

NEC

User's Manual

V25™, V35™

16/8-, 16-BIT SINGLE-CHIP MICROCONTROLLERS

HARDWARE

μPD70320

μPD70330

Document No. U13030EJDV0UM00 (13th edition)
(O.D.No. IEM-1220)
Date Published January 1998 N CP(K)

© NEC Corporation 1995
Printed in Japan

[MEMO]

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

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

② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

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

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

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

V20, V25, V25+, V30, V35, and V35+ are trademarks of NEC Corporation.
Intertool is a trademark of Intermetrics Microsystems Software, Inc.

The application circuits and their parameters are for reference only and are not intended for use in actual design-ins.

The information in this document is subject to change without notice.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document.

NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or others.

While NEC Corporation has been making continuous effort to enhance the reliability of its semiconductor devices, the possibility of defects cannot be eliminated entirely. To minimize risks of damage or injury to persons or property arising from a defect in an NEC semiconductor device, customers must incorporate sufficient safety measures in its design, such as redundancy, fire-containment, and anti-failure features.

NEC devices are classified into the following three quality grades:

“Standard”, “Special”, and “Specific”. The Specific quality grade applies only to devices developed based on a customer designated “quality assurance program” for a specific application. The recommended applications of a device depend on its quality grade, as indicated below. Customers must check the quality grade of each device before using it in a particular application.

Standard: Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots

Special: Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support)

Specific: Aircrafts, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems or medical equipment for life support, etc.

The quality grade of NEC devices is “Standard” unless otherwise specified in NEC’s Data Sheets or Data Books. If customers intend to use NEC devices for applications other than those specified for Standard quality grade, they should contact an NEC sales representative in advance.

Anti-radioactive design is not implemented in this product.

Regional Information

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

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

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

NEC Electronics Inc. (U.S.)

Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Germany) GmbH

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

NEC Electronics (UK) Ltd.

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

NEC Electronics Italiana s.r.l.

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

NEC Electronics (Germany) GmbH

Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.

Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.

Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics (Germany) GmbH

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

NEC Electronics Hong Kong Ltd.

Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.

Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.

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

NEC Electronics Taiwan Ltd.

Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC do Brasil S.A.

Cumbica-Guarulhos-SP, Brasil
Tel: 011-6465-6810
Fax: 011-6465-6829

Major Revisions in This Edition

Page	Description
Throughout	Deleted the μ PD70320(A), 70330(A), and 70P322.
	Deleted the description of the μ PD70P322 pins (V_{25}/V_{35} , \overline{PROG} , V_{PP} , \overline{CE} , and \overline{OE} pins).
p. 27, 28	Changed the I/O direction of pin No. 78 in Section 1.4 (1) 84-pin plastic QFJ (1150 x 1150 mil) .
p. 29	Changed the I/O direction of pin No. 53 in Section 1.4 (2) (a) V25 .
p. 30	Changed the I/O direction of pin No. 53 and No. 82 to 89 in Section 1.4 (2) (b) V35 .
p. 32, 33	Changed the description related to the deletion of the μ PD70P322 in Section 1.5 Internal Block Diagram .
p. 57	Deleted the description of Internal ROM in Section 3.1.3 BCU (Bus control unit) .
p. 73	Deleted the internal ROM area in Figure 3-4. Memory Map .
p. 74	Deleted the description of internal ROM in Section 3.5.1 (1) Access to internal data area .
p. 75	Deleted the internal ROM area in Figure 3-5. Memory Space Access Conditions .
p. 83	Changed the description related to the deletion of the μ PD70P322 in Section 3.5.6 External memory area .
p. 138	Added the partial description to Table 5-3. Address Time-division Multiplexing Output (V35) .
p. 163	Added the partial description to Figure 5-8. μPD41256 Connection Circuit Example (When 5MHz) .
p. 173	Deleted the description of the internal ROM in Section 6.2 DMA Operation .
p. 224	Changed the description partially in Section 11.2.3 Reception .

The mark ★ shows major revised points.

PREFACE

Target Readers This manual is intended for user engineers who understand the functions of the μ PD70320 (also called V25) and μ PD70330 (also called V35) and design application systems using these devices.

Purpose This manual explains how you understand the hardware functions of the μ PD70320 and 70330, as listed below.

Organization The μ PD70320 and 70330 User's Manuals are divided into two editions: a hardware edition (this manual) and an instruction edition.

Hardware edition

- General description
- Pin functions
- Internal CPU functions
- Internal peripheral devices
- Standby function
- Reset function
- Appendix (Restrictions, Q&A)

Instruction edition

- General description
- Explanation of instructions
- Additional instructions to V20, V30
- Appendix

How to read this manual This manual assumes that the reader has general knowledge of electricity, logic circuits, and microcontrollers. Unless otherwise stated, the descriptions apply to the μ PD70320 and 70330.

For readers who have experience using the μ PD70108 or 70116 (also called V20™ or V30™)

→ Instructions of the V25 and V35 are compatible with those of the V20 and V30 in the native mode.

See **CHAPTER 3 ADDITIONAL INSTRUCTIONS FOR THE V20 AND V30** in the **instruction edition** of the **V25, V35 FAMILY USER'S MANUAL**.

For details of instruction functions

→ See the **instruction edition** of the **V25, V35 FAMILY USER'S MANUAL**.

For a general understanding of V25 and V35 functions

→ Read this manual from chapter 1.

If you have questions about operation of the V25 or V35
 → See **APPENDIX C Q & A**.

For information on the V25 and V35's electrical characteristics
 → See the relevant data sheet (separately available).

For information on application examples of the V25 and V35's functions
 → See the relevant application note (separately available).

Terms and conventions used in this manual	Data representation weight	:	High-order digits at left and low-order digits at right
	Active low representation	:	$\overline{\text{xxx}}$ (pin or signal name is overlined)
	Memory map address	:	High order at low stage and low order at high stage
	Note	:	Explanation of (Note) in the text
	Caution	:	Item deserving extra attention
	Remark	:	Supplementary explanation to the text
Number representation	:	Binary number is xxxxB Decimal number is xxxxD Hexadecimal number is xxxxH	

Suffix representing an exponent of 2 (in address space or memory capacity)

K (Kilo): $2^{10} = 1024$

M (Mega): $2^{20} = 1024^2$

Related Documents

Document name Product name	Data sheet	User's manual		Application note
		Hardware	Instruction	
μ PD70320	U10090E	This manual	U12120E	IