3Iϕ
DTC enable register C	DTCERC	16	H'FFFF24	DTC	16	2Iϕ/3Iϕ
DTC enable register D	DTCERD	16	H'FFFF26	DTC	16	2Iϕ/3Iϕ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
DTC enable register E	DTCERE	16	H'FFFF28	DTC	16	2Iϕ/3Iϕ
DTC enable register F	DTCERF	16	H'FFFF2A	DTC	16	2Iϕ/3Iϕ
DTC enable register G	DTCERG	16	H'FFFF2C	DTC	16	2Iϕ/3Iϕ
DTC control register	DTCCR	8	H'FFFF30	DTC	16	2Iϕ/3Iϕ
Interrupt control register	INTCR	8	H'FFFF32	INTC	16	2Iϕ/3Iϕ
CPU priority control register	CPUPCR	8	H'FFFF33	INTC	16	2Iϕ/3Iϕ
IRQ enable register	IER	16	H'FFFF34	INTC	16	2Iϕ/3Iϕ
IRQ status register	ISR	16	H'FFFF36	INTC	16	2Iϕ/3Iϕ
Port 1 register	PORT1	8	H'FFFF40	I/O port	8	2Pϕ/-
Port 3 register	PORT3	8	H'FFFF42	I/O port	8	2Pϕ/-
Port 4 register	PORT4	8	H'FFFF43	I/O port	8	2Pϕ/-
Port 5 register	PORT5	8	H'FFFF44	I/O port	8	2Pϕ/-
Port 6 register	PORT6	8	H'FFFF45	I/O port	8	2Pϕ/-
Port A register	PORTA	8	H'FFFF49	I/O port	8	2Pϕ/-
Port D register	PORTD	8	H'FFFF4C	I/O port	8	2Pϕ/-
Port 1 data register	P1DR	8	H'FFFF50	I/O port	8	2Pϕ/2Pϕ
Port 3 data register	P3DR	8	H'FFFF52	I/O port	8	2Pϕ/2Pϕ
Port 6 data register	P6DR	8	H'FFFF55	I/O port	8	2Pϕ/2Pϕ
Port A data register	PADR	8	H'FFFF59	I/O port	8	2Pϕ/2Pϕ
Port D data register	PDDR	8	H'FFFF5C	I/O port	8	2Pϕ/2Pϕ
PPG output control register	PCR	8	H'FFFF76	PPG	8	2Pϕ/2Pϕ
PPG output mode register	PMR	8	H'FFFF77	PPG	8	2Pϕ/2Pϕ
Next data enable register H	NDERH	8	H'FFFF78	PPG	8	2Pϕ/2Pϕ
Next data enable register L	NDERL	8	H'FFFF79	PPG	8	2Pϕ/2Pϕ
Output data register H	PODRH	8	H'FFFF7A	PPG	8	2Pϕ/2Pϕ
Output data register L	PODRL	8	H'FFFF7B	PPG	8	2Pϕ/2Pϕ
Next data register H*3	NDRH	8	H'FFFF7C	PPG	8	2Pϕ/2Pϕ
Next data register L*3	NDRL	8	H'FFFF7D	PPG	8	2Pϕ/2Pϕ
Next data register H*3	NDRH	8	H'FFFF7E	PPG	8	2Pϕ/2Pϕ
Next data register L*3	NDRL	8	H'FFFF7F	PPG	8	2Pϕ/2Pϕ
Timer control/status register	TCSR	8	H'FFFFA4	WDT	16	2Pϕ/3Pϕ
Timer counter	TCNT	8	H'FFFFA5	WDT	16	2Pϕ/3Pϕ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
Reset control/status register	RSTCSR	8	H'FFFFA7	WDT	16	2Pϕ/3Pϕ
Timer start register	TSTR	8	H'FFFFBC	TPU (unit 0)	16	2Pϕ/2Pϕ
Timer synchronous register	TSYR	8	H'FFFFBD	TPU (unit 0)	16	2Pϕ/2Pϕ
Timer control register_0	TCR_0	8	H'FFFFC0	TPU_0	16	2Pϕ/2Pϕ
Timer mode register_0	TMDR_0	8	H'FFFFC1	TPU_0	16	2Pϕ/2Pϕ
Timer I/O control register H_0	TIORH_0	8	H'FFFFC2	TPU_0	16	2Pϕ/2Pϕ
Timer I/O control register L_0	TIORL_0	8	H'FFFFC3	TPU_0	16	2Pϕ/2Pϕ
Timer interrupt enable register_0	TIER_0	8	H'FFFFC4	TPU_0	16	2Pϕ/2Pϕ
Timer status register_0	TSR_0	8	H'FFFFC5	TPU_0	16	2Pϕ/2Pϕ
Timer counter_0	TCNT_0	16	H'FFFFC6	TPU_0	16	2Pϕ/2Pϕ
Timer general register A_0	TGRA_0	16	H'FFFFC8	TPU_0	16	2Pϕ/2Pϕ
Timer general register B_0	TGRB_0	16	H'FFFFCA	TPU_0	16	2Pϕ/2Pϕ
Timer general register C_0	TGRC_0	16	H'FFFFCC	TPU_0	16	2Pϕ/2Pϕ
Timer general register D_0	TGRD_0	16	H'FFFFCE	TPU_0	16	2Pϕ/2Pϕ
Timer control register_1	TCR_1	8	H'FFFFD0	TPU_1	16	2Pϕ/2Pϕ
Timer mode register_1	TMDR_1	8	H'FFFFD1	TPU_1	16	2Pϕ/2Pϕ
Timer I/O control register_1	TIOR_1	8	H'FFFFD2	TPU_1	16	2Pϕ/2Pϕ
	—		H'FFFFD3	TPU_1	16	
Timer interrupt enable register_1	TIER_1	8	H'FFFFD4	TPU_1	16	2Pϕ/2Pϕ
Timer status register_1	TSR_1	8	H'FFFFD5	TPU_1	16	2Pϕ/2Pϕ
Timer counter_1	TCNT_1	16	H'FFFFD6	TPU_1	16	2Pϕ/2Pϕ
Timer general register A_1	TGRA_1	16	H'FFFFD8	TPU_1	16	2Pϕ/2Pϕ
Timer general register B_1	TGRB_1	16	H'FFFFDA	TPU_1	16	2Pϕ/2Pϕ
	—		H'FFFFDC to H'FFFFDF	TPU_1	16	
Timer control register_2	TCR_2	8	H'FFFFE0	TPU_2	16	2Pϕ/2Pϕ
Timer mode register_2	TMDR_2	8	H'FFFFE1	TPU_2	16	2Pϕ/2Pϕ
Timer I/O control register_2	TIOR_2	8	H'FFFFE2	TPU_2	16	2Pϕ/2Pϕ
	—		H'FFFFE3	TPU_2	16	
Timer interrupt enable register_2	TIER_2	8	H'FFFFE4	TPU_2	16	2Pϕ/2Pϕ
Timer status register_2	TSR_2	8	H'FFFFE5	TPU_2	16	2Pϕ/2Pϕ

Register Name	Abbreviation	Number of Bits	Address*1	Module	Data Width	Access Cycles (Read/Write)*2
Timer counter_2	TCNT_2	16	H'FFFFE6	TPU_2	16	2Pϕ/2Pϕ
Timer general register A_2	TGRA_2	16	H'FFFFE8	TPU_2	16	2Pϕ/2Pϕ
Timer general register B_2	TGRB_2	16	H'FFFFEA	TPU_2	16	2Pϕ/2Pϕ
	—		H'FFFFEC to H'FFFFEF	TPU_2	16	
Timer control register_3	TCR_3	8	H'FFFFF0	TPU_3	16	2Pϕ/2Pϕ
Timer mode register_3	TMDR_3	8	H'FFFFF1	TPU_3	16	2Pϕ/2Pϕ
Timer I/O control register H_3	TIORH_3	8	H'FFFFF2	TPU_3	16	2Pϕ/2Pϕ
Timer I/O control register L_3	TIORL_3	8	H'FFFFF3	TPU_3	16	2Pϕ/2Pϕ
Timer interrupt enable register_3	TIER_3	8	H'FFFFF4	TPU_3	16	2Pϕ/2Pϕ
Timer status register_3	TSR_3	8	H'FFFFF5	TPU_3	16	2Pϕ/2Pϕ
Timer counter_3	TCNT_3	16	H'FFFFF6	TPU_3	16	2Pϕ/2Pϕ
Timer general register A_3	TGRA_3	16	H'FFFFF8	TPU_3	16	2Pϕ/2Pϕ
Timer general register B_3	TGRB_3	16	H'FFFFFA	TPU_3	16	2Pϕ/2Pϕ
Timer general register C_3	TGRC_3	16	H'FFFFFC	TPU_3	16	2Pϕ/2Pϕ
Timer general register D_3	TGRD_3	16	H'FFFFFE	TPU_3	16	2Pϕ/2Pϕ

Notes: 1. The lower 24 bits of the address are indicated.

2. In accesses to the mailbox areas of RCAN-TL1_0 and RCAN-TL1_1, waits of 0 to 5 Pϕ may occur.

3. When the same output trigger is specified for pulse output groups 2 and 3 by the PCR setting, the NDRH address is H'FFF7C. When different output triggers are specified, the NDRH addresses for pulse output groups 2 and 3 are H'FFF7E and H'FFF7C, respectively. Similarly, when the same output trigger is specified for pulse output groups 0 and 1 by the PCR setting, the NDRL address is H'FFF7D. When different output triggers are specified, the NDRL addresses for pulse output groups 0 and 1 are H'FFF7F and H'FFF7D, respectively.

24.2 Register Bits

Register addresses and bit names of the on-chip peripheral modules are described below.

Each line covers eight bits, and 16-bit and 32-bit registers are shown as 2 or 4 lines, respectively.

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
MCR_0	MCR15	MCR14	—	—	—	TST2	TST1	TST0	RCAN-TL1_0
	MCR7	MCR6	MCR5	—	—	MCR2	MCR1	MCR0	
GSR_0	—	—	—	—	—	—	—	—	
	—	—	GSR5	GSR4	GSR3	GSR2	GSR1	GSR0	
BCR1_0	TSG13	TSG12	TSG11	TSG10	—	TSG22	TSG21	TSG20	
	—	—	SJW1	SJW0	—	—	—	BSP	
BCR0_0	—	—	—	—	—	—	—	—	
	BRP7	BRP6	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	
IRR_0	IRR15	IRR14	IRR13	IRR12	IRR11	IRR10	IRR9	IRR8	
	IRR7	IRR6	IRR5	IRR4	IRR3	IRR2	IRR1	IRR0	
IMR_0	IMR15	IMR14	IMR13	IMR12	IMR11	IMR10	IMR9	IMR8	
	IMR7	IMR6	IMR5	IMR4	IMR3	IMR2	IMR1	IMR0	
TEC_0	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	
REC_0	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	
TXPR1_0	TXPR1_15	TXPR1_14	TXPR1_13	TXPR1_12	TXPR1_11	TXPR1_10	TXPR1_9	TXPR1_8	
	TXPR1_7	TXPR1_6	TXPR1_5	TXPR1_4	TXPR1_3	TXPR1_2	TXPR1_1	TXPR1_0	
TXPR0_0	TXPR0_15	TXPR0_14	TXPR0_13	TXPR0_12	TXPR0_11	TXPR0_10	TXPR0_9	TXPR0_8	
	TXPR0_7	TXPR0_6	TXPR0_5	TXPR0_4	TXPR0_3	TXPR0_2	TXPR0_1	—	
TXCR1_0	TXCR1_15	TXCR1_14	TXCR1_13	TXCR1_12	TXCR1_11	TXCR1_10	TXCR1_9	TXCR1_8	
	TXCR1_7	TXCR1_6	TXCR1_5	TXCR1_4	TXCR1_3	TXCR1_2	TXCR1_1	TXCR1_0	
TXCR0_0	TXCR0_15	TXCR0_14	TXCR0_13	TXCR0_12	TXCR0_11	TXCR0_10	TXCR0_9	TXCR0_8	
	TXCR0_7	TXCR0_6	TXCR0_5	TXCR0_4	TXCR0_3	TXCR0_2	TXCR0_1	—	
TXACK1_0	TXACK1_15	TXACK1_14	TXACK1_13	TXACK1_12	TXACK1_11	TXACK1_10	TXACK1_9	TXACK1_8	
	TXACK1_7	TXACK1_6	TXACK1_5	TXACK1_4	TXACK1_3	TXACK1_2	TXACK1_1	TXACK1_0	
TXACK0_0	TXACK0_15	TXACK0_14	TXACK0_13	TXACK0_12	TXACK0_11	TXACK0_10	TXACK0_9	TXACK0_8	
	TXACK0_7	TXACK0_6	TXACK0_5	TXACK0_4	TXACK0_3	TXACK0_2	TXACK0_1	—	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
ABACK1_0	ABACK1_15	ABACK1_14	ABACK1_13	ABACK1_12	ABACK1_11	ABACK1_10	ABACK1_9	ABACK1_8	RCAN-TL1_0
	ABACK1_7	ABACK1_6	ABACK1_5	ABACK1_4	ABACK1_3	ABACK1_2	ABACK1_1	ABACK1_0	
ABACK0_0	ABACK0_15	ABACK0_14	ABACK0_13	ABACK0_12	ABACK0_11	ABACK0_10	ABACK0_9	ABACK0_8	
	ABACK0_7	ABACK0_6	ABACK0_5	ABACK0_4	ABACK0_3	ABACK0_2	ABACK0_1	—	
RXPR1_0	RXPR1_15	RXPR1_14	RXPR1_13	RXPR1_12	RXPR1_11	RXPR1_10	RXPR1_9	RXPR1_8	
	RXPR1_7	RXPR1_6	RXPR_15	RXPR1_4	RXPR1_3	RXPR1_2	RXPR1_1	RXPR1_0	
RXPR0_0	RXPR0_15	RXPR0_14	RXPR0_13	RXPR0_12	RXPR0_11	RXPR0_10	RXPR0_9	RXPR0_8	
	RXPR0_7	RXPR0_6	RXPR0_5	RXPR0_4	RXPR0_3	RXPR0_2	RXPR0_1	RXPR0_0	
RFPR1_0	RFPR1_15	RFPR1_14	RFPR1_13	RFPR1_12	RFPR1_11	RFPR1_10	RFPR1_9	RFPR1_8	
	RFPR1_7	RFPR1_6	RFPR1_5	RFPR1_4	RFPR1_3	RFPR1_2	RFPR1_1	RFPR1_0	
RFPR0_0	RFPR0_15	RFPR0_14	RFPR0_13	RFPR0_12	RFPR0_11	RFPR0_10	RFPR0_9	RFPR0_8	
	RFPR0_7	RFPR0_6	RFPR0_5	RFPR0_4	RFPR0_3	RFPR0_2	RFPR0_1	RFPR0_0	
MBIMR1_0	MBIMR1_15	MBIMR1_14	MBIMR1_13	MBIMR1_12	MBIMR1_11	MBIMR1_10	MBIMR1_9	MBIMR1_8	
	MBIMR1_7	MBIMR1_6	MBIMR1_5	MBIMR1_4	MBIMR1_3	MBIMR1_2	MBIMR1_1	MBIMR1_0	
MBIMR0_0	MBIMR0_15	MBIMR0_14	MBIMR0_13	MBIMR0_12	MBIMR0_11	MBIMR0_10	MBIMR0_9	MBIMR0_8	
	MBIMR0_7	MBIMR0_6	MBIMR0_5	MBIMR0_4	MBIMR0_3	MBIMR0_2	MBIMR0_1	MBIMR0_0	
UMSR1_0	UMSR1_15	UMSR1_14	UMSR1_13	UMSR1_12	UMSR1_11	UMSR1_10	UMSR1_9	UMSR1_8	
	UMSR1_7	UMSR1_6	UMSR1_5	UMSR1_4	UMSR1_3	UMSR1_2	UMSR1_1	UMSR1_0	
UMSR0_0	UMSR0_15	UMSR0_14	UMSR0_13	UMSR0_12	UMSR0_11	UMSR0_10	UMSR0_9	UMSR0_8	
	UMSR0_7	UMSR0_6	UMSR0_5	UMSR0_4	UMSR0_3	UMSR0_2	UMSR0_1	UMSR0_0	
TTCR0_0	TCR15	TCR14	TCR13	TCR12	TCR11	TCR10	—	—	
	—	TCR6	TPSC5	TPSC4	TPSC3	TPSC2	TPSC1	TPSC0	
CMAX_ TEW_0	—	—	—	—	—	CMAX2	CMAX1	CMAX0	
	—	—	—	—	TEW3	TEW2	TEW1	TEW0	
RFTROFF_0	RFTROFF7	RFTROFF6	RFTROFF5	RFTROFF4	RFTROFF3	RFTROFF2	RFTROFF1	RFTROFF0	
	—	—	—	—	—	—	—	—	
TSR_0	—	—	—	—	—	—	—	—	
	—	—	—	TSR4	TSR3	TSR2	TSR1	TSR0	
CCR_0	—	—	—	—	—	—	—	—	
	—	—	CCR5	CCR4	CCR3	CCR2	CCR1	CCR0	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
TCNTR_0	TCNTR15	TCNTR14	TCNTR13	TCNTR12	TCNTR11	TCNTR10	TCNTR9	TCNTR8	RCAN-TL1_0
	TCNTR7	TCNTR6	TCNTR5	TCNTR4	TCNTR3	TCNTR2	TCNTR1	TCNTR0	
CYCTR_0	CYCTR15	CYCTR14	CYCTR13	CYCTR12	CYCTR11	CYCTR10	CYCTR9	CYCTR8	
	CYCTR7	CYCTR6	CYCTR5	CYCTR4	CYCTR3	CYCTR2	CYCTR1	CYCTR0	
RFMK_0	RFMK15	RFMK14	RFMK13	RFMK12	RFMK11	RFMK10	RFMK9	RFMK8	
	RFMK7	RFMK6	RFMK5	RFMK4	RFMK3	RFMK2	RFMK1	RFMK0	
TCMR0_0	TCMR0_15	TCMR0_14	TCMR0_13	TCMR0_12	TCMR0_11	TCMR0_10	TCMR0_9	TCMR0_8	
	TCMR0_7	TCMR0_6	TCMR0_5	TCMR0_4	TCMR0_3	TCMR0_2	TCMR0_1	TCMR0_0	
TCMR1_0	TCMR1_15	TCMR1_14	TCMR1_13	TCMR1_12	TCMR1_11	TCMR1_10	TCMR1_9	TCMR1_8	
	TCMR1_7	TCMR1_6	TCMR1_5	TCMR1_4	TCMR1_3	TCMR1_2	TCMR1_1	TCMR1_0	
TCMR2_0	TCMR2_15	TCMR2_14	TCMR2_13	TCMR2_12	TCMR2_11	TCMR2_10	TCMR2_9	TCMR2_8	
	TCMR2_7	TCMR2_6	TCMR2_5	TCMR2_4	TCMR2_3	TCMR2_2	TCMR2_1	TCMR2_0	
TTTSEL_0	TTTSEL15	TTTSEL14	TTTSEL13	TTTSEL12	TTTSEL11	TTTSEL10	TTTSEL9	TTTSEL8	
	TTTSEL7	TTTSEL6	TTTSEL5	TTTSEL4	TTTSEL3	TTTSEL2	TTTSEL1	TTTSEL0	
MB_0[0]. CONTROL0H	IDE	RTR	—	STDID10	STDID9	STDID8	STDID7	STDID6	
	STDID5	STDID4	STDID3	STDID2	STDID1	STDID0	EXTID17	EXTID16	
MB_0[0]. CONTROL0L	EXTID15	EXTID14	EXTID13	EXTID12	EXTID11	EXTID10	EXTID9	EXTID8	
	EXTID7	EXTID6	EXTID5	EXTID4	EXTID3	EXTID2	EXTID1	EXTID0	
MB_0[0]. LAFMH	IDE_LAFM	—	—	STDID_ LAFM10	STDID_ LAFM9	STDID_ LAFM8	STDID_ LAFM7	STDID_ LAFM6	
	STDID_ LAFM5	STDID_ LAFM4	STDID_ LAFM3	STDID_ LAFM2	STDID_ LAFM1	STDID_ LAFM0	EXTID_ LAFM17	EXTID_ LAFM16	
MB_0[0]. LAFML	EXTID_ LAFM15	EXTID_ LAFM14	EXTID_ LAFM13	EXTID_ LAFM12	EXTID_ LAFM11	EXTID_ LAFM10	EXTID_ LAFM9	EXTID_ LAFM8	
	EXTID_ LAFM7	EXTID_ LAFM6	EXTID_ LAFM5	EXTID_ LAFM4	EXTID_ LAFM3	EXTID_ LAFM2	EXTID_ LAFM1	EXTID_ LAFM0	
MB_0[0]. MSG_DATA[0]	MSG_DATA0								
MB_0[0]. MSG_DATA[1]	MSG_DATA1								
MB_0[0]. MSG_DATA[2]	MSG_DATA2								

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
MB_0[0]. MSG_DATA[3]	MSG_DATA3								RCAN-TL1_0
MB_0[0]. MSG_DATA[4]	MSG_DATA4								
MB_0[0]. MSG_DATA[5]	MSG_DATA5								
MB_0[0]. MSG_DATA[6]	MSG_DATA6								
MB_0[0]. MSG_DATA[7]	MSG_DATA7								
MB_0[0]. CONTROL1H	—	—	NMC	—	—	MBC2	MBC1	MBC0	
MB_0[0]. CONTROL1L	—	—	—	—	DLC3	DLC2	DLC1	DLC0	
MB_0[0]. TIMESTAMP	TS15	TS14	TS13	TS12	TS11	TS10	TS9	TS8	
	TS7	TS6	TS5	TS4	TS3	TS2	TS1	TS0	
MB_0[1]. CONTROL0H	IDE	RTR	—	STDID10	STDID9	STDID8	STDID7	STDID6	
	STDID5	STDID4	STDID3	STDID2	STDID1	STDID0	EXTID17	EXTID16	
MB_0[1]. CONTROL0L	EXTID15	EXTID14	EXTID13	EXTID12	EXTID11	EXTID10	EXTID9	EXTID8	
	EXTID7	EXTID6	EXTID5	EXTID4	EXTID3	EXTID2	EXTID1	EXTID0	
MB_0[1]. LAFMH	IDE_LAFM	—	—	STDID_ LAFM10	STDID_ LAFM9	STDID_ LAFM8	STDID_ LAFM7	STDID_ LAFM6	
	STDID_ LAFM5	STDID_ LAFM4	STDID_ LAFM3	STDID_ LAFM2	STDID_ LAFM1	STDID_ LAFM0	EXTID_ LAFM17	EXTID_ LAFM16	
MB_0[1]. LAFML	EXTID_ LAFM15	EXTID_ LAFM14	EXTID_ LAFM13	EXTID_ LAFM12	EXTID_ LAFM11	EXTID_ LAFM10	EXTID_ LAFM9	EXTID_ LAFM8	
	EXTID_ LAFM7	EXTID_ LAFM6	EXTID_ LAFM5	EXTID_ LAFM4	EXTID_ LAFM3	EXTID_ LAFM2	EXTID_ LAFM1	EXTID_ LAFM0	
MB_0[1]. MSG_DATA[0]	MSG_DATA0								
MB_0[1]. MSG_DATA[1]	MSG_DATA1								

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
MB_0[1]. MSG_DATA[2]					MSG_DATA2				RCAN-TL1_0
MB_0[1]. MSG_DATA[3]					MSG_DATA3				
MB_0[1]. MSG_DATA[4]					MSG_DATA4				
MB_0[1]. MSG_DATA[5]					MSG_DATA5				
MB_0[1]. MSG_DATA[6]					MSG_DATA6				
MB_0[1]. MSG_DATA[7]					MSG_DATA7				
MB_0[1]. CONTROL1H	—	—	NMC	ATX	DART	MBC2	MBC1	MBC0	
MB_0[1]. CONTROL1L	—	—	—	—	DLC3	DLC2	DLC1	DLC0	
MB_0[1]. TIMESTAMP	TS15	TS14	TS13	TS12	TS11	TS10	TS9	TS8	
	TS7	TS6	TS5	TS4	TS3	TS2	TS1	TS0	
MB_0[2]. CONTROL0H to MB_0[2]. TIMESTAMP	Same as bit configuration of MB_0[1].CONTROL0H to MB_0[1].TIMESTAMP								
↓	(Repeated)								
MB_0[15]. CONTROL0H to MB_0[15]. TIMESTAMP	Same as bit configuration of MB_0[1].CONTROL0H to MB_0[1].TIMESTAMP								
MB_0[16]. CONTROL0H	IDE	RTR	—	STDID10	STDID9	STDID8	STDID7	STDID6	
	STDID5	STDID4	STDID3	STDID2	STDID1	STDID0	EXTID17	EXTID16	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
MB_0[16]. CONTROL0L	EXTID15	EXTID14	EXTID13	EXTID12	EXTID11	EXTID10	EXTID9	EXTID8	RCAN-TL1_0
	EXTID7	EXTID6	EXTID5	EXTID4	EXTID3	EXTID2	EXTID1	EXTID0	
MB_0[16]. LAFMH	IDE_LAFM	—	—	STDID_ LAFM10	STDID_ LAFM9	STDID_ LAFM8	STDID_ LAFM7	STDID_ LAFM6	
	STDID_ LAFM5	STDID_ LAFM4	STDID_ LAFM3	STDID_ LAFM2	STDID_ LAFM1	STDID_ LAFM0	EXTID_ LAFM17	EXTID_ LAFM16	
MB_0[16]. LAFML	EXTID_ LAFM15	EXTID_ LAFM14	EXTID_ LAFM13	EXTID_ LAFM12	EXTID_ LAFM11	EXTID_ LAFM10	EXTID_ LAFM9	EXTID_ LAFM8	
	EXTID_ LAFM7	EXTID_ LAFM6	EXTID_ LAFM5	EXTID_ LAFM4	EXTID_ LAFM3	EXTID_ LAFM2	EXTID_ LAFM1	EXTID_ LAFM0	
MB_0[16]. MSG_DATA[0]	MSG_DATA0								
MB_0[16]. MSG_DATA[1]	MSG_DATA1								
MB_0[16]. MSG_DATA[2]	MSG_DATA2								
MB_0[16]. MSG_DATA[3]	MSG_DATA3								
MB_0[16]. MSG_DATA[4]	MSG_DATA4								
MB_0[16]. MSG_DATA[5]	MSG_DATA5								
MB_0[16]. MSG_DATA[6]	MSG_DATA6								
MB_0[16]. MSG_DATA[7]	MSG_DATA7								
MB_0[16]. CONTROL1H	—	—	NMC	ATX	DART	MBC2	MBC1	MBC0	
MB_0[16]. CONTROL1L	—	—	—	—	DLC3	DLC2	DLC1	DLC0	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
MB_0[17]. CONTROL0H to MB_0[17]. CONTROL1L	Same as bit configuration of MB_0[16].CONTROL0H to MB_0[16].CONTROL1L								RCAN-TL1_0
↓	(Repeated)								
MB_0[23]. CONTROL0H to MB_0[23]. CONTROL1L	Same as bit configuration of MB_0[16].CONTROL0H to MB_0[16].CONTROL1L								
MB_0[24]. CONTROL0H	IDE	RTR	—	STDID10	STDID9	STDID8	STDID7	STDID6	
	STDID5	STDID4	STDID3	STDID2	STDID1	STDID0	EXTID17	EXTID16	
MB_0[24]. CONTROL0L	EXTID15	EXTID14	EXTID13	EXTID12	EXTID11	EXTID10	EXTID9	EXTID8	
	EXTID7	EXTID6	EXTID5	EXTID4	EXTID3	EXTID2	EXTID1	EXTID0	
MB_0[24]. LAFMH	IDE_LAFM	—	—	STDID_ LAFM10	STDID_ LAFM9	STDID_ LAFM8	STDID_ LAFM7	STDID_ LAFM6	
	STDID_ LAFM5	STDID_ LAFM4	STDID_ LAFM3	STDID_ LAFM2	STDID_ LAFM1	STDID_ LAFM0	EXTID_ LAFM17	EXTID_ LAFM16	
MB_0[24]. LAFML	EXTID_ LAFM15	EXTID_ LAFM14	EXTID_ LAFM13	EXTID_ LAFM12	EXTID_ LAFM11	EXTID_ LAFM10	EXTID_ LAFM9	EXTID_ LAFM8	
	EXTID_ LAFM7	EXTID_ LAFM6	EXTID_ LAFM5	EXTID_ LAFM4	EXTID_ LAFM3	EXTID_ LAFM2	EXTID_ LAFM1	EXTID_ LAFM0	
MB_0[24]. MSG_DATA[0]	MSG_DATA0								
MB_0[24]. MSG_DATA[1]	MSG_DATA1								
MB_0[24]. MSG_DATA[2]	MSG_DATA2								
MB_0[24]. MSG_DATA[3]	MSG_DATA3								
MB_0[24]. MSG_DATA[4]	MSG_DATA4								

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
MB_0[24]. MSG_DATA[5]	MSG_DATA5								RCAN-TL1_0
MB_0[24]. MSG_DATA[6]	MSG_DATA6								
MB_0[24]. MSG_DATA[7]	MSG_DATA7								
MB_0[24]. CONTROL1H	—	—	NMC	ATX	DART	MBC2	MBC1	MBC0	
MB_0[24]. CONTROL1L	—	—	—	—	DLC3	DLC2	DLC1	DLC0	
MB_0[24]. TTT	TTT15	TTT14	TTT13	TTT12	TTT11	TTT10	TTT9	TTT8	
	TTT7	TTT6	TTT5	TTT4	TTT3	TTT2	TTT1	TTT0	
MB_0[24]. TT_CONTROL	TTW1	TTW0	OFFSET5	OFFSET4	OFFSET3	OFFSET2	OFFSET1	OFFSET0	
	—	—	—	—	—	REP_ FACTOR2	REP_ FACTOR1	REP_ FACTOR0	
MB_0[25]. CONTROL0H to MB_0[25]. TT_CONTROL	Same as bit configuration of MB_0[24].CONTROL0H to MB_0[24].TT_CONTROL								
↓	(Repeated)								
MB_0[29]. CONTROL0H to MB_0[29]. TT_CONTROL	Same as bit configuration of MB_0[24].CONTROL0H to MB_0[24].TT_CONTROL								
MB_0[30]. CONTROL0H	IDE	RTR	—	STDID10	STDID9	STDID8	STDID7	STDID6	
	STDID5	STDID4	STDID3	STDID2	STDID1	STDID0	EXTID17	EXTID16	
MB_0[30]. CONTROL0L	EXTID15	EXTID14	EXTID13	EXTID12	EXTID11	EXTID10	EXTID9	EXTID8	
	EXTID7	EXTID6	EXTID5	EXTID4	EXTID3	EXTID2	EXTID1	EXTID0	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
MB_0[30]. LAFMH	IDE_LAFM	—	—	STDID_ LAFM10	STDID_ LAFM9	STDID_ LAFM8	STDID_ LAFM7	STDID_ LAFM6	RCAN-TL1_0
	STDID_ LAFM5	STDID_ LAFM4	STDID_ LAFM3	STDID_ LAFM2	STDID_ LAFM1	STDID_ LAFM0	EXTID_ LAFM17	EXTID_ LAFM16	
MB_0[30]. LAFML	EXTID_ LAFM15	EXTID_ LAFM14	EXTID_ LAFM13	EXTID_ LAFM12	EXTID_ LAFM11	EXTID_ LAFM10	EXTID_ LAFM9	EXTID_ LAFM8	
	EXTID_ LAFM7	EXTID_ LAFM6	EXTID_ LAFM5	EXTID_ LAFM4	EXTID_ LAFM3	EXTID_ LAFM2	EXTID_ LAFM1	EXTID_ LAFM0	
MB_0[30]. MSG_DATA[0]	MSG_DATA0								
MB_0[30]. MSG_DATA[1]	MSG_DATA1								
MB_0[30]. MSG_DATA[2]	MSG_DATA2								
MB_0[30]. MSG_DATA[3]	MSG_DATA3								
MB_0[30]. MSG_DATA[4]	MSG_DATA4								
MB_0[30]. MSG_DATA[5]	MSG_DATA5								
MB_0[30]. MSG_DATA[6]	MSG_DATA6								
MB_0[30]. MSG_DATA[7]	MSG_DATA7								
MB_0[30]. CONTROL1H	—	—	NMC	ATX	DART	MBC2	MBC1	MBC0	
MB_0[30]. CONTROL1L	—	—	—	—	DLC3	DLC2	DLC1	DLC0	
MB_0[30]. TIMESTAMP	TS15	TS14	TS13	TS12	TS11	TS10	TS9	TS8	
	TS7	TS6	TS5	TS4	TS3	TS2	TS1	TS0	
MB_0[30]. TTT	TTT15	TTT14	TTT13	TTT12	TTT11	TTT10	TTT9	TTT8	
	TTT7	TTT6	TTT5	TTT4	TTT3	TTT2	TTT1	TTT0	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
MB_0[31]. CONTROL0H	IDE	RTR	—	STDID10	STDID9	STDID8	STDID7	STDID6	RCAN-TL1_0
	STDID5	STDID4	STDID3	STDID2	STDID1	STDID0	EXTID17	EXTID16	
MB_0[31]. CONTROL0L	EXTID15	EXTID14	EXTID13	EXTID12	EXTID11	EXTID10	EXTID9	EXTID8	
	EXTID7	EXTID6	EXTID5	EXTID4	EXTID3	EXTID2	EXTID1	EXTID0	
MB_0[31]. LAFMH	IDE_LAFM	—	—	STDID_ LAFM10	STDID_ LAFM9	STDID_ LAFM8	STDID_ LAFM7	STDID_ LAFM6	
	STDID_ LAFM5	STDID_ LAFM4	STDID_ LAFM3	STDID_ LAFM2	STDID_ LAFM1	STDID_ LAFM0	EXTID_ LAFM17	EXTID_ LAFM16	
MB_0[31]. LAFML	EXTID_ LAFM15	EXTID_ LAFM14	EXTID_ LAFM13	EXTID_ LAFM12	EXTID_ LAFM11	EXTID_ LAFM10	EXTID_ LAFM9	EXTID_ LAFM8	
	EXTID_ LAFM7	EXTID_ LAFM6	EXTID_ LAFM5	EXTID_ LAFM4	EXTID_ LAFM3	EXTID_ LAFM2	EXTID_ LAFM1	EXTID_ LAFM0	
MB_0[31]. MSG_DATA[0]	MSG_DATA0								
MB_0[31]. MSG_DATA[1]	MSG_DATA1								
MB_0[31]. MSG_DATA[2]	MSG_DATA2								
MB_0[31]. MSG_DATA[3]	MSG_DATA3								
MB_0[31]. MSG_DATA[4]	MSG_DATA4								
MB_0[31]. MSG_DATA[5]	MSG_DATA5								
MB_0[31]. MSG_DATA[6]	MSG_DATA6								
MB_0[31]. MSG_DATA[7]	MSG_DATA7								
MB_0[31]. CONTROL1H	—	—	NMC	ATX	DART	MBC2	MBC1	MBC0	
MB_0[31]. CONTROL1L	—	—	—	—	DLC3	DLC2	DLC1	DLC0	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
MB_0[31]. TIMESTAMP	TS15	TS14	TS13	TS12	TS11	TS10	TS9	TS8	RCAN-TL1_0
	TS7	TS6	TS5	TS4	TS3	TS2	TS1	TS0	
RCANMON_0	—	CTxSTP	RCANE	—	—	—	CTxD	CRxD	
RCANMON_1	—	CTxSTP	RCANE	—	—	—	CTxD	CRxD	RCAN-TL1_1
MCR_1 ↓ MB_1[31]. CONTROL0H to MB_1[31]. TIMESTAMP	RCAN-TL1_1: Same as bit configuration of RCAN-TL1_0								
SPCRA	SPRIE	SPE	SPTIE	SPEIE	MSTR	MODFEN	TXMD	SPMS	RSPI
SSLPA	—	—	—	—	SSL3P	SSL2P	SSL1P	SSL0P	
SPPCRA	—	—	MOIFE	MOIFV	—	SPOM	SPLP2	SPLP	
SPSRA	SPRF	—	SPTEF	—	PERF	MODF	IDLNF	OVRF	
SPDRA	SPD31	SPD30	SPD29	SPD28	SPD27	SPD26	SPD25	SPD24	
	SPD23	SPD22	SPD21	SPD20	SPD19	SPD18	SPD17	SPD16	
	SPD15	SPD14	SPD13	SPD12	SPD11	SPD10	SPD9	SPD8	
	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0	
SPSCRA	—	—	—	—	—	SPSLN2	SPSLN1	SPSLN0	
SPSSRA	—	SPECM2	SPECM1	SPECM0	—	SPCP2	SPCP1	SPCP0	
SPBRA	SPR7	SPR6	SPR5	SPR4	SPR3	SPR2	SPR1	SPR0	
SPDCRA	—	—	SPLW	SPRDTD	SLSEL1	SLSEL0	SPFC1	SPFC0	
SPCKDA	—	—	—	—	—	SCKDL2	SCKDL1	SCKDL0	
SSLNDA	—	—	—	—	—	SLNDL2	SLNDL1	SLNDL0	
SPNDA	—	—	—	—	—	SPNDL2	SPNDL1	SPNDL0	
SPCR2A	—	—	—	—	PTE	SPIIE	SPOE	SPPE	
SPCMDA0	SCKDEN_0	SLNDEN_0	SPNDEN_0	LSBF_0	SPB3_0	SPB2_0	SPB1_0	SPB0_0	
	SSLKP_0	—	SSLA1_0	SSLA0_0	BRDV1_0	BRDV0_0	CPOL_0	CPHA_0	
SPCMDA1	SCKDEN_1	SLNDEN_1	SPNDEN_1	LSBF_1	SPB3_1	SPB2_1	SPB1_1	SPB0_1	
	SSLKP_1	—	SSLA1_1	SSLA0_1	BRDV1_1	BRDV0_1	CPOL_1	CPHA_1	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
SPCMDA2	SCKDEN_2	SLNDEN_2	SPNDEN_2	LSBF_2	SPB3_2	SPB2_2	SPB1_2	SPB0_2	RSPI
	SSLKP_2	—	SSLA1_2	SSLA0_2	BRDV1_2	BRDV0_2	CPOL_2	CPHA_2	
SPCMDA3	SCKDEN_3	SLNDEN_3	SPNDEN_3	LSBF_3	SPB3_3	SPB2_3	SPB1_3	SPB0_3	
	SSLKP_3	—	SSLA1_3	SSLA0_3	BRDV1_3	BRDV0_3	CPOL_3	CPHA_3	
SPCMDA4	SCKDEN_4	SLNDEN_4	SPNDEN_4	LSBF_4	SPB3_4	SPB2_4	SPB1_4	SPB0_4	
	SSLKP_4	—	SSLA1_4	SSLA0_4	BRDV1_4	BRDV0_4	CPOL_4	CPHA_4	
SPCMDA5	SCKDEN_5	SLNDEN_5	SPNDEN_5	LSBF_5	SPB3_5	SPB2_5	SPB1_5	SPB0_5	
	SSLKP_5	—	SSLA1_5	SSLA0_5	BRDV1_5	BRDV0_5	CPOL_5	CPHA_5	
SPCMDA6	SCKDEN_6	SLNDEN_6	SPNDEN_6	LSBF_6	SPB3_6	SPB2_6	SPB1_6	SPB0_6	
	SSLKP_6	—	SSLA1_6	SSLA0_6	BRDV1_6	BRDV0_6	CPOL_6	CPHA_6	
SPCMDA7	SCKDEN_7	SLNDEN_7	SPNDEN_7	LSBF_7	SPB3_7	SPB2_7	SPB1_7	SPB0_7	
	SSLKP_7	—	SSLA1_7	SSLA0_7	BRDV1_7	BRDV0_7	CPOL_7	CPHA_7	
SPCRB	SPRIE	SPE	SPTIE	SPEIE	MSTR	MODFEN	TXMD	SPMS	
SSLPB	—	—	—	—	SSL3P	SSL2P	SSL1P	SSL0P	
SPPCRB	—	—	MOIFE	MOIFV	—	SPOM	SPLP2	SPLP	
SPSRB	SPRF	—	SPTEF	—	PERF	MODF	IDLNF	OVRF	
SPDRB	SPD31	SPD30	SPD29	SPD28	SPD27	SPD26	SPD25	SPD24	
	SPD23	SPD22	SPD21	SPD20	SPD19	SPD18	SPD17	SPD16	
	SPD15	SPD14	SPD13	SPD12	SPD11	SPD10	SPD9	SPD8	
	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0	
SPSCRB	—	—	—	—	—	SPSLN2	SPSLN1	SPSLN0	
SPSSRB	—	SPECM2	SPECM1	SPECM0	—	SPCP2	SPCP1	SPCP0	
SPBRB	SPR7	SPR6	SPR5	SPR4	SPR3	SPR2	SPR1	SPR0	
SPDCRB	—	—	SPLW	SPRDTD	SLSEL1	SLSEL0	SPFC1	SPFC0	
SPCKDB	—	—	—	—	—	SCKDL2	SCKDL1	SCKDL0	
SSLNDB	—	—	—	—	—	SLNDL2	SLNDL1	SLNDL0	
SPNDB	—	—	—	—	—	SPNDL2	SPNDL1	SPNDL0	
SPCR2B	—	—	—	—	PTE	SPIIE	SPOE	SPPE	
SPCMDB0	SCKDEN_0	SLNDEN_0	SPNDEN_0	LSBF_0	SPB3_0	SPB2_0	SPB1_0	SPB0_0	
	SSLKP_0	—	SSLA1_0	SSLA0_0	BRDV1_0	BRDV0_0	CPOL_0	CPHA_0	
SPCMDB1	SCKDEN_1	SLNDEN_1	SPNDEN_1	LSBF_1	SPB3_1	SPB2_1	SPB1_1	SPB0_1	
	SSLKP_1	—	SSLA1_1	SSLA0_1	BRDV1_1	BRDV0_1	CPOL_1	CPHA_1	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
SPCMDB2	SCKDEN_2	SLNDEN_2	SPNDEN_2	LSBF_2	SPB3_2	SPB2_2	SPB1_2	SPB0_2	RSPI
	SSLKP_2	—	SSLA1_2	SSLA0_2	BRDV1_2	BRDV0_2	CPOL_2	CPHA_2	
SPCMDB3	SCKDEN_3	SLNDEN_3	SPNDEN_3	LSBF_3	SPB3_3	SPB2_3	SPB1_3	SPB0_3	
	SSLKP_3	—	SSLA1_3	SSLA0_3	BRDV1_3	BRDV0_3	CPOL_3	CPHA_3	
SPCMDB4	SCKDEN_4	SLNDEN_4	SPNDEN_4	LSBF_4	SPB3_4	SPB2_4	SPB1_4	SPB0_4	
	SSLKP_4	—	SSLA1_4	SSLA0_4	BRDV1_4	BRDV0_4	CPOL_4	CPHA_4	
SPCMDB5	SCKDEN_5	SLNDEN_5	SPNDEN_5	LSBF_5	SPB3_5	SPB2_5	SPB1_5	SPB0_5	
	SSLKP_5	—	SSLA1_5	SSLA0_5	BRDV1_5	BRDV0_5	CPOL_5	CPHA_5	
SPCMDB6	SCKDEN_6	SLNDEN_6	SPNDEN_6	LSBF_6	SPB3_6	SPB2_6	SPB1_6	SPB0_6	
	SSLKP_6	—	SSLA1_6	SSLA0_6	BRDV1_6	BRDV0_6	CPOL_6	CPHA_6	
SPCMDB7	SCKDEN_7	SLNDEN_7	SPNDEN_7	LSBF_7	SPB3_7	SPB2_7	SPB1_7	SPB0_7	
	SSLKP_7	—	SSLA1_7	SSLA0_7	BRDV1_7	BRDV0_7	CPOL_7	CPHA_7	
SPCRC	SPRIE	SPE	SPTIE	SPEIE	MSTR	MODFEN	TXMD	SPMS	
SSLPC	—	—	—	—	SSL3P	SSL2P	SSL1P	SSL0P	
SPPCRC	—	—	MOIFE	MOIFV	—	SPOM	SPLP2	SPLP	
SPSRC	SPRF	—	SPTEF	—	PERE	MODF	IDLNF	OVRF	
SPDRC	SPD31	SPD30	SPD29	SPD28	SPD27	SPD26	SPD25	SPD24	
	SPD23	SPD22	SPD21	SPD20	SPD19	SPD18	SPD17	SPD16	
	SPD15	SPD14	SPD13	SPD12	SPD11	SPD10	SPD9	SPD8	
	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0	
SPSCRC	—	—	—	—	—	SPSLN2	SPSLN1	SPSLN0	
SPSSRC	—	SPECM2	SPECM1	SPECM0	—	SPCP2	SPCP1	SPCP0	
SPBRC	SPR7	SPR6	SPR5	SPR4	SPR3	SPR2	SPR1	SPR0	
SPDCRC	—	—	SPLW	SPRDTD	SLSEL1	SLSEL0	SPFC1	SPFC0	
SPCKDC	—	—	—	—	—	SCKDL2	SCKDL1	SCKDL0	
SSLNDC	—	—	—	—	—	SLNDL2	SLNDL1	SLNDL0	
SPNDC	—	—	—	—	—	SPNDL2	SPNDL1	SPNDL0	
SPCR2C	—	—	—	—	PTE	SPIIE	SPOE	SPPE	
SPCMDC0	SCKDEN_0	SLNDEN_0	SPNDEN_0	LSBF_0	SPB3_0	SPB2_0	SPB1_0	SPB0_0	
	SSLKP_0	—	SSLA1_0	SSLA0_0	BRDV1_0	BRDV0_0	CPOL_0	CPHA_0	
SPCMDC1	SCKDEN_1	SLNDEN_1	SPNDEN_1	LSBF_1	SPB3_1	SPB2_1	SPB1_1	SPB0_1	
	SSLKP_1	—	SSLA1_1	SSLA0_1	BRDV1_1	BRDV0_1	CPOL_1	CPHA_1	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
SPCMDC2	SCKDEN_2	SLNDEN_2	SPNDEN_2	LSBF_2	SPB3_2	SPB2_2	SPB1_2	SPB0_2	RSPI
	SSLKP_2	—	SSLA1_2	SSLA0_2	BRDV1_2	BRDV0_2	CPOL_2	CPHA_2	
SPCMDC3	SCKDEN_3	SLNDEN_3	SPNDEN_3	LSBF_3	SPB3_3	SPB2_3	SPB1_3	SPB0_3	
	SSLKP_3	—	SSLA1_3	SSLA0_3	BRDV1_3	BRDV0_3	CPOL_3	CPHA_3	
SPCMDC4	SCKDEN_4	SLNDEN_4	SPNDEN_4	LSBF_4	SPB3_4	SPB2_4	SPB1_4	SPB0_4	
	SSLKP_4	—	SSLA1_4	SSLA0_4	BRDV1_4	BRDV0_4	CPOL_4	CPHA_4	
SPCMDC5	SCKDEN_5	SLNDEN_5	SPNDEN_5	LSBF_5	SPB3_5	SPB2_5	SPB1_5	SPB0_5	
	SSLKP_5	—	SSLA1_5	SSLA0_5	BRDV1_5	BRDV0_5	CPOL_5	CPHA_5	
SPCMDC6	SCKDEN_6	SLNDEN_6	SPNDEN_6	LSBF_6	SPB3_6	SPB2_6	SPB1_6	SPB0_6	
	SSLKP_6	—	SSLA1_6	SSLA0_6	BRDV1_6	BRDV0_6	CPOL_6	CPHA_6	
SPCMDC7	SCKDEN_7	SLNDEN_7	SPNDEN_7	LSBF_7	SPB3_7	SPB2_7	SPB1_7	SPB0_7	
	SSLKP_7	—	SSLA1_7	SSLA0_7	BRDV1_7	BRDV0_7	CPOL_7	CPHA_7	
SPCRD	SPRIE	SPE	SPTIE	SPEIE	MSTR	MODFEN	TXMD	SPMS	
SSLPD	—	—	—	—	SSL3P	SSL2P	SSL1P	SSL0P	
SPPCRD	—	—	MOIFE	MOIFV	—	SPOM	SPLP2	SPLP	
SPSRD	SPRF	—	SPTEF	—	PERF	MODF	IDLNF	OVRF	
SPDRD	SPD31	SPD30	SPD29	SPD28	SPD27	SPD26	SPD25	SPD24	
	SPD23	SPD22	SPD21	SPD20	SPD19	SPD18	SPD17	SPD16	
	SPD15	SPD14	SPD13	SPD12	SPD11	SPD10	SPD9	SPD8	
	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0	
SPSCRD	—	—	—	—	—	SPSLN2	SPSLN1	SPSLN0	
SPSSRD	—	SPECM2	SPECM1	SPECM0	—	SPCP2	SPCP1	SPCP0	
SPBRD	SPR7	SPR6	SPR5	SPR4	SPR3	SPR2	SPR1	SPR0	
SPDCRD	—	—	SPLW	SPRDTD	SLSEL1	SLSEL0	SPFC1	SPFC0	
SPCKDD	—	—	—	—	—	SCKDL2	SCKDL1	SCKDL0	
SSLNDD	—	—	—	—	—	SLNDL2	SLNDL1	SLNDL0	
SPNDD	—	—	—	—	—	SPNDL2	SPNDL1	SPNDL0	
SPCR2D	—	—	—	—	PTE	SPIIE	SPOE	SPPE	
SPCMDD0	SCKDEN_0	SLNDEN_0	SPNDEN_0	LSBF_0	SPB3_0	SPB2_0	SPB1_0	SPB0_0	
	SSLKP_0	—	SSLA1_0	SSLA0_0	BRDV1_0	BRDV0_0	CPOL_0	CPHA_0	
SPCMDD1	SCKDEN_1	SLNDEN_1	SPNDEN_1	LSBF_1	SPB3_1	SPB2_1	SPB1_1	SPB0_1	
	SSLKP_1	—	SSLA1_1	SSLA0_1	BRDV1_1	BRDV0_1	CPOL_1	CPHA_1	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
SPCMDD2	SCKDEN_2	SLNDEN_2	SPNDEN_2	LSBF_2	SPB3_2	SPB2_2	SPB1_2	SPB0_2	RSPI
	SSLKP_2	—	SSLA1_2	SSLA0_2	BRDV1_2	BRDV0_2	CPOL_2	CPHA_2	
SPCMDD3	SCKDEN_3	SLNDEN_3	SPNDEN_3	LSBF_3	SPB3_3	SPB2_3	SPB1_3	SPB0_3	
	SSLKP_3	—	SSLA1_3	SSLA0_3	BRDV1_3	BRDV0_3	CPOL_3	CPHA_3	
SPCMDD4	SCKDEN_4	SLNDEN_4	SPNDEN_4	LSBF_4	SPB3_4	SPB2_4	SPB1_4	SPB0_4	
	SSLKP_4	—	SSLA1_4	SSLA0_4	BRDV1_4	BRDV0_4	CPOL_4	CPHA_4	
SPCMDD5	SCKDEN_5	SLNDEN_5	SPNDEN_5	LSBF_5	SPB3_5	SPB2_5	SPB1_5	SPB0_5	
	SSLKP_5	—	SSLA1_5	SSLA0_5	BRDV1_5	BRDV0_5	CPOL_5	CPHA_5	
SPCMDD6	SCKDEN_6	SLNDEN_6	SPNDEN_6	LSBF_6	SPB3_6	SPB2_6	SPB1_6	SPB0_6	
	SSLKP_6	—	SSLA1_6	SSLA0_6	BRDV1_6	BRDV0_6	CPOL_6	CPHA_6	
SPCMDD7	SCKDEN_7	SLNDEN_7	SPNDEN_7	LSBF_7	SPB3_7	SPB2_7	SPB1_7	SPB0_7	
	SSLKP_7	—	SSLA1_7	SSLA0_7	BRDV1_7	BRDV0_7	CPOL_7	CPHA_7	
LINCR	LINE	MSS	RXDMRS	SBSTR	RXDSF	BCIE	SBIE	SFIE	HWLIN
LINSTR	BCE	—	TOIE	RXDS	TOER	BCER	SBEND	SFEND	
LINTCR	TOCSTR	TCK1	TCK0	—	—	—	TCSTF	TCSTR	
LINTCNT									
LINTOCNT									
CRCCR	DORCLR					LMS	G1	G0	CRC
CRCDIR									
CRCDOR									
FMODR	—	—	—	FRDMD	—	—	—	—	FLASH/ EEPROM
FASTAT	ROMAE	—	—	CMDLK	EEPAE	—	EEPRPE	EEPWPE	
FAEINT	ROMAEIE	—	—	CMDLKIE	EEPAEIE	—	EEPRPEIE	EEPWPEIE	
ROMMAT	RMKEY7	RMKEY6	RMKEY5	RMKEY4	RMKEY3	RMKEY2	RMKEY1	RMKEY0	
	—	—	—	—	—	—	—	ROMSEL	
EEPRE0	REKEY07	REKEY06	REKEY05	REKEY04	REKEY03	REKEY02	REKEY01	REKEY00	
	DBRE07	DBRE06	DBRE05	DBRE04	DBRE03	DBRE02	DBRE01	DBRE00	
EEPRE1	REKEY15	REKEY14	REKEY13	REKEY12	REKEY11	REKEY10	REKEY09	REKEY08	
	DBRE15	DBRE14	DBRE13	DBRE12	DBRE11	DBRE10	DBRE09	DBRE08	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
EEPWE0	WEKEY07	WEKEY06	WEKEY05	WEKEY04	WEKEY03	WEKEY02	WEKEY01	WEKEY00	FLASH/ EEPROM
	DBWE07	DBWE06	DBWE05	DBWE04	DBWE03	DBWE02	DBWE01	DBWE00	
EEPWE1	WEKEY15	WEKEY14	WE0KEY13	WEKEY12	WEKEY11	WEKEY10	WEKEY09	WEKEY08	
	DBWE15	DBWE14	DBWE13	DBWE12	DBWE11	DBWE10	DBWE09	DBWE08	
FCURAME	FCKEY7	FCKEY6	FCKEY5	FCKEY4	FCKEY3	FCKEY2	FCKEY1	FCKEY0	
	—	—	—	—	—	—	—	FCRME	
FSTSTR0	FRDY	ILGLERR	ERSERR	PRGERR	SUSRDY	—	ERSSPD	PRGSPD	
FSTATR1	FCUERR	—	—	FLOCKST	—	—	FRDTCT	FRCRCT	
FENTRYR	FEKEY7	FEKEY6	FEKEY5	FEKEY4	FEKEY3	FEKEY2	FEKEY1	FEKEY0	
	FENTRYD	—	—	—	—	—	—	FENTRY0	
FPROTR	FPKEY7	FPKEY6	FPKEY5	FPKEY4	FPKEY3	FPKEY2	FPKEY1	FPKEY0	
	—	—	—	—	—	—	—	FPROTCN	
FRESETR	FRKEY7	FRKEY6	FRKEY5	FRKEY4	FRKEY3	FRKEY2	FRKEY1	FRKEY0	
	—	—	—	—	—	—	—	FRESET	
FCMDR	CMDR7	CMDR6	CMDR5	CMDR4	CMDR3	CMDR2	CMDR1	CMDR0	
	PCMDR7	PCMDR6	PCMDR5	PCMDR4	PCMDR3	PCMDR2	PCMDR1	PCMDR0	
FRAMECCR	—	—	—	—	—	—	FRDCLE	FRCCLE	
FCPSR	—	—	—	—	—	—	—	—	
	—	—	—	—	—	—	—	ESUSPMD	
EEPBCCNT	—	—	BCADR13	BCADR12	BCADR11	BCADR10	BCADR09	BCADR08	
	BCADR07	BCADR06	BCADR05	BCADR04	BCADR03	—	—	BCSIZE	
FPESTAT	—	—	—	—	—	—	—	—	
	PEERRST7	PEERRST6	PEERRST5	PEERRST4	PEERRST3	PEERRST2	PEERRST1	PEERRST0	
EEPBCSTAT	—	—	—	—	—	—	—	—	
	—	—	—	—	—	—	—	BCST	
FCKAR	—	—	—	—	—	—	—	—	
	FCKA7	FCKA6	FCKA5	FCKA4	FCKA3	FCKA2	FCKA1	FCKA0	
EEPMAT	EMKEY7	EMKEY6	EMKEY5	EMKEY4	EMKEY3	EMKEY2	EMKEY1	EMKEY0	
	—	—	—	—	—	—	—	EEPSEL	
RAMEN	RNKEY7	RNKEY6	RNKEY5	RNKEY4	RNKEY3	RNKEY2	RNKEY1	RNKEY0	RAM
	—	—	—	RAME4	RAME3	RAME2	RAME1	RAME0	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
RAMWEN	RWNKEY7	RWNKEY6	RWNKEY5	RWNKEY4	RWNKEY3	RWNKEY2	RWNKEY1	RWNKEY0	RAM
	—	—	—	RAMWE4	RAMWE3	RAMWE2	RAMWE1	RAMWE0	
RAMECC	REKEY7	REKEY6	REKEY5	REKEY4	REKEY3	REKEY2	REKEY1	REKEY0	
	—	—	—	—	—	—	—	RECCA	
RAMERR	—	—	—	—	—	—	RDTCT	RERRC	
RAMINT	—	—	—	—	—	—	REDIE	RINTC	
RAMCYC	RAKEY7	RAKEY6	RAKEY5	RAKEY4	RAKEY3	RAKEY2	RAKEY1	RAKEY0	
	—	—	WRCYC1	WRCYC0	—	—	—	—	
ADDRA_0			—	—	—	—	—	—	A/D_0
ADDRB_0			—	—	—	—	—	—	
ADDRC_0			—	—	—	—	—	—	
ADDRD_0			—	—	—	—	—	—	
ADDRE_0			—	—	—	—	—	—	
ADDRF_0			—	—	—	—	—	—	
ADDRG_0			—	—	—	—	—	—	
ADDRH_0			—	—	—	—	—	—	
ADCSR_0	ADF	ADIE	ADST	—	CH3	CH2	CH1	CH0	
ADCR_0	TRGS1	TRGS0	SCANE	SCANS	CKS1	CKS0	ADSTCLR	EXTRGS	
APPDCR_0	AN7PD	AN6PD	AN5PD	AN4PD	AN3PD	AN2PD	AN1PD	AN0PD	
ADDIAGR_0							DIAG1	DIAG0	
ADDRA_1			—	—	—	—	—	—	A/D_1
ADDRB_1			—	—	—	—	—	—	
ADDRC_1			—	—	—	—	—	—	
ADDRD_1			—	—	—	—	—	—	
ADDRE_1			—	—	—	—	—	—	
ADDRF_1			—	—	—	—	—	—	
ADDRG_1			—	—	—	—	—	—	
ADDRH_1			—	—	—	—	—	—	
ADCSR_1	ADF	ADIE	ADST	—	CH3	CH2	CH1	CH0	
ADCR_1	TRGS1	TRGS0	SCANE	SCANS	CKS1	CKS0	ADSTCLR	EXTRGS	
APPDCR_1	AN15PD	AN14PD	AN13PD	AN12PD	A11PD	AN10PD	AN9PD	AN8PD	
ADDIAGR_1							DIAG1	DIAG0	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
TSTRB	—	—	CST5	CST4	CST3	CST2	CST1	CST0	TPU (unit 1)
TSYRB	—	—	SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0	
TCR_6	CCLR2	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	TPU_6
TMDR_6	—	—	BFB	BFA	MD3	MD2	MD1	MD0	
TIORH_6	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0	
TIORL_6	IOD3	IOD2	IOD1	IOD0	IOC3	IOC2	IOC1	IOC0	
TIER_6	TTGE	—	—	TCIEV	TGIED	TGIEC	TGIEB	TGIEA	
TSR_6	—	—	—	TCFV	TGFD	TGFC	TGFB	TGFA	
TCNT_6									
TGRA_6									
TGRB_6									
TGRC_6									
TGRD_6									
TCR_7	—	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	TPU_7
TMDR_7	—	—	—	—	MD3	MD2	MD1	MD0	
TIOR_7	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0	
TIER_7	TTGE	—	TCIEU	TCIEV	—	—	TGIEB	TGIEA	
TSR_7	TCFD	—	TCFU	TCFV	—	—	TGFB	TGFA	
TCNT_7									
TGRA_7									
TGRB_7									
TCR_8	—	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	TPU_8
TMDR_8	—	—	—	—	MD3	MD2	MD1	MD0	
TIOR_8	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
TIER_8	TTGE	—	TCIEU	TCIEV	—	—	TGIEB	TGIEA	TPU_8
TSR_8	TCFD	—	TCFU	TCFV	—	—	TGFB	TGFA	
TCNT_8									
TGRA_8									
TGRB_8									
TCR_9	CCLR2	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	TPU_9
TMDR_9	—	—	BFB	BFA	MD3	MD2	MD1	MD0	
TIORH_9	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0	
TIORL_9	IOD3	IOD2	IOD1	IOD0	IOC3	IOC2	IOC1	IOC0	
TIER_9	TTGE	—	—	TCIEV	TGIED	TGIEC	TGIEB	TGIEA	
TSR_9	—	—	—	TCFV	TGFD	TGFC	TGFB	TGFA	
TCNT_9									
TGRA_9									
TGRB_9									
TGRC_9									
TGRD_9									
TCR_10	—	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	TPU_10
TMDR_10	—	—	—	—	MD3	MD2	MD1	MD0	
TIOR_10	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0	
TIER_10	TTGE	—	TCIEU	TCIEV	—	—	TGIEB	TGIEA	
TSR_10	TCFD	—	TCFU	TCFV	—	—	TGFB	TGFA	
TCNT_10									

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
TGRA_10									TPU_10
TGRB_10									
TCR_11	—	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	TPU_11
TMDR_11	—	—	—	—	MD3	MD2	MD1	MD0	
TIOR_11	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0	
TIER_11	TTGE	—	TCIEU	TCIEV	—	—	TGIEB	TGIEA	
TSR_11	TCFD	—	TCFU	TCFV	—	—	TGFB	TGFA	
TCNT_11									
TGRA_11									
TGRB_11									
P1DDR	P17DDR	P16DDR	P15DDR	P14DDR	P13DDR	P12DDR	P11DDR	P10DDR	I/O port
P3DDR	P37DDR	P36DDR	P35DDR	P34DDR	P33DDR	P32DDR	P31DDR	P30DDR	
P6DDR	—	P66DDR	P65DDR	P64DDR	P63DDR	P62DDR	P61DDR	P60DDR	
PADDR	PA7DDR	PA6DDR	PA5DDR	PA4DDR	PA3DDR	PA2DDR	PA1DDR	—	
PDDDR	PD7DDR	PD6DDR	PD5DDR	PD4DDR	PD3DDR	PD2DDR	PD1DDR	PD0DDR	
P1ICR	P17ICR	P16ICR	P15ICR	P14ICR	P13ICR	P12ICR	P11ICR	P10ICR	
P3ICR	P37ICR	P36ICR	P35ICR	P34ICR	P33ICR	P32ICR	P31ICR	P30ICR	
P4ICR	P47ICR	P46ICR	P45ICR	P44ICR	P43ICR	P42ICR	P41ICR	P40ICR	
P5ICR	P57ICR	P56ICR	P55ICR	P54ICR	P53ICR	P52ICR	P51ICR	P50ICR	
P6ICR	—	P66ICR	P65ICR	P64ICR	P63ICR	P62ICR	P61ICR	P60ICR	
PAICR	PA7ICR	PA6ICR	PA5ICR	PA4ICR	PA3ICR	PA2ICR	PA1ICR	—	
PDICR	PD7ICR	PD6ICR	PD5ICR	PD4ICR	PD3ICR	PD2ICR	PD1ICR	PD0ICR	
PORTH	PH7	PH6	PH5	PH4	PH3	PH2	PH1	PH0	
PORTJ	PJ7	PJ6	PJ5	PJ4	PJ3	PJ2	PJ1	PJ0	
PORTK	PK7	PK6	PK5	PK4	PK3	PK2	PK1	PK0	
PHDR	PH7DR	PH6DR	PH5DR	PH4DR	PH3DR	PH2DR	PH1DR	PH0DR	
PJDR	PJ7DR	PJ6DR	PJ5DR	PJ4DR	PJ3DR	PJ2DR	PJ1DR	PJ0DR	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
PKDR	PK7DR	PK6DR	PK5DR	PK4DR	PK3DR	PK2DR	PK1DR	PK0DR	I/O port
PHDDR	PH7DDR	PH6DDR	PH5DDR	PH4DDR	PH3DDR	PH2DDR	PH1DDR	PH0DDR	
PJDDR	PJ7DDR	PJ6DDR	PJ5DDR	PJ4DDR	PJ3DDR	PJ2DDR	PJ1DDR	PJ0DDR	
PKDDR	PK7DDR	PK6DDR	PK5DDR	PK4DDR	PK3DDR	PK2DDR	PK1DDR	PK0DDR	
PHICR	PH7ICR	PH6ICR	PH5ICR	PH4ICR	PH3ICR	PH2ICR	PH1ICR	PH0ICR	
PJICR	PJ7ICR	PJ6ICR	PJ5ICR	PJ4ICR	PJ3ICR	PJ2ICR	PJ1ICR	PJ0ICR	
PKICR	PK7ICR	PK6ICR	PK5ICR	PK4ICR	PK3ICR	PK2ICR	PK1ICR	PK0ICR	
PDPCR	PD7PCR	PD6PCR	PD5PCR	PD4PCR	PD3PCR	PD2PCR	PD1PCR	PD0PCR	
PHPCR	PH7PCR	PH6PCR	PH5PCR	PH4PCR	PH3PCR	PH2PCR	PH1PCR	PH0PCR	
PJPCR	PJ7PCR	PJ6PCR	PJ5PCR	PJ4PCR	PJ3PCR	PJ2PCR	PJ1PCR	PJ0PCR	
PKPCR	PK7PCR	PK6PCR	PK5PCR	PK4PCR	PK3PCR	PK2PCR	PK1PCR	PK0PCR	
PFCR5	—	—	—	RCANMD	—	—	RCAN1S	RCAN0S	
PFCR6	—	—	—	—	TCLKS	—	—	—	
PFCR8	—	—	HLIS2	HLIS1	HLIS0	RSPISA	—	—	
PFCR9	—	—	—	—	TPUMS2	TPUMS1	TPUMS0A	TPUMS0B	
PFCRA	TPUMS11	TPUMS10	TPUMS9A	TPUMS9B	TPUMS8	TPUMS7	TPUMS6A	TPUMS6B	
PFCRB	—	ITS14	ITS13	ITS12	ITS11	ITS10	ITS9	ITS8	
PFCRC	ITS7	ITS6	ITS5	ITS4	ITS3	ITS2	ITS1	ITS0	
PFCRD	—	—	—	—	—	—	—	DRVDWNE	
SSIER	—	SSI14	SSI13	SSI12	SSI11	SSI10	SSI9	SSI8	INTC
	SSI7	SSI6	SSI5	SSI4	SSI3	SSI2	SSI1	SSI0	
P1DSR	P17DSR	P16DSR	P15DSR	P14DSR	P13DSR	P12DSR	P11DSR	P10DSR	I/O port
P6DSR	—	P66DSR	P65DSR	P64DSR	P63DSR	P62DSR	P61DSR	P60DSR	
PADSR	PA7DSR	PA6DSR	PA5DSR	PA4DSR	PA3DSR	PA2DSR	PA1DSR	—	
PDDSR	PD7DSR	PD6DSR	PD5DSR	PD4DSR	PD3DSR	PD2DSR	PD1DSR	PD0DSR	
PHDSR	PH7DSR	PH6DSR	PH5DSR	PH4DSR	PH3DSR	PH2DSR	PH1DSR	PH0DSR	
P1PSR	P17PSR	P16PSR	P15PSR	P14PSR	P13PSR	P12PSR	P11PSR	P10PSR	
P6PSR	—	P66PSR	P65PSR	P64PSR	P63PSR	P62PSR	P61PSR	P60PSR	
PAPSR	PA7PSR	PA6PSR	PA5PSR	PA4PSR	PA3PSR	PA2PSR	PA1PSR	—	
PDPSR	PD7PSR	PD6PSR	PD5PSR	PD4PSR	PD3PSR	PD2PSR	PD1PSR	PD0PSR	
PHPSR	PH7PSR	PH6PSR	PH5PSR	PH4PSR	PH3PSR	PH2PSR	PH1PSR	PH0PSR	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
DSAR_0									DMAC_0
DDAR_0									
DOFR_0									
DTCR_0									
DBSR_0	BKSZH31	BKSZH30	BKSZH29	BKSZH28	BKSZH27	BKSZH26	BKSZH25	BKSZH24	
	BKSZH23	BKSZH22	BKSZH21	BKSZH20	BKSZH19	BKSZH18	BKSZH17	BKSZH16	
	BKSZ15	BKSZ14	BKSZ13	BKSZ12	BKSZ11	BKSZ10	BKSZ9	BKSZ8	
	BKSZ7	BKSZ6	BKSZ5	BKSZ4	BKSZ3	BKSZ2	BKSZ1	BKSZ0	
DMDR_0	DTE	DACKE	TENDE	—	DREQS	NRD	—	—	
	ACT	—	—	—	ERRF	—	ESIF	DTIF	
	DTSZ1	DTSZ0	MDS1	MDS0	TSEIE	—	ESIE	DTIE	
	DTF1	DTF0	DTA	—	—	DMAP2	DMAP1	DMAP0	
DACR_0	AMS	DIRS	—	—	—	RPTIE	ARS1	ARS0	
	—	—	SAT1	SAT0	—	—	DAT1	DAT0	
	SARIE	—	—	SARA4	SARA3	SARA2	SARA1	SARA0	
	DARIE	—	—	DARA4	DARA3	DARA2	DARA1	DARA0	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
DSAR_1									DMAC_1
DDAR_1									
DOFR_1									
DTCR_1									
DBSR_1	BKSZH31	BKSZH30	BKSZH29	BKSZH28	BKSZH27	BKSZH26	BKSZH25	BKSZH24	
	BKSZH23	BKSZH22	BKSZH21	BKSZH20	BKSZH19	BKSZH18	BKSZH17	BKSZH16	
	BKSZ15	BKSZ14	BKSZ13	BKSZ12	BKSZ11	BKSZ10	BKSZ9	BKSZ8	
	BKSZ7	BKSZ6	BKSZ5	BKSZ4	BKSZ3	BKSZ2	BKSZ1	BKSZ0	
DMDR_1	DTE	DACKE	TENDE	—	DREQS	NRD	—	—	
	ACT	—	—	—	—	—	ESIF	DTIF	
	DTSZ1	DTSZ0	MDS1	MDS0	TSEIE	—	ESIE	DTIE	
	DTF1	DTF0	DTA	—	—	DMAP2	DMAP1	DMAP0	
DACR_1	AMS	DIRS	—	—	—	RPTIE	ARS1	ARS0	
	—	—	SAT1	SAT0	—	—	DAT1	DAT0	
	SARIE	—	—	SARA4	SARA3	SARA2	SARA1	SARA0	
	DARIE	—	—	DARA4	DARA3	DARA2	DARA1	DARA0	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
DSAR_2									DMAC_2
DDAR_2									
DOFR_2									
DTCR_2									
DBSR_2	BKSZH31	BKSZH30	BKSZH29	BKSZH28	BKSZH27	BKSZH26	BKSZH25	BKSZH24	
	BKSZH23	BKSZH22	BKSZH21	BKSZH20	BKSZH19	BKSZH18	BKSZH17	BKSZH16	
	BKSZ15	BKSZ14	BKSZ13	BKSZ12	BKSZ11	BKSZ10	BKSZ9	BKSZ8	
	BKSZ7	BKSZ6	BKSZ5	BKSZ4	BKSZ3	BKSZ2	BKSZ1	BKSZ0	
DMDR_2	DTE	DACKE	TENDE	—	DREQS	NRD	—	—	
	ACT	—	—	—	—	—	ESIF	DTIF	
	DTSZ1	DTSZ0	MDS1	MDS0	TSEIE	—	ESIE	DTIE	
	DTF1	DTF0	DTA	—	—	DMAP2	DMAP1	DMAP0	
DACR_2	AMS	DIRS	—	—	—	RPTIE	ARS1	ARS0	
	—	—	SAT1	SAT0	—	—	DAT1	DAT0	
	SARIE	—	—	SARA4	SARA3	SARA2	SARA1	SARA0	
	DARIE	—	—	DARA4	DARA3	DARA2	DARA1	DARA0	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
DSAR_3									DMAC_3
DDAR_3									
DOFR_3									
DTCR_3									
DBSR_3	BKSZH31	BKSZH30	BKSZH29	BKSZH28	BKSZH27	BKSZH26	BKSZH25	BKSZH24	
	BKSZH23	BKSZH22	BKSZH21	BKSZH20	BKSZH19	BKSZH18	BKSZH17	BKSZH16	
	BKSZ15	BKSZ14	BKSZ13	BKSZ12	BKSZ11	BKSZ10	BKSZ9	BKSZ8	
	BKSZ7	BKSZ6	BKSZ5	BKSZ4	BKSZ3	BKSZ2	BKSZ1	BKSZ0	
DMDR_3	DTE	DACKE	TENDE	—	DREQS	NRD	—	—	
	ACT	—	—	—	—	—	ESIF	DTIF	
	DTSZ1	DTSZ0	MDS1	MDS0	TSEIE	—	ESIE	DTIE	
	DTF1	DTF0	DTA	—	—	DMAP2	DMAP1	DMAP0	
DACR_3	AMS	DIRS	—	—	—	RPTIE	ARS1	ARS0	
	—	—	SAT1	SAT0	—	—	DAT1	DAT0	
	SARIE	—	—	SARA4	SARA3	SARA2	SARA1	SARA0	
	DARIE	—	—	DARA4	DARA3	DARA2	DARA1	DARA0	
DMRSR_0									DMAC_0
DMRSR_1									DMAC_1
DMRSR_2									DMAC_2
DMRSR_3									DMAC_3

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
IPRA	—	IPRA14	IPRA13	IPRA12	—	IPRA10	IPRA9	IPRA8	INTC
	—	IPRA6	IPRA5	IPRA4	—	IPRA2	IPRA1	IPRA0	
IPRB	—	IPRB14	IPRB13	IPRB12	—	IPRB10	IPRB9	IPRB8	
	—	IPRB6	IPRB5	IPRB4	—	IPRB2	IPRB1	IPRB0	
IPRC	—	IPRC14	IPRC13	IPRC12	—	IPRC10	IPRC9	IPRC8	
	—	IPRC6	IPRC5	IPRC4	—	IPRC2	IPRC1	IPRC0	
IPRD	—	IPRD14	IPRD13	IPRD12	—	IPRD10	IPRD9	IPRD8	
	—	IPRD6	IPRD5	IPRD4	—	IPRD2	IPRD1	IPRD0	
IPRE	—	IPRE14	IPRE13	IPRE12	—	IPRE10	IPRE9	IPRE8	
	—	IPRE6	IPRE5	IPRE4	—	IPRE2	IPRE1	IPRE0	
IPRF	—	IPRF14	IPRF13	IPRF12	—	IPRF10	IPRF9	IPRF8	
	—	IPRF6	IPRF5	IPRF4	—	IPRF2	IPRF1	IPRF0	
IPRG	—	IPRG14	IPRG13	IPRG12	—	IPRG10	IPRG9	IPRG8	
	—	IPRG6	IPRG5	IPRG4	—	IPRG2	IPRG1	IPRG0	
IPRI	—	IPRI14	IPRI13	IPRI12	—	IPRI10	IPRI9	IPRI8	
	—	IPRI6	IPRI5	IPRI4	—	IPRI2	IPRI1	IPRI0	
IPRJ	—	IPRJ14	IPRJ13	IPRJ12	—	IPRJ10	IPRJ9	IPRJ8	
	—	IPRJ6	IPRJ5	IPRJ4	—	IPRJ2	IPRJ1	IPRJ0	
IPRL	—	IPRL14	IPRL13	IPRL12	—	IPRL10	IPRL9	IPRL8	
	—	IPRL6	IPRL5	IPRL4	—	IPRL2	IPRL1	IPRL0	
IPRM	—	IPRM14	IPRM13	IPRM12	—	IPRM10	IPRM9	IPRM8	
	—	IPRM6	IPRM5	IPRM4	—	IPRM2	IPRM1	IPRM0	
IPRN	—	IPRN14	IPRN13	IPRN12	—	IPRN10	IPRN9	IPRN8	
	—	IPRN6	IPRN5	IPRN4	—	IPRN2	IPRN1	IPRN0	
IPRO	—	IPRO14	IPRO13	IPRO12	—	IPRO10	IPRO9	IPRO8	
	—	IPRO6	IPRO5	IPRO4	—	IPRO2	IPRO1	IPRO0	
IPRP	—	IPRP14	IPRP13	IPRP12	—	IPRP10	IPRP9	IPRP8	
	—	IPRP6	IPRP5	IPRP4	—	IPRP2	IPRP1	IPRP0	
ISCRH	—	—	IRQ14SR	IRQ14SF	IRQ13SR	IRQ13SF	IRQ12SR	IRQ12SF	
	IRQ11SR	IRQ11SF	IRQ10SR	IRQ10SF	IRQ9SR	IRQ9SF	IRQ8SR	IRQ8SF	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
ISCRL	IRQ7SR	IRQ7SF	IRQ6SR	IRQ6SF	IRQ5SR	IRQ5SF	IRQ4SR	IRQ4SF	INTC
	IRQ3SR	IRQ3SF	IRQ2SR	IRQ2SF	IRQ1SR	IRQ1SF	IRQ0SR	IRQ0SF	
DTCVBR									BSC
BCR2	—	—	—	IBCCS	—	—	—	PWDBE	
MDCR	—	—	—	—	MDS3	MDS2	MDS1	MDS0	SYSTEM
	—	—	—	—	—	—	—	—	
SYSCR0	—	—	MACS	—	—	—	—	RAME	
	—	—	—	—	—	—	—	—	
SCKCR0	PSTOP1	—	POSEL1	—	—	ICK2	ICK1	ICK0	
	—	PCK2	PCK1	PCK0	—	BCK2	BCK1	BCK0	
SBYCR	SSBY	OPE	SSBYF	STS4	STS3	STS2	STS1	STS0	
	SLPIE	—	—	—	—	—	—	—	
MSTPCRA	ACSE	MSTPA14	MSTPA13	MSTPA12	MSTPA11	MSTPA10	MSTPA9	MSTPA8	
	MSTPA7	MSTPA6	MSTPA5	MSTPA4	MSTPA3	MSTPA2	MSTPA1	MSTPA0	
MSTPCRB	MSTPB15	MSTPB14	MSTPB13	MSTPB12	MSTPB11	MSTPB10	MSTPB9	MSTPB8	
	MSTPB7	MSTPB6	MSTPB5	MSTPB4	MSTPB3	MSTPB2	MSTPB1	MSTPB0	
MSTPCRC	MSTPC15	MSTPC14	MSTPC13	MSTPC12	MSTPC11	MSTPC10	MSTPC9	MSTPC8	
	MSTPC7	MSTPC6	MSTPC5	MSTPC4	MSTPC3	MSTPC2	MSTPC1	MSTPC0	
ROSCCR	INOSCE	OSCERR	OSCIE	—	—	—	—	ERRTEST	
	—	—	—	—	—	—	—	—	
SYSCR1	—	—	—	—	—	—	—	—	
	—	—	—	—	—	—	—	EEPWT	
SCKCR1	—	—	—	—	—	FCK2	FCK1	FCK0	
	—	ACK2	ACK1	ACK0	—	RCK2	RCK1	RCK0	
MSTPCRD	MSTPD15	MSTPD14	MSTPD13	MSTPD12	MSTPD11	MSTPD10	MSTPD9	MSTPD8	
	MSTPD7	MSTPD6	MSTPD5	MSTPD4	MSTPD3	MSTPD2	MSTPD1	MSTPD0	
MSTPCRE	MSTPE15	MSTPE14	MSTPE13	MSTPE12	MSTPE11	MSTPE10	MSTPE9	MSTPE8	
	MSTPE7	MSTPE6	MSTPE5	MSTPE4	MSTPE3	MSTPE2	MSTPE1	MSTPE0	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
SMR_3	C/$\overline{A}$	CHR	PE	O/$\overline{E}$	STOP	MP	CKS1	CKS0	SCI_3
BRR_3									
SCR_3	TIE	RIE	TE	RE	MPIE	TEIE	CKE1	CKE0	
TDR_3									
SSR_3	TDRE	RDRF	ORER	FER	PER	TEND	MPB	MPBT	
RDR_3									
SMR_4	C/$\overline{A}$	CHR	PE	O/$\overline{E}$	STOP	MP	CKS1	CKS0	SCI_4
BRR_4									
SCR_4	TIE	RIE	TE	RE	MPIE	TEIE	CKE1	CKE0	
TDR_4									
SSR_4	TDRE	RDRF	ORER	FER	PER	TEND	MPB	MPBT	
RDR_4									
TCR_4	—	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	TPU_4
TMDR_4	—	—	—	—	MD3	MD2	MD1	MD0	
TIOR_4	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0	
TIER_4	TTGE	—	TCIEU	TCIEV	—	—	TGIEB	TGIEA	
TSR_4	TCFD	—	TCFU	TCFV	—	—	TGFB	TGFA	
TCNT4									
TGRA_4									
TGRB_4									
TCR_5	—	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	TPU_5
TMDR_5	—	—	—	—	MD3	MD2	MD1	MD0	
TIOR_5	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0	
TIER_5	TTGE	—	TCIEU	TCIEV	—	—	TGIEB	TGIEA	
TSR_5	TCFD	—	TCFU	TCFV	—	—	TGFB	TGFA	
TCNT_5									
TGRA_5									

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
TGRB_5									TPU_5
DTCERA	DTCE15	DTCE14	DTCE13	DTCE12	DTCE11	DTCE10	DTCE9	DTCE8	DTC
	DTCE7	DTCE6	DTCE5	DTCE4	DTCE3	DTCE2	DTCE1	DTCE0	
DTCERB	DTCE15	DTCE14	DTCE13	DTCE12	DTCE11	DTCE10	DTCE9	DTCE8	
	DTCE7	DTCE6	DTCE5	DTCE4	DTCE3	DTCE2	DTCE1	DTCE0	
DTCERC	DTCE15	DTCE14	DTCE13	DTCE12	DTCE11	DTCE10	DTCE9	DTCE8	
	DTCE7	DTCE6	DTCE5	DTCE4	DTCE3	DTCE2	DTCE1	DTCE0	
DTCERD	DTCE15	DTCE14	DTCE13	DTCE12	DTCE11	DTCE10	DTCE9	DTCE8	
	DTCE7	DTCE6	DTCE5	DTCE4	DTCE3	DTCE2	DTCE1	DTCE0	
DTCERE	DTCE15	DTCE14	DTCE13	DTCE12	DTCE11	DTCE10	DTCE9	DTCE8	
	DTCE7	DTCE6	DTCE5	DTCE4	DTCE3	DTCE2	DTCE1	DTCE0	
DTCERF	DTCE15	DTCE14	DTCE13	DTCE12	DTCE11	DTCE10	DTCE9	DTCE8	
	DTCE7	DTCE6	DTCE5	DTCE4	DTCE3	DTCE2	DTCE1	DTCE0	
DTCERG	DTCE15	DTCE14	DTCE13	DTCE12	DTCE11	DTCE10	DTCE9	DTCE8	
	DTCE7	DTCE6	DTCE5	DTCE4	DTCE3	DTCE2	DTCE1	DTCE0	
DTCCR	—	—	—	—	—	—	—	—	
INTCR	—	—	INTM1	INTM0	NMIEG	—	—	—	INTC
CPUPCR	CPUPCE	DTCP2	DTCP1	DTCP0	IPSETE	CPUP2	CPUP1	CPUP0	
IER	IRQ15E	IRQ14E	IRQ13E	IRQ12E	IRQ11E	IRQ10E	IRQ9E	IRQ8E	
	IRQ7E	IRQ6E	IRQ5E	IRQ4E	IRQ3E	IRQ2E	IRQ1E	IRQ0E	
ISR	IRQ15F	IRQ14F	IRQ13F	IRQ12F	IRQ11F	IRQ10F	IRQ9F	IRQ8F	
	IRQ7F	IRQ6F	IRQ5F	IRQ4F	IRQ3F	IRQ2F	IRQ1F	IRQ0F	
PORT1	P17	P16	P15	P14	P13	P12	P11	P10	I/O port
PORT3	P37	P36	P35	P34	P33	P32	P31	P30	
PORT4	P47	P46	P45	P44	P43	P42	P41	P40	
PORT5	P57	P56	P55	P54	P53	P52	P51	P50	
PORT6	—	P66	P65	P64	P63	P62	P61	P60	
PORTA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	—	
PORTD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0	
P1DR	P17DR	P16DR	P15DR	P14DR	P13DR	P12DR	P11DR	P10DR	
P3DR	P37DR	P36DR	P35DR	P34DR	P33DR	P32DR	P31DR	P30DR	

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
P6DR	—	P66DR	P65DR	P64DR	P63DR	P62DR	P61DR	P60DR	I/O port
PADR	PA7DR	PA6DR	PA5DR	PA4DR	PA3DR	PA2DR	PA1DR	—	
PDDR	PD7DR	PD6DR	PD5DR	PD4DR	PD3DR	PD2DR	PD1DR	PD0DR	
PCR	G3CMS1	G3CMS0	G2CMS1	G2CMS0	—	—	—	—	PPG
PMR	G3INV	G2INV	—	—	G3NOV	G2NOV	—	—	
NDERH	NDER15	NDER14	NDER13	NDER12	NDER11	NDER10	NDER9	NDER8	
NDERL	NDER7	NDER6	NDER5	NDER4	NDER3	NDER2	NDER1	NDER0	
PODRH	POD15	POD14	POD13	POD12	POD11	POD10	POD9	POD8	
PODRL	POD7	POD6	POD5	POD4	POD3	POD2	POD1	POD0	
NDRH	NDR15	NDR14	NDR13	NDR12	NDR11	NDR10	NDR9	NDR8	
NDRL	NDR7	NDR6	NDR5	NDR4	NDR3	NDR2	NDR1	NDR0	
TCSR	OVF	WT/$\overline{IT}$	TME	—	—	CKS2	CKS1	CKS0	WDT
TCNT									
RSTCSR	WOVF	RSTE	—	—	—	—	—	—	
TSTR	—	—	CST5	CST4	CST3	CST2	CST1	CST0	TPU (unit 0)
TSYR	—	—	SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0	
TCR_0	CCLR2	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	TPU_0
TMDR_0	—	—	BFB	BFA	MD3	MD2	MD1	MD0	
TIORH_0	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0	
TIORL_0	IOD3	IOD2	IOD1	IOD0	IOC3	IOC2	IOC1	IOC0	
TIER_0	TTGE	—	—	TCIEV	TGIED	TGIEC	TGIEB	TGIEA	
TSR_0	—	—	—	TCFV	TGFD	TGFC	TGFB	TGFA	
TCNT_0									
TGRA_0									
TGRB_0									
TGRC_0									
TGRD_0									

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
TCR_1	—	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	TPU_1
TMDR_1	—	—	—	—	MD3	MD2	MD1	MD0	
TIOR_1	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0	
TIER_1	TTGE	—	TCIEU	TCIEV	—	—	TGIEB	TGIEA	
TSR_1	TCFD	—	TCFU	TCFV	—	—	TGFB	TGFA	
TCNT_1									
TGRA_1									
TGRB_1									
TCR_2	—	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	TPU_2
TMDR_2	—	—	—	—	MD3	MD2	MD1	MD0	
TIOR_2	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0	
TIER_2	TTGE	—	TCIEU	TCIEV	—	—	TGIEB	TGIEA	
TSR_2	TCFD	—	TCFU	TCFV	—	—	TGFB	TGFA	
TCNT_2									
TGRA_2									
TGRB_2									
TCR_3	CCLR2	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0	TPU_3
TMDR_3	—	—	BFB	BFA	MD3	MD2	MD1	MD0	
TIORH_3	IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0	
TIORL_3	IOD3	IOD2	IOD1	IOD0	IOC3	IOC2	IOC1	IOC0	
TIER_3	TTGE	—	—	TCIEV	TGIED	TGIEC	TGIEB	TGIEA	
TSR_3	—	—	—	TCFV	TGFD	TGFC	TGFB	TGFA	
TCNT_3									
TGRA_3									

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0	Module
TGRB_3									TPU_3
TGRC_3									
TGRD_3									

24.3 Register States in Each Operating Mode

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
MCR_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	RCAN_0
GSR_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
BCR1_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
BCR0_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
IRR_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
IMR_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
TEC_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
REC_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
TXPR1_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
TXPR0_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
TXCR1_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
TXCR0_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
TXACK1_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
TXACK0_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
ABACK1_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
ABACK0_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
RXPR1_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
RXPR0_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
RFPR1_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
RFPR0_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MBIMR1_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MBIMR0_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
UMSR1_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
UMSR0_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
TTCR0_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
CMAX_TEW_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
RFTROFF_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
TSR_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
CCR_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
TCNTR_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	RCAN_0
CYCTR_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
RFMK_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
TCMR0_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
TCMR1_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
TCMR2_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
TTTSEL_0	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[0]. CONTROL0H	—	—	—	—	—	—	—	
MB_0[0]. CONTROL0L	—	—	—	—	—	—	—	
MB_0[0]. LAFMH	—	—	—	—	—	—	—	
MB_0[0]. LAFML	—	—	—	—	—	—	—	
MB_0[0]. MSG_DATA[0]	—	—	—	—	—	—	—	
MB_0[0]. MSG_DATA[1]	—	—	—	—	—	—	—	
MB_0[0]. MSG_DATA[2]	—	—	—	—	—	—	—	
MB_0[0]. MSG_DATA[3]	—	—	—	—	—	—	—	
MB_0[0]. MSG_DATA[4]	—	—	—	—	—	—	—	
MB_0[0]. MSG_DATA[5]	—	—	—	—	—	—	—	
MB_0[0]. MSG_DATA[6]	—	—	—	—	—	—	—	
MB_0[0]. MSG_DATA[7]	—	—	—	—	—	—	—	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
MB_0[0]. CONTROL1H	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	RCAN_0
MB_0[0]. CONTROL1L	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[0]. TIMESTAMP	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[1]. CONTROL0H	—	—	—	—	—	—	—	
MB_0[1]. CONTROL0L	—	—	—	—	—	—	—	
MB_0[1]. LAFMH	—	—	—	—	—	—	—	
MB_0[1]. LAFML	—	—	—	—	—	—	—	
MB_0[1]. MSG_DATA[0]	—	—	—	—	—	—	—	
MB_0[1]. MSG_DATA[1]	—	—	—	—	—	—	—	
MB_0[1]. MSG_DATA[2]	—	—	—	—	—	—	—	
MB_0[1]. MSG_DATA[3]	—	—	—	—	—	—	—	
MB_0[1]. MSG_DATA[4]	—	—	—	—	—	—	—	
MB_0[1]. MSG_DATA[5]	—	—	—	—	—	—	—	
MB_0[1]. MSG_DATA[6]	—	—	—	—	—	—	—	
MB_0[1]. MSG_DATA[7]	—	—	—	—	—	—	—	
MB_0[1]. CONTROL1H	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
MB_0[1]. CONTROL1L	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	RCAN_0
MB_0[1]. TIMESTAMP	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[2]. CONTROL0H to MB_0[2]. TIMESTAMP	Same as MB_0[1].CONTROL0H to MB_0[1]. TIMESTAMP							
↓	(Repeated)							
MB_0[15]. CONTROL0H to MB_0[15]. TIMESTAMP	Same as MB_0[1].CONTROL0H to MB_0[1]. TIMESTAMP							
MB_0[16]. CONTROL0H	—	—	—	—	—	—	—	
MB_0[16]. CONTROL0L	—	—	—	—	—	—	—	
MB_0[16]. LAFMH	—	—	—	—	—	—	—	
MB_0[16]. LAFML	—	—	—	—	—	—	—	
MB_0[16]. MSG_DATA[0]	—	—	—	—	—	—	—	
MB_0[16]. MSG_DATA[1]	—	—	—	—	—	—	—	
MB_0[16]. MSG_DATA[2]	—	—	—	—	—	—	—	
MB_0[16]. MSG_DATA[3]	—	—	—	—	—	—	—	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
MB_0[16]. MSG_DATA[4]	—	—	—	—	—	—	—	RCAN_0
MB_0[16]. MSG_DATA[5]	—	—	—	—	—	—	—	
MB_0[16]. MSG_DATA[6]	—	—	—	—	—	—	—	
MB_0[16]. MSG_DATA[7]	—	—	—	—	—	—	—	
MB_0[16]. CONTROL1H	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[16]. CONTROL1L	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[17]. CONTROL0H to MB_0[17]. CONTROL1L	Same as MB_0[16].CONTROL0H to MB_0[16].CONTROL1L							
↓	(Repeated)							
MB_0[23]. CONTROL0H to MB_0[23]. CONTROL1L	Same as MB_0[16].CONTROL0H to MB_0[16].CONTROL1L							
MB_0[24]. CONTROL0H	—	—	—	—	—	—	—	
MB_0[24]. CONTROL0L	—	—	—	—	—	—	—	
MB_0[24]. LAFMH	—	—	—	—	—	—	—	
MB_0[24]. LAFML	—	—	—	—	—	—	—	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
MB_0[24]. MSG_DATA[0]	—	—	—	—	—	—	—	RCAN_0
MB_0[24]. MSG_DATA[1]	—	—	—	—	—	—	—	
MB_0[24]. MSG_DATA[2]	—	—	—	—	—	—	—	
MB_0[24]. MSG_DATA[3]	—	—	—	—	—	—	—	
MB_0[24]. MSG_DATA[4]	—	—	—	—	—	—	—	
MB_0[24]. MSG_DATA[5]	—	—	—	—	—	—	—	
MB_0[24]. MSG_DATA[6]	—	—	—	—	—	—	—	
MB_0[24]. MSG_DATA[7]	—	—	—	—	—	—	—	
MB_0[24]. CONTROL1H	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[24]. CONTROL1L	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[24]. TTT	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[24]. TT_CONTROL	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[25]. CONTROL0H to MB_0[25]. TT_CONTROL	Same as MB_0[24].CONTROL0H to MB_0[24].TT_CONTROL							
↓	(Repeated)							

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
MB_0[29]. CONTROL0H to MB_0[29]. TT_CONTROL	Same as MB_0[24].CONTROL0H to MB_0[24].TT_CONTROL							RCAN_0
MB_0[30]. CONTROL0H	—	—	—	—	—	—	—	
MB_0[30]. CONTROL0L	—	—	—	—	—	—	—	
MB_0[30]. LAFMH	—	—	—	—	—	—	—	
MB_0[30]. LAFML	—	—	—	—	—	—	—	
MB_0[30]. MSG_DATA[0]	—	—	—	—	—	—	—	
MB_0[30]. MSG_DATA[1]	—	—	—	—	—	—	—	
MB_0[30]. MSG_DATA[2]	—	—	—	—	—	—	—	
MB_0[30]. MSG_DATA[3]	—	—	—	—	—	—	—	
MB_0[30]. MSG_DATA[4]	—	—	—	—	—	—	—	
MB_0[30]. MSG_DATA[5]	—	—	—	—	—	—	—	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
MB_0[30]. MSG_DATA[6]	—	—	—	—	—	—	—	RCAN_0
MB_0[30]. MSG_DATA[7]	—	—	—	—	—	—	—	
MB_0[30]. CONTROL1H	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[30]. CONTROL1L	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[30]. TIMESTAMP	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[30]. TTT	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[31]. CONTROL0H	—	—	—	—	—	—	—	
MB_0[31]. CONTROL0L	—	—	—	—	—	—	—	
MB_0[31]. LAFMH	—	—	—	—	—	—	—	
MB_0[31]. LAFML	—	—	—	—	—	—	—	
MB_0[31]. MSG_DATA[0]	—	—	—	—	—	—	—	
MB_0[31]. MSG_DATA[1]	—	—	—	—	—	—	—	
MB_0[31]. MSG_DATA[2]	—	—	—	—	—	—	—	
MB_0[31]. MSG_DATA[3]	—	—	—	—	—	—	—	
MB_0[31]. MSG_DATA[4]	—	—	—	—	—	—	—	
MB_0[31]. MSG_DATA[5]	—	—	—	—	—	—	—	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
MB_0[31]. MSG_DATA[6]	—	—	—	—	—	—	—	RCAN_0
MB_0[31]. MSG_DATA[7]	—	—	—	—	—	—	—	
MB_0[31]. CONTROL1H	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[31]. CONTROL1L	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
MB_0[31]. TIMESTAMP	Initialized	—	—	Initialized	Initialized	Initialized	Initialized	
RCANMON_0	Initialized	—	—	—	—	—	Initialized	
RCANMON_1	Initialized	—	—	—	—	—	Initialized	RCAN_1
MCR_1 ↓ MB_1[31]. CONTROL0H to MB_1[31]. TIMESTAMP	RCAN_1: Same as RCAN_0							
SPCRA	Initialized	—	—	—	—	—	Initialized	RSPI
SSLPA	Initialized	—	—	—	—	—	Initialized	
SPPCRA	Initialized	—	—	—	—	—	Initialized	
SPSRA	Initialized	—	—	—	—	—	Initialized	
SPDRA	Initialized	—	—	—	—	—	Initialized	
SPSCRA	Initialized	—	—	—	—	—	Initialized	
SPSSRA	Initialized	—	—	—	—	—	Initialized	
SPBRA	Initialized	—	—	—	—	—	Initialized	
SPDCRA	Initialized	—	—	—	—	—	Initialized	
SPCKDA	Initialized	—	—	—	—	—	Initialized	
SSLNDA	Initialized	—	—	—	—	—	Initialized	
SPNDA	Initialized	—	—	—	—	—	Initialized	
SPCR2A	Initialized	—	—	—	—	—	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
SPCMDA0	Initialized	—	—	—	—	—	Initialized	RSPI
SPCMDA1	Initialized	—	—	—	—	—	Initialized	
SPCMDA2	Initialized	—	—	—	—	—	Initialized	
SPCMDA3	Initialized	—	—	—	—	—	Initialized	
SPCMDA4	Initialized	—	—	—	—	—	Initialized	
SPCMDA5	Initialized	—	—	—	—	—	Initialized	
SPCMDA6	Initialized	—	—	—	—	—	Initialized	
SPCMDA7	Initialized	—	—	—	—	—	Initialized	
SPCRB	Initialized	—	—	—	—	—	Initialized	
SSLPB	Initialized	—	—	—	—	—	Initialized	
SPPCRB	Initialized	—	—	—	—	—	Initialized	
SPSRB	Initialized	—	—	—	—	—	Initialized	
SPDRB	Initialized	—	—	—	—	—	Initialized	
SPSCRB	Initialized	—	—	—	—	—	Initialized	
SPSSRB	Initialized	—	—	—	—	—	Initialized	
SPBRB	Initialized	—	—	—	—	—	Initialized	
SPDCRB	Initialized	—	—	—	—	—	Initialized	
SPCKDB	Initialized	—	—	—	—	—	Initialized	
SSLNDB	Initialized	—	—	—	—	—	Initialized	
SPNDB	Initialized	—	—	—	—	—	Initialized	
SPCR2B	Initialized	—	—	—	—	—	Initialized	
SPCMDB0	Initialized	—	—	—	—	—	Initialized	
SPCMDB1	Initialized	—	—	—	—	—	Initialized	
SPCMDB2	Initialized	—	—	—	—	—	Initialized	
SPCMDB3	Initialized	—	—	—	—	—	Initialized	
SPCMDB4	Initialized	—	—	—	—	—	Initialized	
SPCMDB5	Initialized	—	—	—	—	—	Initialized	
SPCMDB6	Initialized	—	—	—	—	—	Initialized	
SPCMDB7	Initialized	—	—	—	—	—	Initialized	
SPCRC	Initialized	—	—	—	—	—	Initialized	
SSLPC	Initialized	—	—	—	—	—	Initialized	
SPPCRC	Initialized	—	—	—	—	—	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
SPSRC	Initialized	—	—	—	—	—	Initialized	RSPI
SPDRC	Initialized	—	—	—	—	—	Initialized	
SPSCRC	Initialized	—	—	—	—	—	Initialized	
SPSSRC	Initialized	—	—	—	—	—	Initialized	
SPBRC	Initialized	—	—	—	—	—	Initialized	
SPDCRC	Initialized	—	—	—	—	—	Initialized	
SPCKDC	Initialized	—	—	—	—	—	Initialized	
SSLNDC	Initialized	—	—	—	—	—	Initialized	
SPNDC	Initialized	—	—	—	—	—	Initialized	
SPCR2C	Initialized	—	—	—	—	—	Initialized	
SPCMDC0	Initialized	—	—	—	—	—	Initialized	
SPCMDC1	Initialized	—	—	—	—	—	Initialized	
SPCMDC2	Initialized	—	—	—	—	—	Initialized	
SPCMDC3	Initialized	—	—	—	—	—	Initialized	
SPCMDC4	Initialized	—	—	—	—	—	Initialized	
SPCMDC5	Initialized	—	—	—	—	—	Initialized	
SPCMDC6	Initialized	—	—	—	—	—	Initialized	
SPCMDC7	Initialized	—	—	—	—	—	Initialized	
SPCRD	Initialized	—	—	—	—	—	Initialized	
SSLPD	Initialized	—	—	—	—	—	Initialized	
SPPCRD	Initialized	—	—	—	—	—	Initialized	
SPSRD	Initialized	—	—	—	—	—	Initialized	
SPDRD	Initialized	—	—	—	—	—	Initialized	
SPSCRD	Initialized	—	—	—	—	—	Initialized	
SPSSRD	Initialized	—	—	—	—	—	Initialized	
SPBRD	Initialized	—	—	—	—	—	Initialized	
SPDCRD	Initialized	—	—	—	—	—	Initialized	
SPCKDD	Initialized	—	—	—	—	—	Initialized	
SSLNDD	Initialized	—	—	—	—	—	Initialized	
SPNDD	Initialized	—	—	—	—	—	Initialized	
SPCR2D	Initialized	—	—	—	—	—	Initialized	
SPCMDD0	Initialized	—	—	—	—	—	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
SPCMDD1	Initialized	—	—	—	—	—	Initialized	RSPI
SPCMDD2	Initialized	—	—	—	—	—	Initialized	
SPCMDD3	Initialized	—	—	—	—	—	Initialized	
SPCMDD4	Initialized	—	—	—	—	—	Initialized	
SPCMDD5	Initialized	—	—	—	—	—	Initialized	
SPCMDD6	Initialized	—	—	—	—	—	Initialized	
SPCMDD7	Initialized	—	—	—	—	—	Initialized	
LINCR	Initialized	—	—	—	—	—	Initialized	HWLIN
LINSTR	Initialized	—	—	—	—	—	Initialized	
LINTCR	Initialized	—	—	—	—	—	Initialized	
LINTCNT	Initialized	—	—	—	—	—	Initialized	
LINTOCNT	Initialized	—	—	—	—	—	Initialized	
CRCCR	Initialized	—	—	—	—	—	Initialized	CRC
CRCDIR	Initialized	—	—	—	—	—	Initialized	
CRCDOR	Initialized	—	—	—	—	—	Initialized	
FMODR	Initialized	—	—	—	—	—	Initialized	FLASH/ EEROM
FASTAT	Initialized	—	—	—	—	—	Initialized	
FAEINT	Initialized	—	—	—	—	—	Initialized	
ROMMAT	Initialized	—	—	—	—	—	Initialized	
EEPRE0	Initialized	—	—	—	—	—	Initialized	
EEPRE1	Initialized	—	—	—	—	—	Initialized	
EEPWE0	Initialized	—	—	—	—	—	Initialized	
EEPWE1	Initialized	—	—	—	—	—	Initialized	
FCURAME	Initialized	—	—	—	—	—	Initialized	
FSTATR0	Initialized	—	—	—	—	—	Initialized	
FSTATR1	Initialized	—	—	—	—	—	Initialized	
FENTRYR	Initialized	—	—	—	—	—	Initialized	
FPROTR	Initialized	—	—	—	—	—	Initialized	
FRESETR	Initialized	—	—	—	—	—	Initialized	
FCMDR	Initialized	—	—	—	—	—	Initialized	
FRAMECCR	Initialized	—	—	—	—	—	Initialized	
FCPSR	Initialized	—	—	—	—	—	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
EEPBCCNT	Initialized	—	—	—	—	—	Initialized	FLASH/ EEROM
FPESTAT	Initialized	—	—	—	—	—	Initialized	
EEPBCSTAT	Initialized	—	—	—	—	—	Initialized	
FCKAR	Initialized	—	—	—	—	—	Initialized	
EEPMAT	Initialized	—	—	—	—	—	Initialized	
RAMEN	Initialized	—	—	—	—	—	Initialized	RAM
RAMWEN	Initialized	—	—	—	—	—	Initialized	
RAMECC	Initialized	—	—	—	—	—	Initialized	
RAMERR	Initialized	—	—	—	—	—	Initialized	
RAMINT	Initialized	—	—	—	—	—	Initialized	
RAMACYC	Initialized	—	—	—	—	—	Initialized	
ADDRA_0	Initialized	—	—	—	—	—	Initialized	A/D_0
ADDRB_0	Initialized	—	—	—	—	—	Initialized	
ADDRC_0	Initialized	—	—	—	—	—	Initialized	
ADDRD_0	Initialized	—	—	—	—	—	Initialized	
ADDRE_0	Initialized	—	—	—	—	—	Initialized	
ADDRF_0	Initialized	—	—	—	—	—	Initialized	
ADDRG_0	Initialized	—	—	—	—	—	Initialized	
ADDRH_0	Initialized	—	—	—	—	—	Initialized	
ADCSR_0	Initialized	—	—	—	—	—	Initialized	
ADCR_0	Initialized	—	—	—	—	—	Initialized	
APPDCR_0	Initialized	—	—	—	—	—	Initialized	
ADDIAGR_0	Initialized	—	—	—	—	—	Initialized	
ADDRA_1	Initialized	—	—	—	—	—	Initialized	A/D_1
ADDRB_1	Initialized	—	—	—	—	—	Initialized	
ADDRC_1	Initialized	—	—	—	—	—	Initialized	
ADDRD_1	Initialized	—	—	—	—	—	Initialized	
ADDRE_1	Initialized	—	—	—	—	—	Initialized	
ADDRF_1	Initialized	—	—	—	—	—	Initialized	
ADDRG_1	Initialized	—	—	—	—	—	Initialized	
ADDRH_1	Initialized	—	—	—	—	—	Initialized	
ADCSR_1	Initialized	—	—	—	—	—	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
ADCR_1	Initialized	—	—	—	—	—	Initialized	A/D_1
APPDCR_1	Initialized	—	—	—	—	—	Initialized	
ADDIAGR_1	Initialized	—	—	—	—	—	Initialized	
TSTRB	Initialized	—	—	—	—	—	Initialized	TPU (unit 1)
TSYRB	Initialized	—	—	—	—	—	Initialized	
TCR_6	Initialized	—	—	—	—	—	Initialized	TPU_6
TMDR_6	Initialized	—	—	—	—	—	Initialized	
TIORH_6	Initialized	—	—	—	—	—	Initialized	
TIORL_6	Initialized	—	—	—	—	—	Initialized	
TIER_6	Initialized	—	—	—	—	—	Initialized	
TSR_6	Initialized	—	—	—	—	—	Initialized	
TCNT_6	Initialized	—	—	—	—	—	Initialized	
TGRA_6	Initialized	—	—	—	—	—	Initialized	
TGRB_6	Initialized	—	—	—	—	—	Initialized	
TGRC_6	Initialized	—	—	—	—	—	Initialized	
TGRD_6	Initialized	—	—	—	—	—	Initialized	
TCR_7	Initialized	—	—	—	—	—	Initialized	TPU_7
TMDR_7	Initialized	—	—	—	—	—	Initialized	
TIOR_7	Initialized	—	—	—	—	—	Initialized	
TIER_7	Initialized	—	—	—	—	—	Initialized	
TSR_7	Initialized	—	—	—	—	—	Initialized	
TCNT_7	Initialized	—	—	—	—	—	Initialized	
TGRA_7	Initialized	—	—	—	—	—	Initialized	
TGRB_7	Initialized	—	—	—	—	—	Initialized	
TCR_8	Initialized	—	—	—	—	—	Initialized	TPU_8
TMDR_8	Initialized	—	—	—	—	—	Initialized	
TIOR_8	Initialized	—	—	—	—	—	Initialized	
TIER_8	Initialized	—	—	—	—	—	Initialized	
TSR_8	Initialized	—	—	—	—	—	Initialized	
TCNT_8	Initialized	—	—	—	—	—	Initialized	
TGRA_8	Initialized	—	—	—	—	—	Initialized	
TGRB_8	Initialized	—	—	—	—	—	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
TCR_9	Initialized	—	—	—	—	—	Initialized	TPU_9
TMDR_9	Initialized	—	—	—	—	—	Initialized	
TIORH_9	Initialized	—	—	—	—	—	Initialized	
TIORL_9	Initialized	—	—	—	—	—	Initialized	
TIER_9	Initialized	—	—	—	—	—	Initialized	
TSR_9	Initialized	—	—	—	—	—	Initialized	
TCNT_9	Initialized	—	—	—	—	—	Initialized	
TGRA_9	Initialized	—	—	—	—	—	Initialized	
TGRB_9	Initialized	—	—	—	—	—	Initialized	
TGRC_9	Initialized	—	—	—	—	—	Initialized	
TGRD_9	Initialized	—	—	—	—	—	Initialized	
TCR_10	Initialized	—	—	—	—	—	Initialized	TPU_10
TMDR_10	Initialized	—	—	—	—	—	Initialized	
TIOR_10	Initialized	—	—	—	—	—	Initialized	
TIER_10	Initialized	—	—	—	—	—	Initialized	
TSR_10	Initialized	—	—	—	—	—	Initialized	
TCNT_10	Initialized	—	—	—	—	—	Initialized	
TGRA_10	Initialized	—	—	—	—	—	Initialized	
TGRB_10	Initialized	—	—	—	—	—	Initialized	
TCR_11	Initialized	—	—	—	—	—	Initialized	TPU_11
TMDR_11	Initialized	—	—	—	—	—	Initialized	
TIOR_11	Initialized	—	—	—	—	—	Initialized	
TIER_11	Initialized	—	—	—	—	—	Initialized	
TSR_11	Initialized	—	—	—	—	—	Initialized	
TCNT_11	Initialized	—	—	—	—	—	Initialized	
TGRA_11	Initialized	—	—	—	—	—	Initialized	
TGRB_11	Initialized	—	—	—	—	—	Initialized	
P1DDR	Initialized	—	—	—	—	—	Initialized	I/O port
P3DDR	Initialized	—	—	—	—	—	Initialized	
P6DDR	Initialized	—	—	—	—	—	Initialized	
PADDR	Initialized	—	—	—	—	—	Initialized	
PDDDR	Initialized	—	—	—	—	—	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
P1ICR	Initialized	—	—	—	—	—	Initialized	I/O port
P3ICR	Initialized	—	—	—	—	—	Initialized	
P4ICR	Initialized	—	—	—	—	—	Initialized	
P5ICR	Initialized	—	—	—	—	—	Initialized	
P6ICR	Initialized	—	—	—	—	—	Initialized	
PAICR	Initialized	—	—	—	—	—	Initialized	
PDICR	Initialized	—	—	—	—	—	Initialized	
PORTH	—	—	—	—	—	—	—	
PORTJ	—	—	—	—	—	—	—	
PORTK	—	—	—	—	—	—	—	
PHDR	Initialized	—	—	—	—	—	Initialized	
PJDR	Initialized	—	—	—	—	—	Initialized	
PKDR	Initialized	—	—	—	—	—	Initialized	
PHDDR	Initialized	—	—	—	—	—	Initialized	
PJDDR	Initialized	—	—	—	—	—	Initialized	
PKDDR	Initialized	—	—	—	—	—	Initialized	
PHICR	Initialized	—	—	—	—	—	Initialized	
PJICR	Initialized	—	—	—	—	—	Initialized	
PKICR	Initialized	—	—	—	—	—	Initialized	
PDPCR	Initialized	—	—	—	—	—	Initialized	
PHPCR	Initialized	—	—	—	—	—	Initialized	
PJPCR	Initialized	—	—	—	—	—	Initialized	
PKPCR	Initialized	—	—	—	—	—	Initialized	
PFCR5	Initialized	—	—	—	—	—	Initialized	
PFCR6	Initialized	—	—	—	—	—	Initialized	
PFCR8	Initialized	—	—	—	—	—	Initialized	
PFCR9	Initialized	—	—	—	—	—	Initialized	
PFCRA	Initialized	—	—	—	—	—	Initialized	
PFCRB	Initialized	—	—	—	—	—	Initialized	
PFCRC	Initialized	—	—	—	—	—	Initialized	
PFCRD	Initialized	—	—	—	—	—	Initialized	
SSIER	Initialized	—	—	—	—	—	Initialized	INTC

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
P1DSR	Initialized	—	—	—	—	—	Initialized	I/O port
P6DSR	Initialized	—	—	—	—	—	Initialized	
PADSR	Initialized	—	—	—	—	—	Initialized	
PDDSR	Initialized	—	—	—	—	—	Initialized	
PHDSR	Initialized	—	—	—	—	—	Initialized	
P1PSR	Initialized	—	—	—	—	—	Initialized	
P6PSR	Initialized	—	—	—	—	—	Initialized	
PAPSR	Initialized	—	—	—	—	—	Initialized	
PDPSR	Initialized	—	—	—	—	—	Initialized	
PHPSR	Initialized	—	—	—	—	—	Initialized	
DSAR_0	Initialized	—	—	—	—	—	Initialized	DMAC_0
DDAR_0	Initialized	—	—	—	—	—	Initialized	
DOFR_0	Initialized	—	—	—	—	—	Initialized	
DTCR_0	Initialized	—	—	—	—	—	Initialized	
DBSR_0	Initialized	—	—	—	—	—	Initialized	
DMDR_0	Initialized	—	—	—	—	—	Initialized	
DACR_0	Initialized	—	—	—	—	—	Initialized	
DSAR_1	Initialized	—	—	—	—	—	Initialized	DMAC_1
DDAR_1	Initialized	—	—	—	—	—	Initialized	
DOFR_1	Initialized	—	—	—	—	—	Initialized	
DTCR_1	Initialized	—	—	—	—	—	Initialized	
DBSR_1	Initialized	—	—	—	—	—	Initialized	
DMDR_1	Initialized	—	—	—	—	—	Initialized	
DACR_1	Initialized	—	—	—	—	—	Initialized	
DSAR_2	Initialized	—	—	—	—	—	Initialized	DMAC_2
DDAR_2	Initialized	—	—	—	—	—	Initialized	
DOFR_2	Initialized	—	—	—	—	—	Initialized	
DTCR_2	Initialized	—	—	—	—	—	Initialized	
DBSR_2	Initialized	—	—	—	—	—	Initialized	
DMDR_2	Initialized	—	—	—	—	—	Initialized	
DACR_2	Initialized	—	—	—	—	—	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
DSAR_3	Initialized	—	—	—	—	—	Initialized	DMAC_3
DDAR_3	Initialized	—	—	—	—	—	Initialized	
DOFR_3	Initialized	—	—	—	—	—	Initialized	
DTCR_3	Initialized	—	—	—	—	—	Initialized	
DBSR_3	Initialized	—	—	—	—	—	Initialized	
DMDR_3	Initialized	—	—	—	—	—	Initialized	
DACR_3	Initialized	—	—	—	—	—	Initialized	
DMRSR_0	Initialized	—	—	—	—	—	Initialized	DMAC_0
DMRSR_1	Initialized	—	—	—	—	—	Initialized	DMAC_1
DMRSR_2	Initialized	—	—	—	—	—	Initialized	DMAC_2
DMRSR_3	Initialized	—	—	—	—	—	Initialized	DMAC_3
IPRA	Initialized	—	—	—	—	—	Initialized	INTC
IPRB	Initialized	—	—	—	—	—	Initialized	
IPRC	Initialized	—	—	—	—	—	Initialized	
IPRD	Initialized	—	—	—	—	—	Initialized	
IPRE	Initialized	—	—	—	—	—	Initialized	
IPRF	Initialized	—	—	—	—	—	Initialized	
IPRG	Initialized	—	—	—	—	—	Initialized	
IPRI	Initialized	—	—	—	—	—	Initialized	
IPRJ	Initialized	—	—	—	—	—	Initialized	
IPRL	Initialized	—	—	—	—	—	Initialized	
IPRM	Initialized	—	—	—	—	—	Initialized	
IPRN	Initialized	—	—	—	—	—	Initialized	
IPRO	Initialized	—	—	—	—	—	Initialized	
IPRP	Initialized	—	—	—	—	—	Initialized	
ISCRH	Initialized	—	—	—	—	—	Initialized	
ISCRL	Initialized	—	—	—	—	—	Initialized	
DTCVBR	Initialized	—	—	—	—	—	Initialized	BSC
BCR2	Initialized	—	—	—	—	—	Initialized	
MDCR	Initialized	—	—	—	—	—	Initialized	SYSTEM
SYSCR0	Initialized	—	—	—	—	—	Initialized	
SCKCR0	Initialized	—	—	—	—	—	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
SBYCR	Initialized	—	—	—	—	—	Initialized	SYSTEM
MSTPCRA	Initialized	—	—	—	—	—	Initialized	
MSTPCRB	Initialized	—	—	—	—	—	Initialized	
MSTPCRC	Initialized	—	—	—	—	—	Initialized	
ROSCCR	Initialized	—	—	—	—	—	Initialized	
SYSCR1	Initialized	—	—	—	—	—	Initialized	
SCKCR1	Initialized	—	—	—	—	—	Initialized	
MSTPCRD	Initialized	—	—	—	—	—	Initialized	
MSTPCRE	Initialized	—	—	—	—	—	Initialized	
SMR_3	Initialized	—	—	—	—	—	Initialized	SCI_3
BRR_3	Initialized	—	—	—	—	—	Initialized	
SCR_3	Initialized	—	—	—	—	—	Initialized	
TDR_3	Initialized	—	—	—	—	—	Initialized	
SSR_3	Initialized	—	—	—	—	—	Initialized	
RDR_3	Initialized	—	—	—	—	—	Initialized	
SMR_4	Initialized	—	—	—	—	—	Initialized	SCI_4
BRR_4	Initialized	—	—	—	—	—	Initialized	
SCR_4	Initialized	—	—	—	—	—	Initialized	
TDR_4	Initialized	—	—	—	—	—	Initialized	
SSR_4	Initialized	—	—	—	—	—	Initialized	
RDR_4	Initialized	—	—	—	—	—	Initialized	
TCR_4	Initialized	—	—	—	—	—	Initialized	TPU_4
TMDR_4	Initialized	—	—	—	—	—	Initialized	
TIOR_4	Initialized	—	—	—	—	—	Initialized	
TIER_4	Initialized	—	—	—	—	—	Initialized	
TSR_4	Initialized	—	—	—	—	—	Initialized	
TCNT_4	Initialized	—	—	—	—	—	Initialized	
TGRA_4	Initialized	—	—	—	—	—	Initialized	
TGRB_4	Initialized	—	—	—	—	—	Initialized	
TCR_5	Initialized	—	—	—	—	—	Initialized	TPU_5
TMDR_5	Initialized	—	—	—	—	—	Initialized	
TIOR_5	Initialized	—	—	—	—	—	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
TIER_5	Initialized	—	—	—	—	—	Initialized	TPU_5
TSR_5	Initialized	—	—	—	—	—	Initialized	
TCNT_5	Initialized	—	—	—	—	—	Initialized	
TGRA_5	Initialized	—	—	—	—	—	Initialized	
TGRB_5	Initialized	—	—	—	—	—	Initialized	
DTCERA	Initialized	—	—	—	—	—	Initialized	DTC
DTCERB	Initialized	—	—	—	—	—	Initialized	
DTCERC	Initialized	—	—	—	—	—	Initialized	
DTCERD	Initialized	—	—	—	—	—	Initialized	
DTCERE	Initialized	—	—	—	—	—	Initialized	
DTCERF	Initialized	—	—	—	—	—	Initialized	
DTCERG	Initialized	—	—	—	—	—	Initialized	
DTCCR	Initialized	—	—	—	—	—	Initialized	
INTCR	Initialized	—	—	—	—	—	Initialized	INTC
CPUPCR	Initialized	—	—	—	—	—	Initialized	
IER	Initialized	—	—	—	—	—	Initialized	
ISR	Initialized	—	—	—	—	—	Initialized	
PORT1	—	—	—	—	—	—	—	I/O port
PORT3	—	—	—	—	—	—	—	
PORT4	—	—	—	—	—	—	—	
PORT5	—	—	—	—	—	—	—	
PORT6	—	—	—	—	—	—	—	
PORTA	—	—	—	—	—	—	—	
PORTD	—	—	—	—	—	—	—	
P1DR	Initialized	—	—	—	—	—	Initialized	
P3DR	Initialized	—	—	—	—	—	Initialized	
P6DR	Initialized	—	—	—	—	—	Initialized	
PADR	Initialized	—	—	—	—	—	Initialized	
PDDR	Initialized	—	—	—	—	—	Initialized	
PCR	Initialized	—	—	—	—	—	Initialized	PPG
PMR	Initialized	—	—	—	—	—	Initialized	
NDERH	Initialized	—	—	—	—	—	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
NDERL	Initialized	—	—	—	—	—	Initialized	PPG
PODRH	Initialized	—	—	—	—	—	Initialized	
PODRL	Initialized	—	—	—	—	—	Initialized	
NDRH	Initialized	—	—	—	—	—	Initialized	
NDRL	Initialized	—	—	—	—	—	Initialized	
TCSR	Initialized	—	—	—	—	—	Initialized	WDT
TCNT	Initialized	—	—	—	—	—	Initialized	
RSTCSR	Initialized	—	—	—	—	—	Initialized	
TSTR	Initialized	—	—	—	—	—	Initialized	TPU (unit 0)
TSYR	Initialized	—	—	—	—	—	Initialized	
TCR_0	Initialized	—	—	—	—	—	Initialized	TPU_0
TMDR_0	Initialized	—	—	—	—	—	Initialized	
TIORH_0	Initialized	—	—	—	—	—	Initialized	
TIORL_0	Initialized	—	—	—	—	—	Initialized	
TIER_0	Initialized	—	—	—	—	—	Initialized	
TSR_0	Initialized	—	—	—	—	—	Initialized	
TCNT_0	Initialized	—	—	—	—	—	Initialized	
TGRA_0	Initialized	—	—	—	—	—	Initialized	
TGRB_0	Initialized	—	—	—	—	—	Initialized	
TGRC_0	Initialized	—	—	—	—	—	Initialized	
TGRD_0	Initialized	—	—	—	—	—	Initialized	
TCR_1	Initialized	—	—	—	—	—	Initialized	TPU_1
TMDR_1	Initialized	—	—	—	—	—	Initialized	
TIOR_1	Initialized	—	—	—	—	—	Initialized	
TIER_1	Initialized	—	—	—	—	—	Initialized	
TSR_1	Initialized	—	—	—	—	—	Initialized	
TCNT_1	Initialized	—	—	—	—	—	Initialized	
TGRA_1	Initialized	—	—	—	—	—	Initialized	
TGRB_1	Initialized	—	—	—	—	—	Initialized	
TCR_2	Initialized	—	—	—	—	—	Initialized	TPU_2
TMDR_2	Initialized	—	—	—	—	—	Initialized	
TIOR_2	Initialized	—	—	—	—	—	Initialized	

Register Abbreviation	Reset	Sleep Mode	Subclock Mode*	Module Stop Mode	All-Module-Clock-Stop	Software Standby	Hardware Standby*	Module
TIER_2	Initialized	—	—	—	—	—	Initialized	TPU_2
TSR_2	Initialized	—	—	—	—	—	Initialized	
TCNT_2	Initialized	—	—	—	—	—	Initialized	
TGRA_2	Initialized	—	—	—	—	—	Initialized	
TGRB_2	Initialized	—	—	—	—	—	Initialized	
TCR_3	Initialized	—	—	—	—	—	Initialized	TPU_3
TMDR_3	Initialized	—	—	—	—	—	Initialized	
TIORH_3	Initialized	—	—	—	—	—	Initialized	
TIORL_3	Initialized	—	—	—	—	—	Initialized	
TIER_3	Initialized	—	—	—	—	—	Initialized	
TSR_3	Initialized	—	—	—	—	—	Initialized	
TCNT_3	Initialized	—	—	—	—	—	Initialized	
TGRA_3	Initialized	—	—	—	—	—	Initialized	
TGRB_3	Initialized	—	—	—	—	—	Initialized	
TGRC_3	Initialized	—	—	—	—	—	Initialized	
TGRD_3	Initialized	—	—	—	—	—	Initialized	

Note: * This LSI does not have hardware standby mode and subclock mode.

Section 25 Electrical Characteristics

25.1 Absolute Maximum Ratings

Table 25.1 Absolute Maximum Ratings

Item	Symbol	Value	Unit
Power supply voltage	Vcc	–0.3 to +7.0	V
Input voltage (except ports 4 and 5)	V_{in}	–0.3 to Vcc + 0.3	V
Input voltage (port 4)	V_{in}	–0.3 to AVcc1 + 0.3	V
Input voltage (port 5)	V_{in}	–0.3 to AVcc0 + 0.3	V
Analog power supply voltage	AVcc0	–0.3 to +7.0	V
	AVcc1	–0.3 to +7.0	V
Analog input voltage (port 4)	V_{AN}	–0.3 to AVcc1 + 0.3	V
Analog input voltage (port 5)	V_{AN}	–0.3 to AVcc0 + 0.3	V
Operating temperature	Topr	–40 to +85	°C
Storage temperature	Tstg	–55 to +125	°C

Caution: Permanent damage to the LSI may result if absolute maximum ratings are exceeded.

25.2 DC Characteristics

Table 25.2 DC Characteristics (1)

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V,
Vss = AVss = 0 V*[1],
Ta = –40 °C to +85 °C

	Item	Symbol	Min.	Typ.	Max.	Unit	Test Conditions
Schmitt trigger input voltage	$\overline{IRQ}$ input pin, TPU input pin, ports 3, J, and K	VT^-	Vcc × 0.2	—	—	V	
		VT^+	—	—	Vcc × 0.7		
		$VT^+ - VT^-$	0.4	—	—		
Input high voltage (except Schmitt trigger input pin)	EMLE, MD, $\overline{RES}$, NMI	V_{IH}	Vcc – 0.7	—	Vcc + 0.3	V	
	EXTAL		Vcc × 0.7	—	Vcc + 0.3		
	Other input pins		Vcc × 0.7	—	Vcc + 0.3		
	Port 4		AVcc1 × 0.7	—	AVcc1 + 0.3		
	Port 5		AVcc0 × 0.7	—	AVcc0 + 0.3		
Input low voltage (except Schmitt trigger input pin)	EMLE, $\overline{RES}$, MD, NMI	V_{IL}	–0.3	—	Vcc × 0.1	V	
	EXTAL		–0.3	—	Vcc × 0.2		
	Other pins		–0.3	—	Vcc × 0.2		
	Port 4		–0.3	—	AVcc1 × 0.2		
	Port 5		–0.3	—	AVcc0 × 0.2		
Output high voltage	All output pins	V_{OH}	Vcc – 0.5	—	—	V	I_{OH} = –200 µA
			Vcc – 1.0	—	—		I_{OH} = –1 mA
Output low voltage	All output pins	V_{OL}	—	—	0.4	V	I_{OL} = 1.6 mA
Input leakage current	EMLE, $\overline{RES}$, NMI, MD	| Iin |	—	—	1.0	µA	Vin = 0.5 to Vcc – 0.5 V
	Port 4		—	—	1.0		Vin = 0.5 to AVcc1 – 0.5 V
	Port 5		—	—	1.0		Vin = 0.5 to AVcc0 – 0.5 V

Table 25.2 DC Characteristics (2)

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V,
Vss = AVss = 0 V*[1],
Ta = –40 °C to +85 °C

Item		Symbol	Min.	Typ.	Max.	Unit	Test Conditions
Tri-state leakage current (off state)	Ports 1, 6, A, D, H, J, and K	$\lvert I_{TSI} \rvert$	—	—	1.0	µA	Vin = 0.5 to Vcc – 0.5 V
Input pull-up MOS current	Ports D, H, J, and K	–Ip	50	—	300	µA	Vin = 0 V
Input pull-down MOS current	Ports 1, 6, A, D, and H	Ip	45	—	300	µA	Vin = Vcc
Analog port pull-down MOS current	Ports 4 and 5	Ip	5	—	50	µA	Vin = AVcc0, Avcc1
Input capacitance	All input pins	Cin	—	—	15	pF	Vin = 0 V f = 1 MHz Ta = 25 °C
Supply current*[2]	Normal operation	Icc*[4]	—	50	70	mA	Max condition Vcc = 5.5 V EXTAL 10 MHz Iϕ = 80 MHz Rϕ = Pϕ = Fϕ = Aϕ = 40 MHz Bϕ = 20 MHz

Item		Symbol	Min.	Typ.	Max.	Unit	Test Conditions
Supply current*2	Sleep mode	Icc*4	—	38	55	mA	Max condition Vcc = 5.5 V EXTAL 10 MHz Iϕ = 80 MHz Rϕ = Pϕ = Fϕ = Aϕ = 40 MHz Bϕ = 20 MHz
	Standby mode*3		—	10	25	mA	Ta ≦ 50 °C
			—	—	25	mA	50 °C < Ta
	All-module-clock-stop mode*5		—	36	45		
Analog power supply current	During A/D conversion	AIcc0	—	3.5	5	mA	AVcc0 = 5.0 V
	Standby for A/D conversion		—	10	100	μA	
	During A/D conversion	AIcc1	—	3.5	5	mA	AVcc1 = 5.0 V
	Standby for A/D conversion		—	10	100	μA	
RAM standby voltage		V_{RAM}	3.0	—	—	V	
Vcc start voltage	Vcc pin	VccSTART	0	—	0.2	V	
Vcc rising gradient		SVcc	—	—	0.01	V/us	

Notes: 1. When the A/D converter is not used, the AVcc0, AVcc1, and AVss pins should not be open. Connect the AVcc0 and AVcc1 pins to Vcc, and the AVss pin to Vss.

2. Supply current values are for V_{IH} = AVcc0 (port 5), AVcc1 (port 4), Vcc (others) and V_{IL} = 0 V with all output pins unloaded and all input pull-up MOSs in the off state.

3. The values are for $V_{RAM} \leq Vcc < 4.5$ V, V_{IH}min. = Vcc − 0.1 V, and V_{IL}max. = 0.1 V.

4. Icc depends on Vcc and f as follows:

   Iccmax = 21 (mA) + 0.26 (mA/MHz) × f + 5.0 (mA/V) × Vcc (normal operation)

   Iccmax = 21 (mA) + 0.35 (mA/MHz) × f + 1.0 (mA/V) × Vcc (sleep mode)

5. The values are for reference.

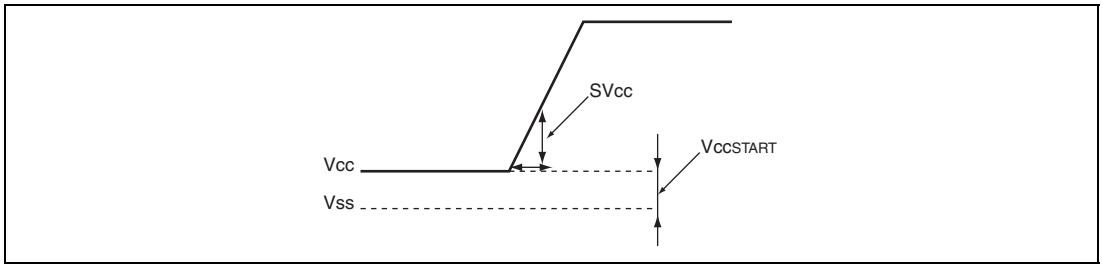


Figure 25.1 Rising of Vcc Power Supply

Table 25.3 Permissible Output Currents

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V,
Vss = AVss = 0 V*
Ta = –40 °C to +85 °C

Item		Symbol	Min.	Typ.	Max.	Unit	Test Conditions
Permissible output low current (per pin)	All output pins	I_{OL}	—	—	10	mA	
Permissible output low current (total)	All output pins	ΣI_{OL}	—	—	120	mA	
Permissible output high current (per pin)	All output pins	$-I_{OH}$	—	—	2.0	mA	
Permissible output high current (total)	All output pins	$-\Sigma I_{OH}$	—	—	40	mA	

Caution: To ensure the LSI's reliability, do not exceed the output current values in table 25.3.

Note: * When the A/D converter is not used, the AVcc0, AVcc1, and AVss pins should not be open. Connect the AVcc0 and AVcc1 pins to Vcc, and the AVss pin to Vss.

25.3 AC Characteristics

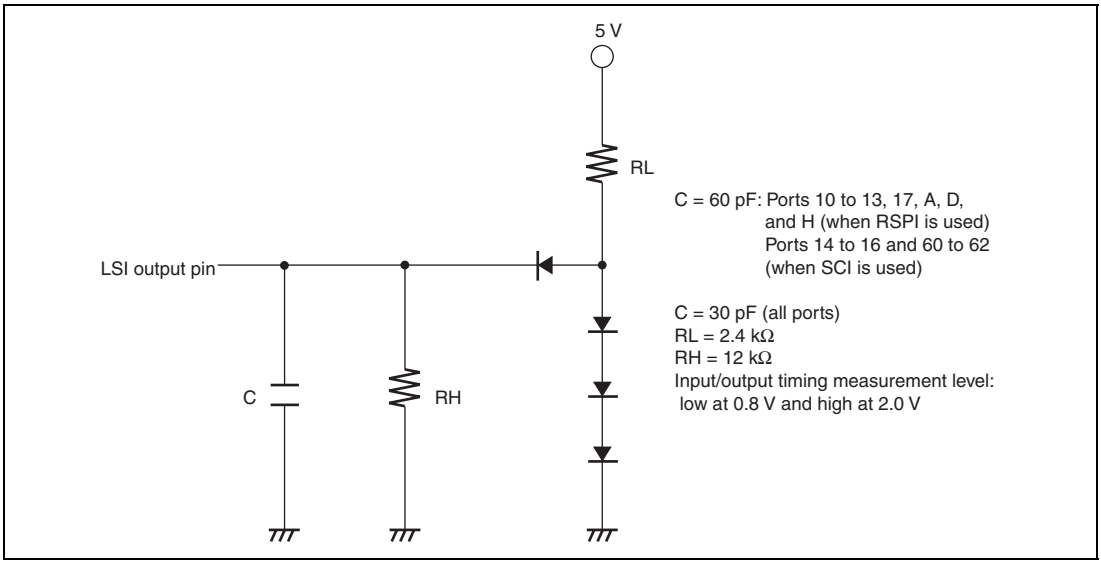


Figure 25.2 Output Load Circuit

25.3.1 Clock Timing

Table 25.4 shows the clock timing.

Table 25.4 Clock Timing

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V,
Vss = AVss = 0 V
Ta = –40 to +85 °C

Item	Symbol	Min.	Max.	Unit.	Reference
External input clock frequency	f_{EX}	8	10	MHz	Figure 25.3
External input clock cycle time	t_{EXcyc}	100	125	ns	
External input clock high-level pulse width	t_{EXH}	40		ns	
External input clock low-level pulse width	t_{EXL}	40	—	ns	
External input clock rising time	t_{EXr}	—	5	ns	
External input clock falling time	t_{EXf}	—	5	ns	
Oscillation settling time after reset (crystal)	t_{OSC1}	20	—	ms	Figure 25.5
Oscillation settling time after leaving software standby mode (crystal)	t_{OSC2}	10	—	ms	Figure 25.4
Oscillation settling time after external clock	t_{DEXT}	2	—	ms	Figure 25.5
System clock (Iϕ) frequency	fI	8	80	MHz	Figure 25.6
System clock (Iϕ) cycle time	t_{Icyc}	12.5	125	ns	
Peripheral module clock (Pϕ) frequency	f_P	8	40	MHz	Figure 25.7
Peripheral module clock (Pϕ) cycle time	t_{Pcyc}	25	125	ns	
External bus clock (Bϕ) frequency	f_{op}	8	20	MHz	Figure 25.8
External bus clock (Bϕ) cycle time	t_{cyc}	50	125	ns	
External bus clock (Bϕ) high-level pulse width	t_{CH}	15	—	ns	
External bus clock (Bϕ) low-level pulse width	t_{CL}	15	—	ns	
External bus clock (Bϕ) rising time	t_{Cr}	—	10	ns	
External bus clock (Bϕ) falling time	t_{Cf}	—	10	ns	
FLASH clock (Fϕ) frequency	f_F	8	40	MHz	Figure 25.9
FLASH clock (Fϕ) cycle time	t_{Fcyc}	25	125	ns	
A/D clock (Aϕ) frequency	f_A	8	40	MHz	Figure 25.10
A/D clock (Aϕ) cycle time	t_{Acyc}	25	125	ns	

Item	Symbol	Min.	Max.	Unit.	Reference
RSPI clock (Rϕ) frequency	f_R	8	40	MHz	Figure 25.11
RSPI clock (Rϕ) cycle time	t_{Rcyc}	25	125	ns	

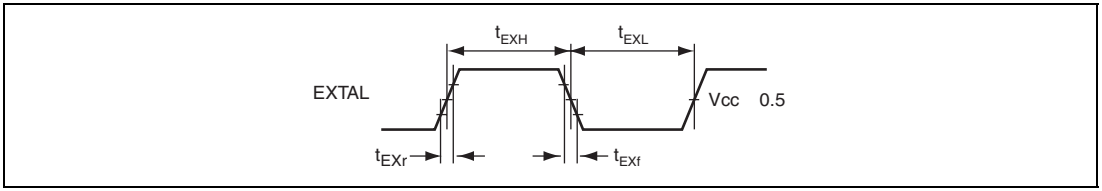


Figure 25.3 External Input Clock Timing

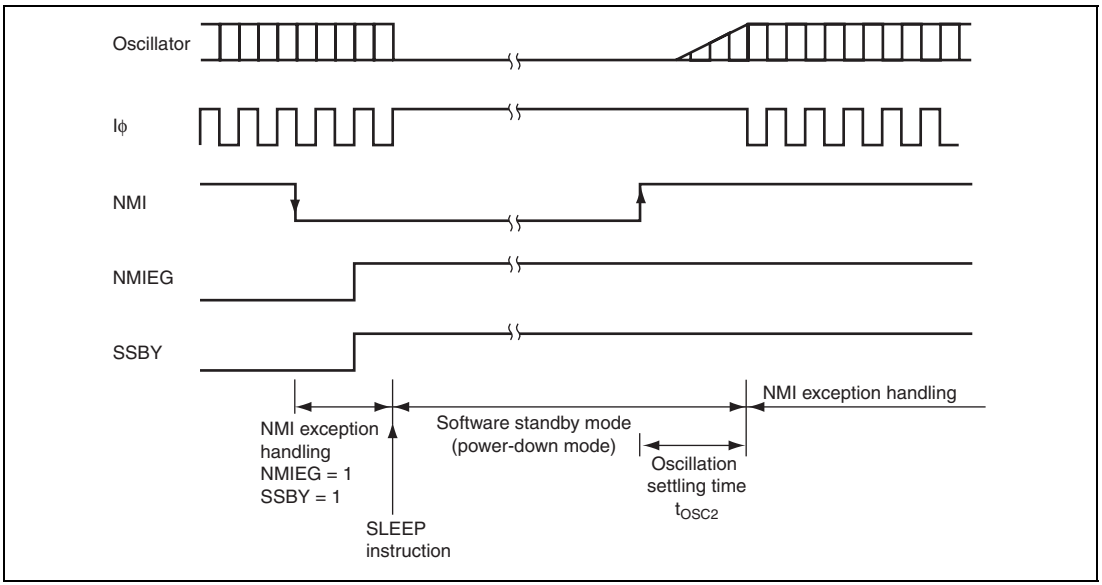


Figure 25.4 Oscillation Settling Timing after Software Standby Mode

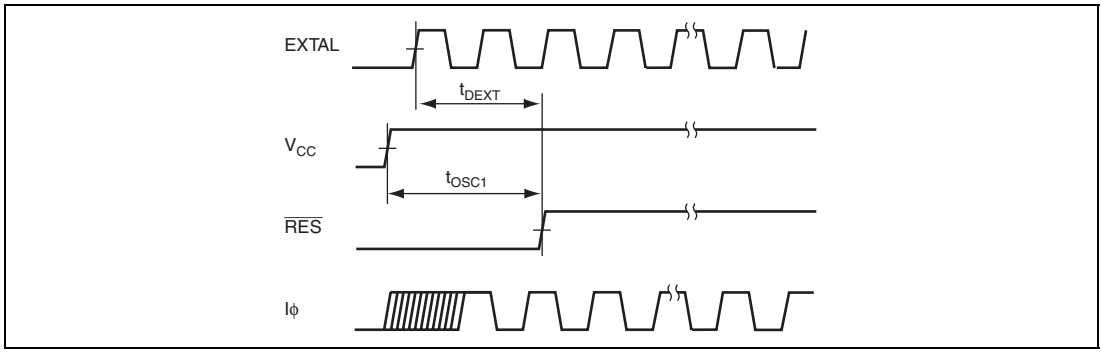


Figure 25.5 Oscillation Settling Timing

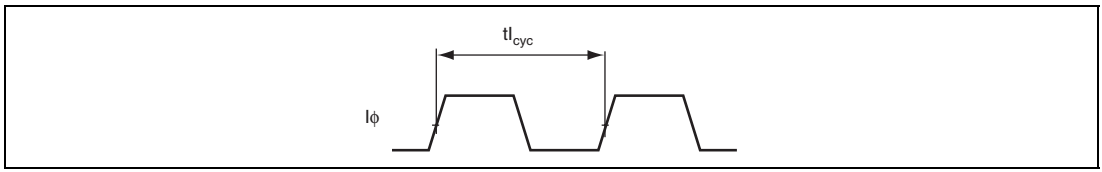


Figure 25.6 System Clock Timing

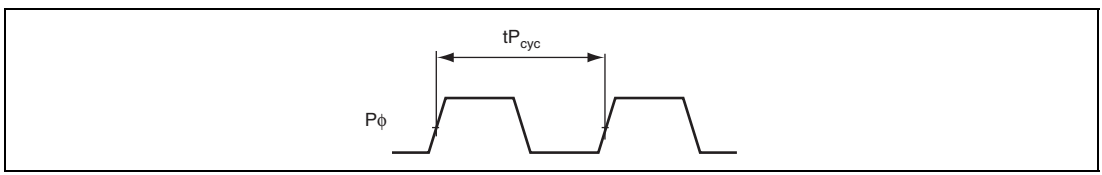


Figure 25.7 Peripheral Module Clock Timing

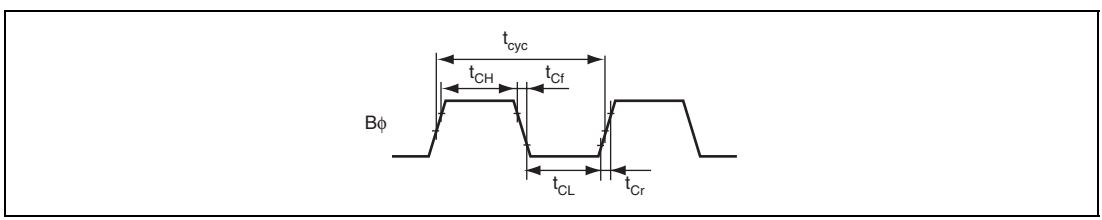


Figure 25.8 External Bus Clock Timing

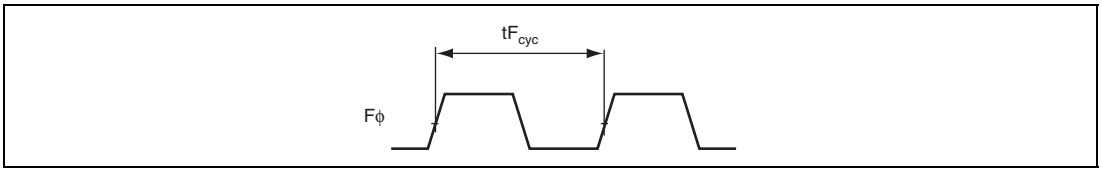


Figure 25.9 FLASH Clock Timing

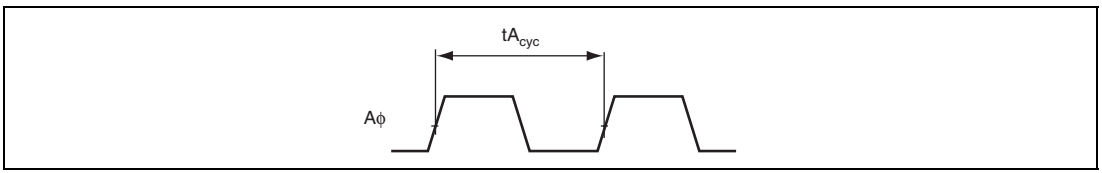


Figure 25.10 A/D Clock Timing

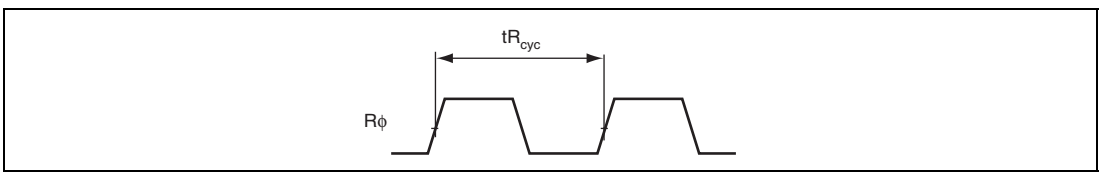


Figure 25.11 RSPI Clock Timing

25.3.2 Control Signal Timing

Table 25.5 Control Signal Timing

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V,
Vss = AVss = 0 V, Iϕ = 8 MHz to 80 MHz
Ta = –40 to +85 °C

Item	Symbol	Min.	Max.	Unit	Reference
$\overline{\text{RES}}$ setup time	t_{RESS}	1000	—	ns	Figure 25.12
$\overline{\text{RES}}$ pulse width (except for flash memory programming/erasing)	t_{RESW}	100	—	tI_{cyc}	
$\overline{\text{RES}}$ pulse width (during flash memory programming/erasing)*	t_{RESW2}	20	—	µs	
NMI setup time	t_{NMIS}	150	—	ns	Figure 25.13
NMI hold time	t_{NMIH}	10	—	ns	
NMI pulse width (after leaving software standby mode)	t_{NMIW}	200	—	ns	
$\overline{\text{IRQ}}$ setup time	t_{IRQS}	150	—	ns	
$\overline{\text{IRQ}}$ hold time	t_{IRQH}	10	—	ns	
$\overline{\text{IRQ}}$ pulse width (after leaving software standby mode)	t_{IRQW}	200	—	ns	

Note: * Waiting time for discharging the high voltage in the LSI which is generated during programming or erasing of the internal flash memory.

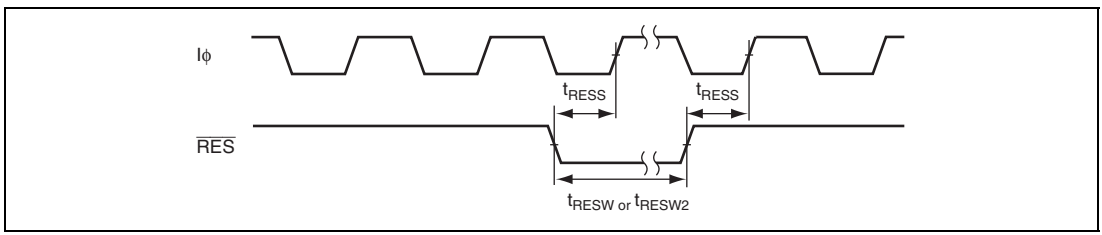


Figure 25.12 Reset Input Timing

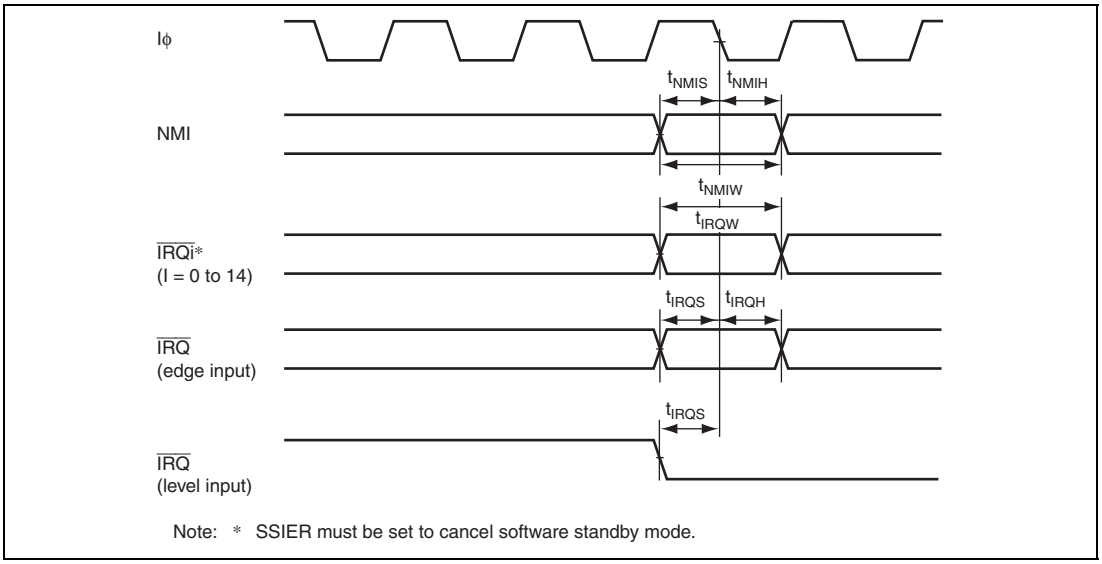


Figure 25.13 Interrupt Input Timing

25.3.3 Timing of On-Chip Peripheral Modules

Table 25.6 Timing of On-Chip Peripheral Modules (1)

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V,
Vss = AVss = 0 V,
Pϕ = 8 MHz to 40 MHz, Aϕ = 8 MHz to 40 MHz
Ta = –40 °C to +85 °C

Item			Symbol	Min.	Max.	Unit	Reference
I/O ports	Output data delay time	High*1	t_{PWD}	—	40	ns	Figure 25.14
		Low*2		—	90	ns	
	Input data setup time		t_{PRS}	25	—	ns	
	Input data hold time		t_{PRH}	25	—	ns	
TPU	Timer output delay time	High*1	t_{TOCD}	—	40	ns	Figure 25.15
		Low*2		—	90	ns	
	Timer input setup time		t_{TICS}	25	—	ns	
	Timer clock input setup time		t_{TCKS}	25	—	ns	Figure 25.16
	Timer clock pulse width	Single-edge setting	t_{TCKWH}	1.5	—	t_{pcyc}	
		Both-edge setting	t_{TCKWL}	2.5	—	t_{pcyc}	
PPG	Pulse output delay time	High*1	t_{POD}	—	40	ns	Figure 25.17
		Low*2		—	90	ns	
SCI	Input clock cycle	Asynchronous	t_{Scyc}	4	—	t_{pcyc}	Figure 25.18
		Clocked synchronous		6	—		
	Input clock pulse width		t_{SCKW}	0.4	0.6	t_{Scyc}	
	Input clock rising time		t_{SCKr}	—	20	ns	Figure 25.18
	Input clock falling time		t_{SCKf}	—	20	ns	Voltage reference level Vcc × 0.3 V to Vcc × 0.7 V
	Output clock cycle	Asynchronous	t_{Scyc}	30	—	t_{pcyc}	Figure 25.18
		Clocked synchronous		4	—		
	Output clock pulse width		t_{SCKW}	0.4	0.6	t_{Scyc}	

Item			Symbol	Min.	Max.	Unit	Reference
SCI	Output clock rising time	High*1	t_{SCKr}	—	20	ns	Figure 25.18
		Low*2		—	40	ns	Voltage reference level: Vcc × 0.3 V to Vcc × 0.7 V
	Output clock falling time	High*1	t_{SCKf}	—	20	ns	
		Low*2		—	40	ns	
	Transmit data delay time	High*1	t_{TXD}	—	40	ns	Figure 25.19
		Low*2		—	90	ns	
	Receive data setup time (clocked synchronous)		t_{RXS}	30	—	ns	
	Receive data hold time (clocked synchronous)		t_{RXH}	30	—	ns	
A/D converter	Trigger input setup time		t_{TRGS}	30	—	ns	Figure 25.20

Notes: 1. I/O port driving ability: High.
2. I/O port driving ability: Low.

Table 25.6 Timing of On-Chip Peripheral Modules (2)

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V,
Vss = AVss = 0 V, Pϕ = 16 MHz to 40 MHz,
Ta = –40 to +85 °C

Item			Symbol	Min.	Max.	Unit	Reference
RACN-TL1*1	Transmit data delay time	High*2	t_{CTXD}	—	100	ns	Figure 25.21
		Low*3		—	200	ns	
	Receive data setup time		t_{CRXS}	100	—	ns	
	Receive data hold time		t_{CRXH}	100	—	ns	

Notes: 1. Although the RCAN-TL1 input/output signals are asynchronous, these are synchronized at the rising edge of Pϕ as shown in figure 25.21.
2. I/O port driving ability: High.
3. I/O port driving ability: Low.

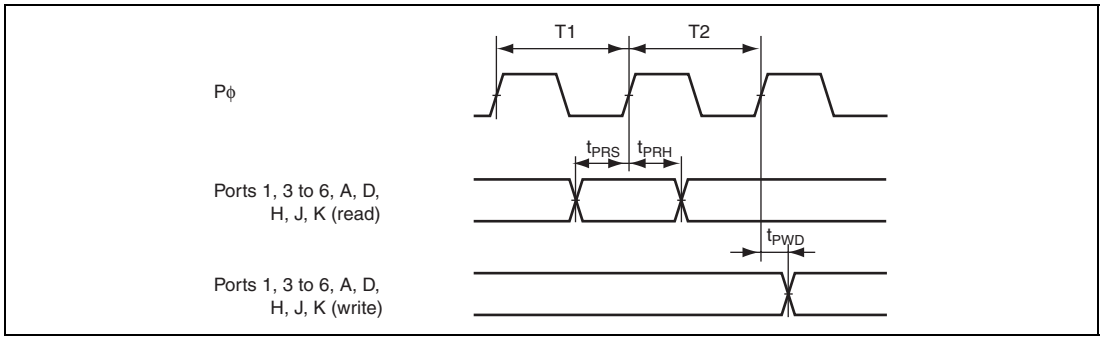


Figure 25.14 I/O Port Input/Output Timing

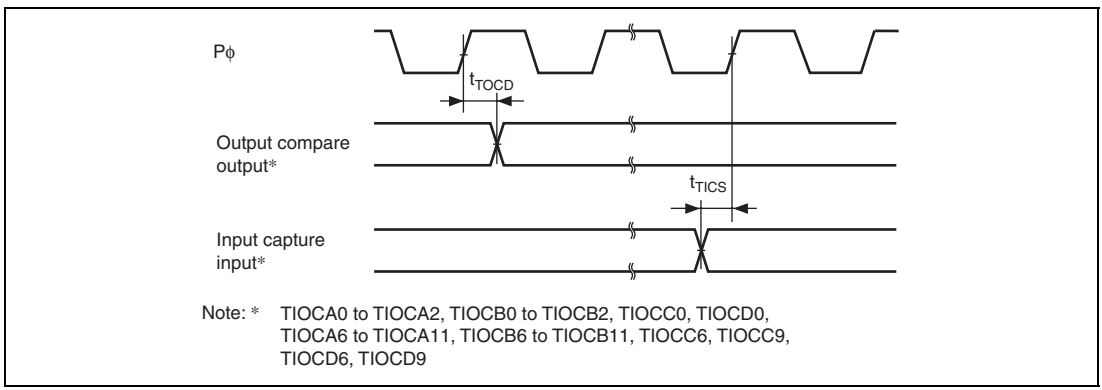


Figure 25.15 TPU Input/Output Timing

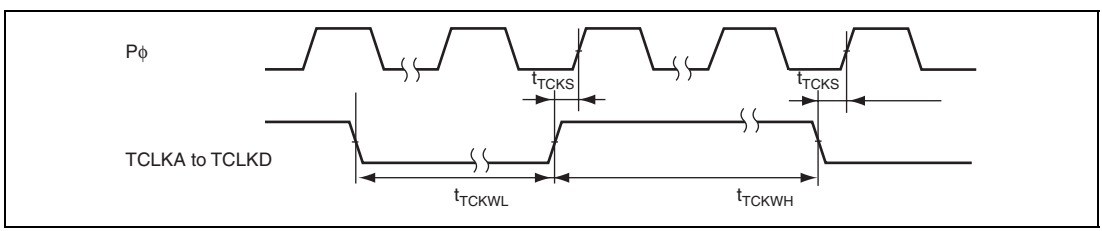


Figure 25.16 TPU Clock Input Timing

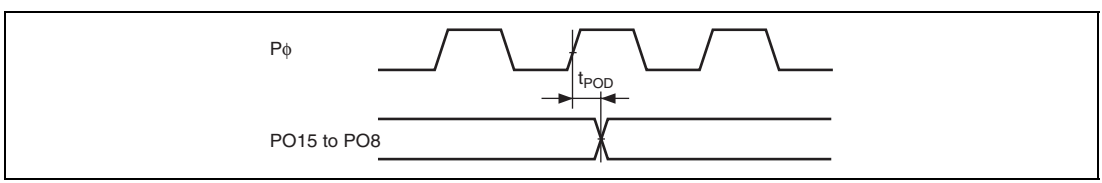


Figure 25.17 PPG Output Timing

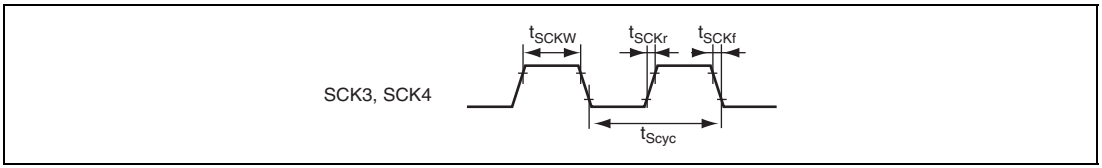


Figure 25.18 SCK Clock Input/Output Timing

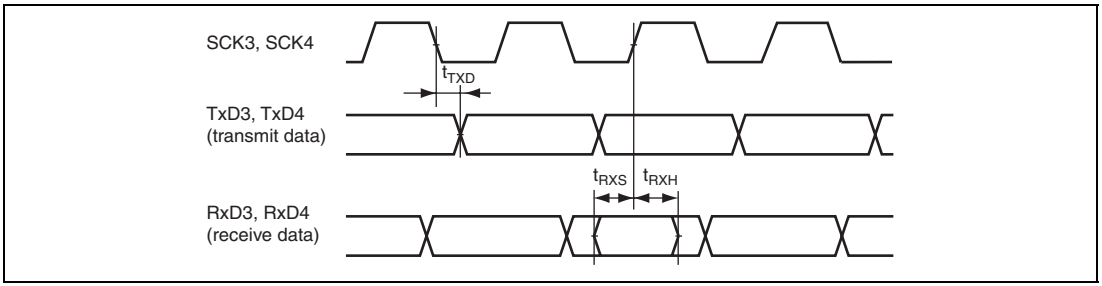


Figure 25.19 SCI Input/Output Timing: Clocked Synchronous Mode

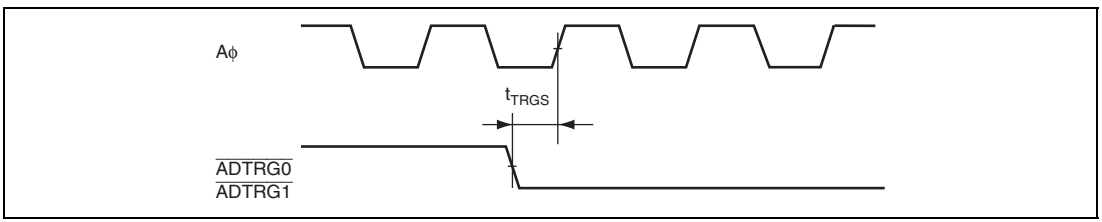


Figure 25.20 A/D Converter External Trigger Input Timing

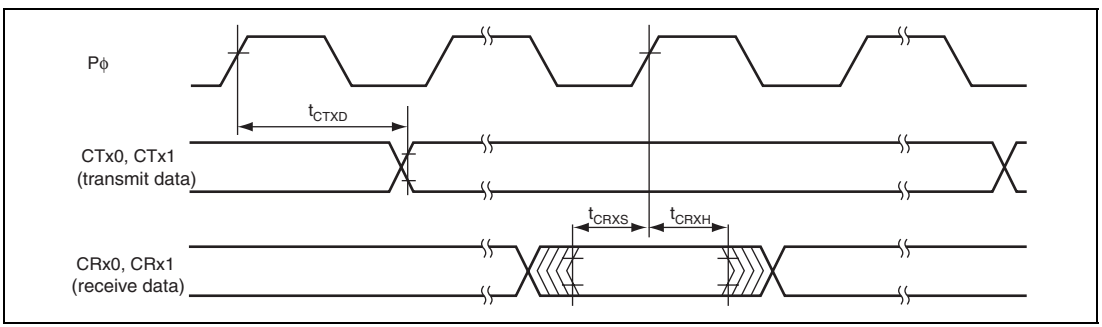


Figure 25.21 RCAN-TL1 Input/Output Timing

Table 25.7 RSPI Timing

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V,
Vss = AVss = 0 V, Rϕ = 8 MHz to 40 MHz
Ta = –40 to +85 °C

Item			Symbol	Min.	Typ.	Max.	Unit	Test Conditions
RSPCK clock cycle		Master	tSPcyc	4	—	4096	tRcyc	Figure 25.22
		Slave		8	—	4096		
RSPCK clock high-level pulse width		Master	tSPCKWH	(tSPcyc – tSPCKR – tSPCKF) /2 – 3	—	—	ns	
		Slave		(tSPcyc – tSPCKR – tSPCKF) /2	—	—		
RSPCK clock low-level pulse width		Master	tSPCKWL	(tSPcyc – tSPCKR – tSPCKF) /2 – 3	—	—	ns	
		Slave		(tSPcyc – tSPCKR – tSPCKF) /2	—	—		
RSPCK clock rising/falling time	Output	High*1	tSPCKR, tSPCKF	—	3	5	ns	
		Low*2		—	20	40		
	Input			—	—	1	µs	
Data input setup time	Master	High*1	tSU	25	—	—	ns	Figures 25.23 to 25.26
		Low*2		75	—	—		
	Slave			20-2 × tRcyc	—	—		
Data input hold time		Master	tH	0	—	—	ns	
		Slave		20	—	—		
SSL setup time		Master	tLEAD	1	—	8	tSPcyc	
		Slave		4	—	—	tRcyc	
SSL hold time		Master	tLAG	1	—	8	tSPcyc	
		Slave		4	—	—	tRcyc	
Data output delay time	Master	High*1	tOD	—	—	25	ns	
		Low*2		—	—	75	ns	
	Slave	High*1		—	—	3 × tRcyc + 40	ns	
		Low*2		—	—	3 × tRcyc + 90		
Data output hold time		Master	tOH	0	—	—	ns	
		Slave		0	—	—		

Item			Symbol	Min.	Typ.	Max.	Unit	Test Conditions
Continuous transmit delay time		Master	tTD	tSPcyc + 2 × tRcyc	—	8 × tSPcyc + 2 × tRcyc	ns	Figures 25.23 to 25.26
		Slave		4 × tRcyc	—	—		
MOSI and MISO rising/falling time	Output	High*1	tDR, tDF	—	—	5	ns	
		Low*2		—	—	55		
	Input			—	—	1	µs	
SSL rising/falling time	Output	High*1	tSSLR, tSSLF	—	—	5	ns	
		Low*2		—	—	55		
	Input			—	—	1	µs	
Slave access time			tSA	—	—	4	tRcyc	Figures 25.25 and 25.26
Slave out release time			tREL	—	—	3	tRcyc	

Notes: 1. I/O port driving ability: High.
2. I/O port driving ability: Low.

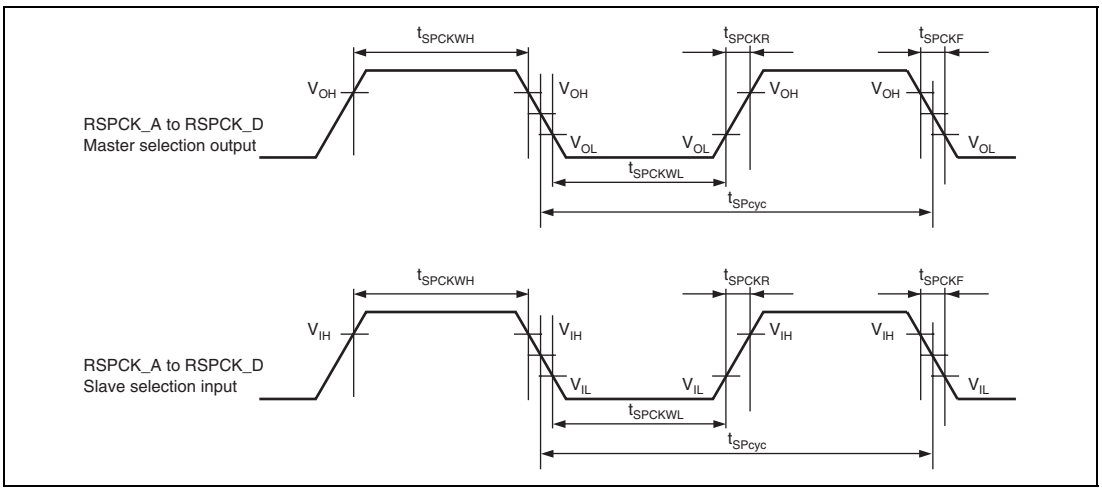


Figure 25.22 RSPI Clock Timing

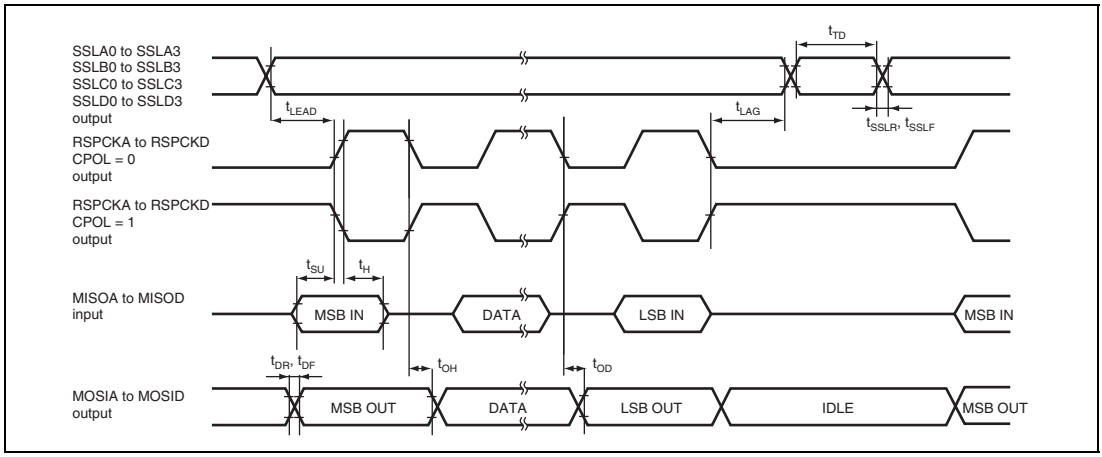


Figure 25.23 RSPI Timing (Master, CPHA = 0)

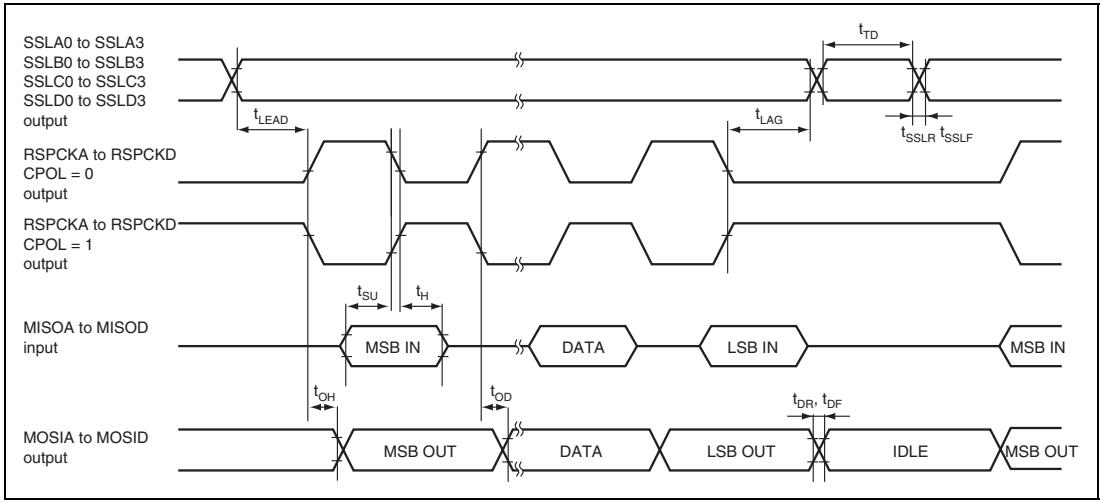


Figure 25.24 RSPI Timing (Master, CPHA = 1)

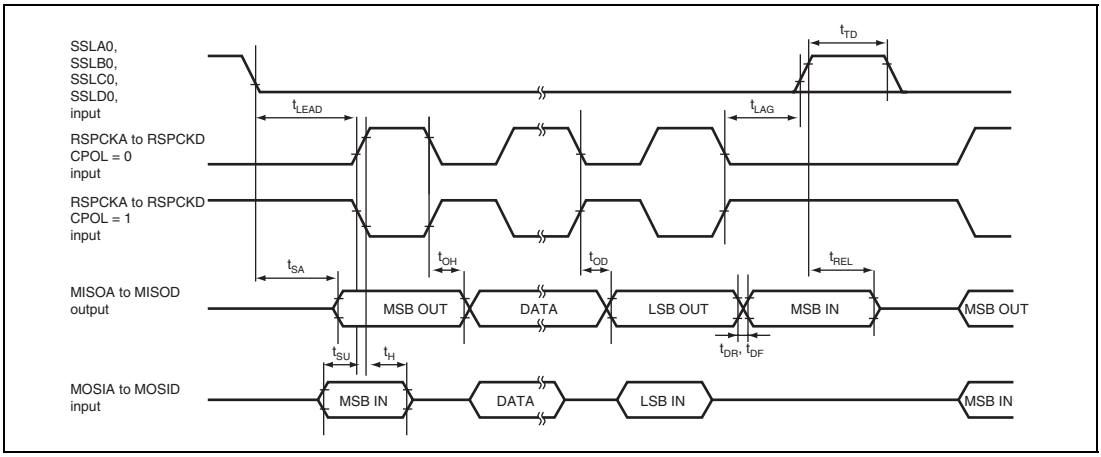


Figure 25.25 RSPI Timing (Slave, CPHA = 0)

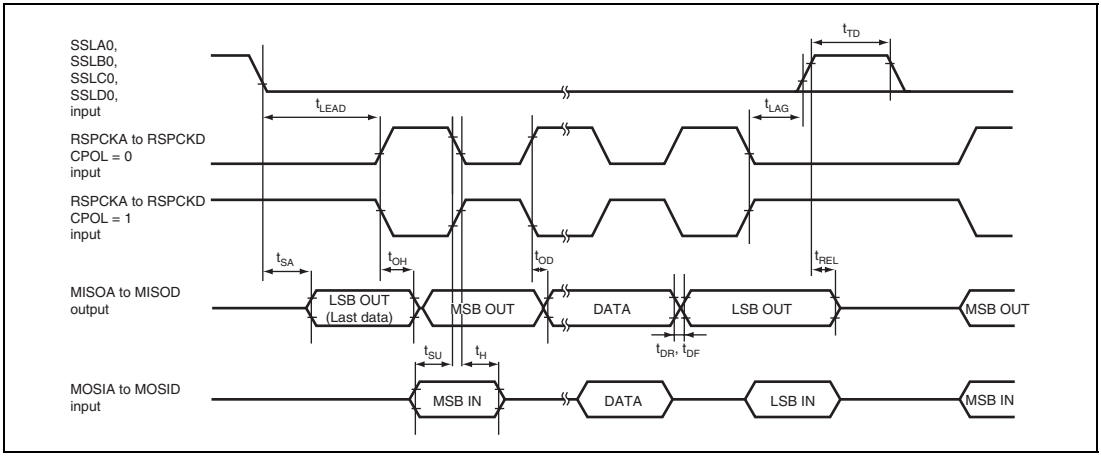


Figure 25.26 RSPI Timing (Slave, CPHA = 1)

25.3.4 A/D Conversion Characteristics

Table 25.8 A/D Conversion Characteristics

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V,
Vss = AVss = 0 V, Aϕ = 8 MHz to 40 MHz, Ta = –40 °C to +85 °C

Item	Min.	Typ.	Max.	Unit
Resolution	10	10	10	bit
Conversion time (40-Aϕ cycle)	1	—	5	µs
Analog input capacitance	—	—	20	pF
Permissible signal source impedance	—	—	5	KΩ
Nonlinearity error	—	—	±3.5	LSB
Offset error	—	—	±3.5	LSB
Full-scale error	—	—	±3.5	LSB
Quantization error	—	±0.5	—	LSB
Absolute accuracy	—	—	±4.0	LSB
Absolute accuracy of self diagnosis	—	—	±40	LSB

25.4 Flash Memory Characteristics

Table 25.9 Flash Memory Characteristics

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V,
Vss = AVss = 0 V, Iϕ = 8 MHz to 80 MHz, Fϕ = 8 MHz to 40 MHz,
Ta = –40 to +85 °C

Item		Symbol	Min.	Typ.	Max.	Unit	Test Conditions
Programming time *1, *2, *4	128 bytes	t_{P128}	—	1	10	ms	
	256 Kbytes	t_{P256K}	—	1.5	3.2	s	
Erasing time *1, *2, *4	4 Kbytes	t_{E4K}	—	25	60	ms	
	32 Kbytes	t_{E32K}	—	200	480	ms	
	64 Kbytes	t_{E64K}	—	400	875	ms	
	256 Kbytes	t_{E256K}	—	1.6	3.8	s	
Reprogramming/erasing cycle		N_{PEC}	100*3	—	—	Times	
Suspend delay during programming		t_{SPD}	—	—	120	µs	Figure 25.27
1st suspend delay during erasing (in suspension-priority mode)		t_{SESD1}	—	—	120	µs	
2nd suspend delay during erasing (in suspension-priority mode)		t_{SESD2}	—	—	1.7	ms	
Suspend delay during erasing (in erasure-priority mode)		t_{SEED}	—	—	1.7	ms	
Data retention time*4		t_{DRP}	20	—	—	Year	

Notes: 1. Programming time and erase time depend on data in the flash memory.
2. Programming time and erase time do not include time for data transfer.
3. All the characteristics after programming are guaranteed within this value (guaranteed value is from 1 to Min. value).
4. Characteristics when programming is performed within the Min. value.

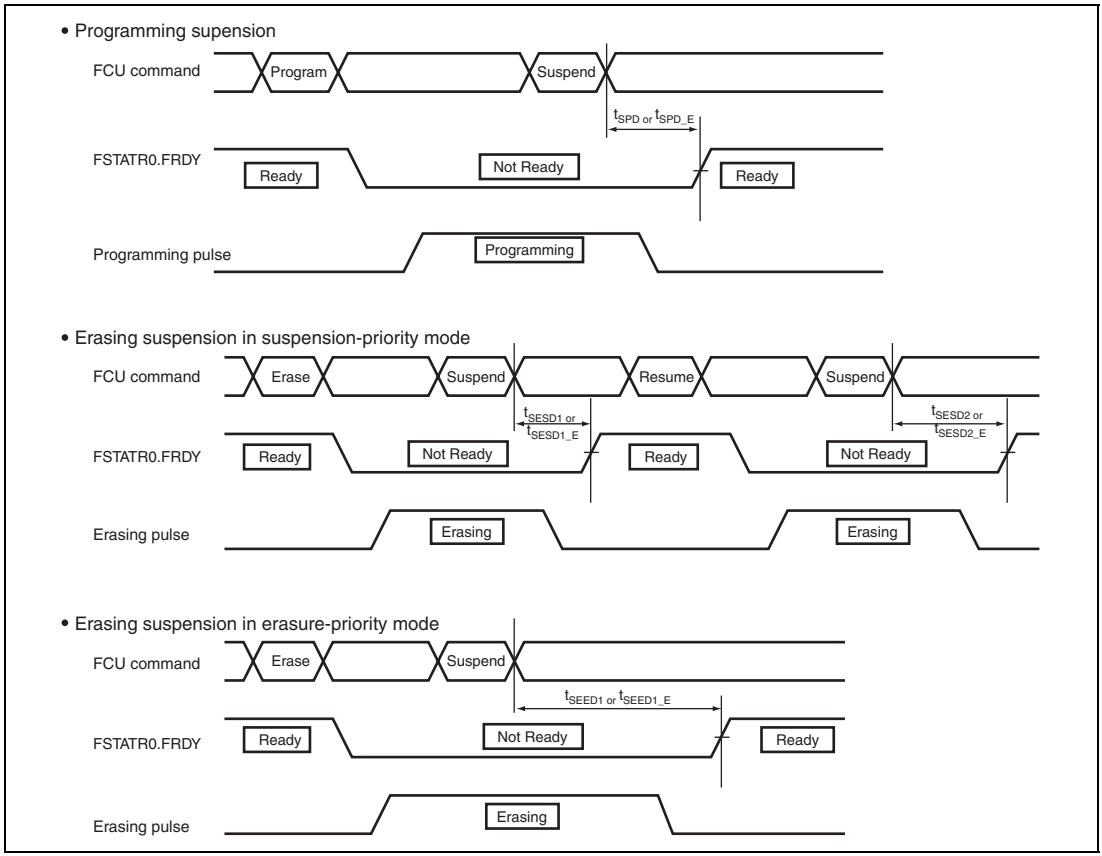


Figure 25.27 Flash Programming/Erasing Suspend Timing

25.5 EEPROM Characteristics

Table 25.10 shows the EEPROM characteristics.

Table 25.10 EEPROM Characteristics

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V,
Vss = AVss = 0 V, Iϕ = 8 MHz to 80 MHz, Fϕ = 8 MHz to 40 MHz,
Ta = –40 to +85 °C

Item		Symbol	Min.	Typ.	Max.	Unit	Reference
Programming time *1, *2, *4	8 bytes	t_{P8_E}	—	0.5	2.6	ms	
	128 bytes	t_{P128_E}	—	1.3	13	ms	
Erasing time *1, *2, *4	2 Kbytes	t_{E2K_E}	—	70	280	ms	
Blank check time *1, *4	8 bytes	t_{BC8_E}	—	—	30	µs	
	2 Kbytes	t_{BC2K_E}	—	—	0.7	ms	
Reprogramming/erasing cycle		N_{PEC_E}	30000 *3	—	—	Times	
Suspend delay during programming		t_{SPD_E}	—	—	120	µs	Figure 25.27
1st suspend delay during erasing (in suspension-priority mode)		t_{SESD1_E}	—	—	120	µs	
2nd suspend delay during erasing (in suspension-priority mode)		t_{SESD2_E}	—	—	1.7	ms	
Suspend delay during erasing (in erasure-priority mode)		t_{SEED_E}	—	—	1.7	ms	
Data retention time *4		T_{DRP_E}	15	—	—	Year	

Notes: 1. Programming time and erase time depend on data in the flash memory.
2. Programming time and erase time do not include time for data transfer.
3. All the characteristics after programming are guaranteed within this value (guaranteed value is from 1 to Min. value).
4. Characteristics when programming is performed within the Min. value.

25.6 Other Characteristics

25.6.1 Condition for External Oscillation Stoppage Detection

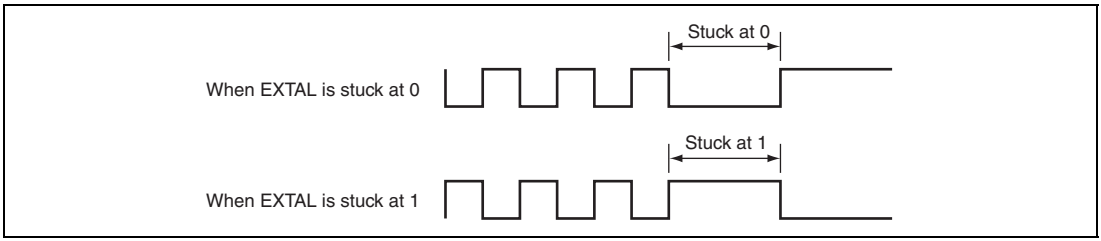


Figure 25.28 Condition for External Oscillation Stoppage Detection

Table 25.11 Condition for External Oscillation Stoppage Detection

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V, Vss = AVss = 0 V, Ta = –40 to +85 °C

Item	Test Condition	Reference
Condition for external oscillation stoppage detection	Stuck at 0 or 1	Figure 25.28

25.6.2 Timing of External Oscillation Stoppage Detection

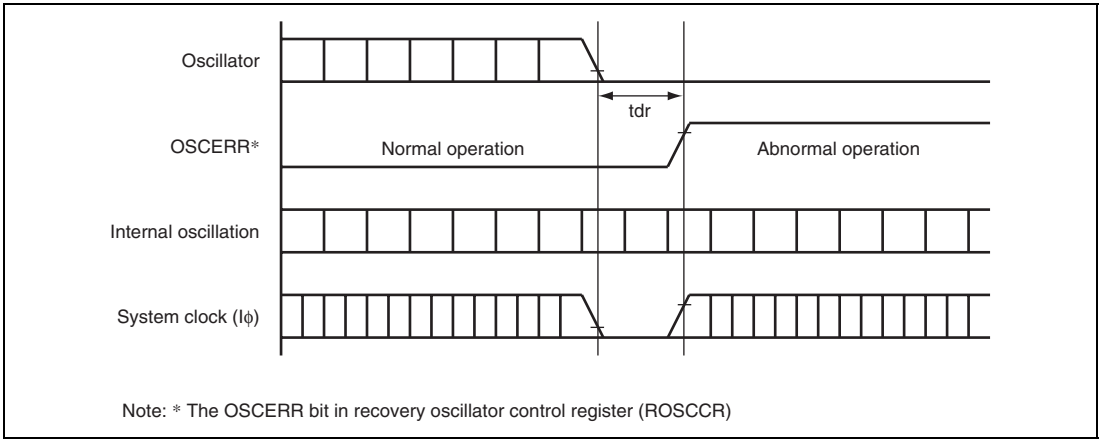


Figure 25.29 Timing of External Oscillation Stoppage Detection

Table 25.12 Detection of Abnormal Operation by the External Oscillation Stoppage Detection Circuit

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V,
Vss = AVss = 0 V, Ta = –40 to +85 °C

Item	Symbol	Min.	Typ.	Max.	Unit	Reference
Detection time	tdr	—	—	1.0	ms	Figure 25.29

25.6.3 Internal Oscillation Frequency

Table 25.13 Internal Oscillation Frequency

Conditions: Vcc = 4.5 V to 5.5 V, AVcc0 = 4.5 V to 5.5 V, AVcc1 = 4.5 V to 5.5 V,
Vss = AVss = 0 V, Ta = –40 to +85 °C

Item	Symbol	Min.	Typ.	Max.	Unit	Reference
Internal oscillation frequency	$f_{MAIN}\phi$	1.00	2.50	4.00	MHz	Figure 25.29

Appendix

A. I/O Port States in Each Processing State

Table A.1 I/O Port States in Each Processing State

Port Name	MCU Operating Mode	Reset	Software Standby Mode
Port 1	All modes	Hi-Z	Kept
Port 3	All modes	Hi-Z	Kept
Port 4	All modes	Hi-Z	Hi-Z
Port 5	All modes	Hi-Z	Hi-Z
Port 6	All modes	Hi-Z	Kept
Port A	All modes	Hi-Z	Kept
Port D	All modes	Hi-Z	Kept
Port H	All modes	Hi-Z	Kept
Port J	All modes	Hi-Z	Kept
Port K	All modes	Hi-Z	Kept

B. Product Lineup

Product Type	Type Code	Mark Code	Package	Code
H8SX/1727S	R5F61727S	R5F61727S	LQFP-100	PLQP0100KB-A (FP-100U)
H8SX/1725S	R5F61725S	R5F61725S		

C. Package Dimensions

For the package dimensions, data in the Renesas IC Package General Catalog has priority.

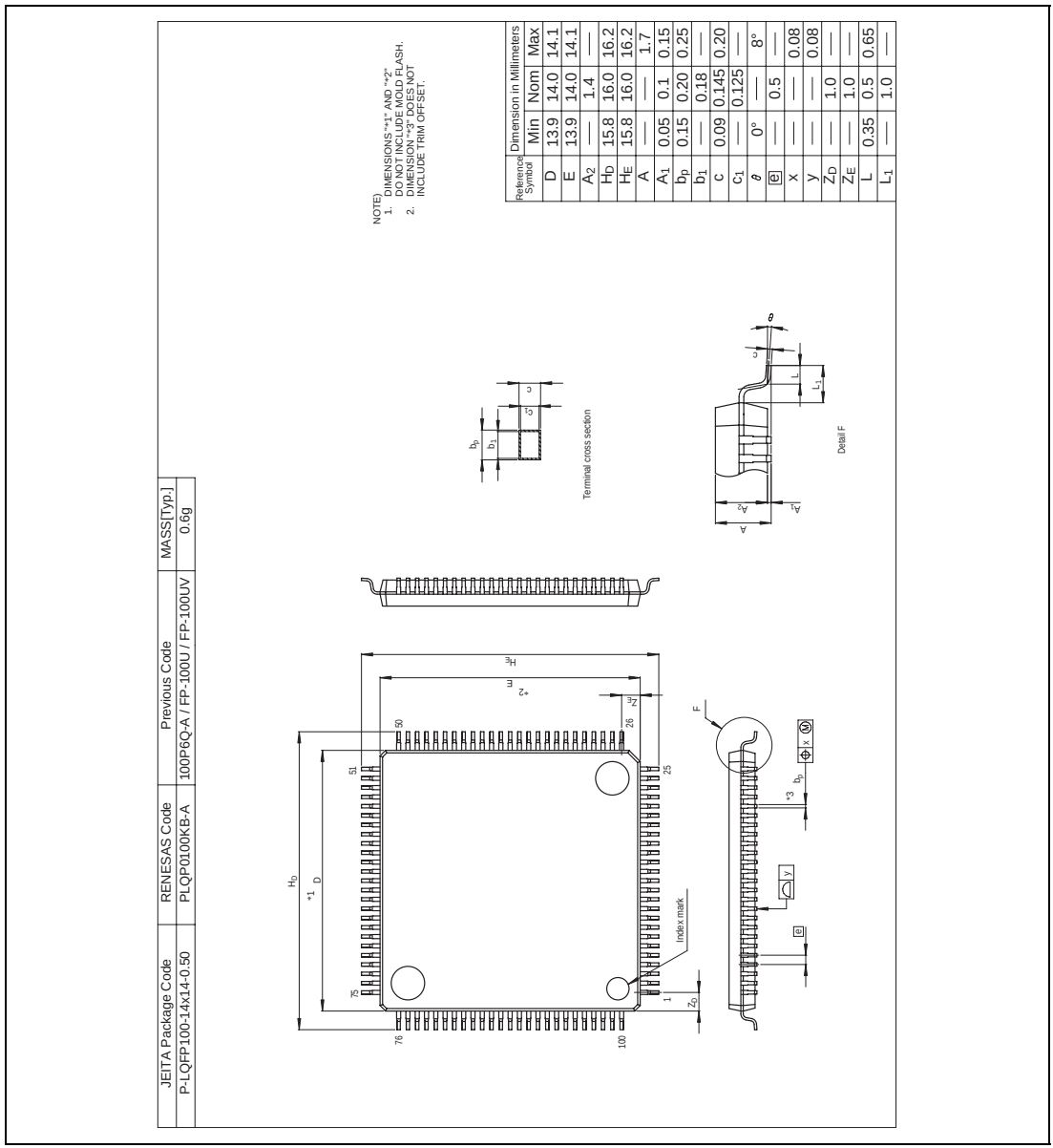


Figure C.1 Package Dimensions (PLQP0100KB-A)

Main Revisions and Additions in this Edition

Item	Page	Revision (See Manual for Details)
5.8.1 Conflict between Interrupt Generation and Disabling	127	Added When an interrupt enable bit is cleared to 0 to mask the interrupt, the masking becomes effective after execution of the instruction. When an interrupt enable bit is cleared to 0 by an instruction such as BCLR or MOV, if an interrupt is generated during execution of the instruction, the interrupt concerned will still be enabled on completion of the instruction, and so interrupt exception handling for that interrupt will be executed on completion of the instruction. If this conflict occurs in interrupt control mode 2, the interrupt exception handling with priority equal to or lower than the CPU interrupt mask level specified in the I2 to I0 bits in EXR may be executed on completion of the interrupt disabling instruction. However, if there is an interrupt request with priority over that interrupt, interrupt exception handling will be executed for the interrupt with priority, and another interrupt will be ignored. The same also applies when an interrupt source flag is cleared to 0. In interrupt control mode 2, the same also applies when the interrupt is disabled by modifying the IPR setting. Figure 5.7 shows an example in which the TCIEV bit in TIER of the TPU is cleared to 0. The above conflict will not occur if an enable bit or interrupt source flag is cleared to 0 or IPR is modified while the interrupt is masked. Figure 5.8 shows the interrupt masking procedure to avoid this conflict.
Figure 5.8 Procedure for Avoiding Interrupt Conflict	128	Added
5.8.6 Interrupt Flags of Peripheral Modules	129	Added and amended When using the CPU to clear an interrupt source flag and an interrupt enable bit of a peripheral module, be sure to obtain synchronization with the peripheral module by reading the register after clearing them from within the interrupt processing routine. This should be done regardless of the division ratio used to obtain the peripheral clock from the system clock.
20.6.2 Conditions for FCU Command Acceptance (2) Flash Memory P/E Mode • Flash memory lock bit read mode	866	Added In table 20.12, whether an error has occurred or not is determined from the CMDLK bit. (Note that when the FRDCLE and FRCCLE bits in the FRAMECCR register are set to 0, an FIFE interrupt by the CMDLK bit does not occur.)

Item	Page	Revision (See Manual for Details)
Table 20.12 FCU Modes/States and Acceptable Commands	867	Amended

	Status Read Mode								
Item	Programming/Erasure Processing	Programming Processing while Erasure-Suspended	Programming/Erasure Suspension Processing	Lock Bit Read 2 Processing	Programming-Suspended	Erasure-Suspended	Command Locked (FRDY = 0)	Command Locked (FRDY = 1)	Other State
FRDY bit in FSTATR0	0	0	0	0	1	1	0	1	1
SUSRDY bit in FSTATR0	1	0	0	0	0	0	0	0	0
ERSSPD bit in FSTATR0	0	1	0/1	0/1	0	1	0/1	0/1	0
PRGSPD bit in FSTATR0	0	0	0/1	0/1	1	0	0/1	0/1	0
CMDLK bit in FASTAT	0	0	0	0	0	0	1	1	0
Normal mode transition	×	×	×	×	A	A	×	×	A
Status read mode transition	×	×	×	×	A	A	×	×	A
Lock bit read mode transition (lock bit read 1)	×	×	×	×	A	A	×	×	A
Flash clock notification	×	×	×	×	×	×	×	×	A
Program	×	×	×	×	×	*	×	×	A
Block erase	×	×	×	×	×	×	×	×	A
P/E suspend	A	×	×	×	×	×	×	×	×
P/E resume	×	×	×	×	A	A	×	×	×
Status register clear	×	×	×	×	A	A	×	A	A
Lock bit read 2	×	×	×	×	A	A	×	×	A
Lock bit program	×	×	×	×	×	*	×	×	A

Item	Page	Revision (See Manual for Details)
20.6.3 FCU Command Usage (5) Using Flash Clock Notification Command	873	Amended First, set FCKAR to the frequency of the flash clock used. The range of settable frequencies is from 8 MHz to 40 MHz. The frequency should not be set outside this range.
Figure 20.19 Procedure for Issuing Flash Clock Notification Command	875	Amended [Before amendment] Timeout 10 µs [After amendment] Timeout 187 µs at 16 MHz 150 µs at 20 MHz 75 µs at 40 MHz
Figure 20.22 Procedure for Programming/Erasure Suspension	880	Amended [Before amendment] ECCERR → [After amendment] FRDTCT, FRCRCT
Table 20.14 Error Protection Types	893	Amended [Before amendment] ECCERR → [After amendment] FRCRCT

Item	Page	Revision (See Manual for Details)
Table 21.9 FCU Modes/States and Acceptable Commands	930	Amended

	Status Read Mode								
Item	Programming/Erasure Processing	Programming Processing while Erasure-Suspended	Programming/Erasure Suspension Processing	Blank Check Processing	Programming-Suspended	Erasure-Suspended	Command Locked (FRDY = 0)	Command Locked (FRDY = 1)	Other State
FRDY bit in FSTATR0	0	0	0	0	1	1	0	1	1
SUSRDY bit in FSTATR0	1	0	0	0	0	0	0	0	0
ERSSPD bit in FSTATR0	0	1	0/1	0/1	0	1	0/1	0/1	0
PRGSPD bit in FSTATR0	0	0	0/1	0/1	1	0	0/1	0/1	0
CMDLK bit in FASTAT	0	0	0	0	0	0	1	1	0
Normal mode transition	×	×	×	×	A	A	×	×	A
Status read mode transition	×	×	×	×	A	A	×	×	A
Lock bit read mode transition (lock bit read 1)	×	×	×	×	A	A	×	×	A
Flash clock notification	×	×	×	×	×	×	×	×	A
Program	×	×	×	×	×	*	×	×	A
Block erase	×	×	×	×	×	×	×	×	A
P/E suspend	A	×	×	×	×	×	×	×	×
P/E resume	×	×	×	×	A	A	×	×	×
Status register clear	×	×	×	×	A	A	×	A	A
Blank check	×	×	×	×	A	A	×	×	A

Item	Page	Revision (See Manual for Details)
Table 21.11 Data Stored in Product Information MAT (H8SX/1727S)	938	Amended **Information** / **Address** / **Example of Data** Device name / H'E0,0000 to H'E0,0008 / H'523546363137323753 = R5F61727S
Table 21.12 Data Stored in Product Information MAT (H8SX/1725S)	938	Amended **Information** / **Address** / **Example of Data** Device name / H'E0,0000 to H'E0,0008 / H'523546363137323553 = R5F61725S
22.2.1 Connecting Crystal Resonator	950	Deleted A crystal resonator can be connected as shown in the example in figure 22.2. Select the damping resistance Rd according to table 22.2. ~~An AT-cut parallel-resonance type should be used.~~
Figure 22.3 Crystal Resonator Equivalent Circuit	950	Deleted ~~AT-cut parallel-resonance type~~

Item	Page	Revision (See Manual for Details)
24.2 Register Bits	1029, 1030	Amended

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
TXPR1_0	TXPR1_15	TXPR1_14	TXPR1_13	TXPR1_12	TXPR1_11	TXPR1_10	TXPR1_9	TXPR1_8
	TXPR1_7	TXPR1_6	TXPR1_5	TXPR1_4	TXPR1_3	TXPR1_2	TXPR1_1	TXPR1_0
TXPR0_0	TXPR0_15	TXPR0_14	TXPR0_13	TXPR0_12	TXPR0_11	TXPR0_10	TXPR0_9	TXPR0_8
	TXPR0_7	TXPR0_6	TXPR0_5	TXPR0_4	TXPR0_3	TXPR0_2	TXPR0_1	—
TXCR1_0	TXCR1_15	TXCR1_14	TXCR1_13	TXCR1_12	TXCR1_11	TXCR1_10	TXCR1_9	TXCR1_8
	TXCR1_7	TXCR1_6	TXCR1_5	TXCR1_4	TXCR1_3	TXCR1_2	TXCR1_1	TXCR1_0
TXCR0_0	TXCR0_15	TXCR0_14	TXCR0_13	TXCR0_12	TXCR0_11	TXCR0_10	TXCR0_9	TXCR0_8
	TXCR0_7	TXCR0_6	TXCR0_5	TXCR0_4	TXCR0_3	TXCR0_2	TXCR0_1	—
TXACK1_0	TXACK1_15	TXACK1_14	TXACK1_13	TXACK1_12	TXACK1_11	TXACK1_10	TXACK1_9	TXACK1_8
	TXACK1_7	TXACK1_6	TXACK1_5	TXACK1_4	TXACK1_3	TXACK1_2	TXACK1_1	TXACK1_0
TXACK0_0	TXACK0_15	TXACK0_14	TXACK0_13	TXACK0_12	TXACK0_11	TXACK0_10	TXACK0_9	TXACK0_8
	TXACK0_7	TXACK0_6	TXACK0_5	TXACK0_4	TXACK0_3	TXACK0_2	TXACK0_1	—
ABACK1_0	ABACK1_15	ABACK1_14	ABACK1_13	ABACK1_12	ABACK1_11	ABACK1_10	ABACK1_9	ABACK1_8
	ABACK1_7	ABACK1_6	ABACK1_5	ABACK1_4	ABACK1_3	ABACK1_2	ABACK1_1	ABACK1_0
ABACK0_0	ABACK0_15	ABACK0_14	ABACK0_13	ABACK0_12	ABACK0_11	ABACK0_10	ABACK0_9	ABACK0_8
	ABACK0_7	ABACK0_6	ABACK0_5	ABACK0_4	ABACK0_3	ABACK0_2	ABACK0_1	—
RXPR1_0	RXPR1_15	RXPR1_14	RXPR1_13	RXPR1_12	RXPR1_11	RXPR1_10	RXPR1_9	RXPR1_8
	RXPR1_7	RXPR1_6	RXPR_15	RXPR1_4	RXPR1_3	RXPR1_2	RXPR1_1	RXPR1_0
RXPR0_0	RXPR0_15	RXPR0_14	RXPR0_13	RXPR0_12	RXPR0_11	RXPR0_10	RXPR0_9	RXPR0_8
	RXPR0_7	RXPR0_6	RXPR0_5	RXPR0_4	RXPR0_3	RXPR0_2	RXPR0_1	RXPR0_0
RFPR1_0	RFPR1_15	RFPR1_14	RFPR1_13	RFPR1_12	RFPR1_11	RFPR1_10	RFPR1_9	RFPR1_8
	RFPR1_7	RFPR1_6	RFPR1_5	RFPR1_4	RFPR1_3	RFPR1_2	RFPR1_1	RFPR1_0
RFPR0_0	RFPR0_15	RFPR0_14	RFPR0_13	RFPR0_12	RFPR0_11	RFPR0_10	RFPR0_9	RFPR0_8
	RFPR0_7	RFPR0_6	RFPR0_5	RFPR0_4	RFPR0_3	RFPR0_2	RFPR0_1	RFPR0_0
MBIMR1_0	MBIMR1_15	MBIMR1_14	MBIMR1_13	MBIMR1_12	MBIMR1_11	MBIMR1_10	MBIMR1_9	MBIMR1_8
	MBIMR1_7	MBIMR1_6	MBIMR1_5	MBIMR1_4	MBIMR1_3	MBIMR1_2	MBIMR1_1	MBIMR1_0
MBIMR0_0	MBIMR0_15	MBIMR0_14	MBIMR0_13	MBIMR0_12	MBIMR0_11	MBIMR0_10	MBIMR0_9	MBIMR0_8
	MBIMR0_7	MBIMR0_6	MBIMR0_5	MBIMR0_4	MBIMR0_3	MBIMR0_2	MBIMR0_1	MBIMR0_0
UMSR1_0	UMSR1_15	UMSR1_14	UMSR1_13	UMSR1_12	UMSR1_11	UMSR1_10	UMSR1_9	UMSR1_8
	UMSR1_7	UMSR1_6	UMSR1_5	UMSR1_4	UMSR1_3	UMSR1_2	UMSR1_1	UMSR1_0
UMSR0_0	UMSR0_15	UMSR0_14	UMSR0_13	UMSR0_12	UMSR0_11	UMSR0_10	UMSR0_9	UMSR0_8
	UMSR0_7	UMSR0_6	UMSR0_5	UMSR0_4	UMSR0_3	UMSR0_2	UMSR0_1	UMSR0_0

Item	Page	Revision (See Manual for Details)
24.2 Register Bits	1031	Amended

Register Abbreviation	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
TCMR0_0	TCMR0_15	TCMR0_14	TCMR0_13	TCMR0_12	TCMR0_11	TCMR0_10	TCMR0_9	TCMR0_8
	TCMR0_7	TCMR0_6	TCMR0_5	TCMR0_4	TCMR0_3	TCMR0_2	TCMR0_1	TCMR0_0
TCMR1_0	TCMR1_15	TCMR1_14	TCMR1_13	TCMR1_12	TCMR1_11	TCMR1_10	TCMR1_9	TCMR1_8
	TCMR1_7	TCMR1_6	TCMR1_5	TCMR1_4	TCMR1_3	TCMR1_2	TCMR1_1	TCMR1_0
TCMR2_0	TCMR2_15	TCMR2_14	TCMR2_13	TCMR2_12	TCMR2_11	TCMR2_10	TCMR2_9	TCMR2_8
	TCMR2_7	TCMR2_6	TCMR2_5	TCMR2_4	TCMR2_3	TCMR2_2	TCMR2_1	TCMR2_0

Index

F

H

I

L

M

N

O

P

Q

R

S

T

U

W

H8SX/1720S Group User's Manual: Hardware

Publication Date: Rev.1.00 Feb 04, 2010
Rev.2.00 Jun 25, 2012

Published by: Renesas Electronics Corporation

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.1

H8SX/1720S Group

Renesas Electronics Corporation

R01UH0370EJ0200
(Previous Number:REJ09B0563-0100)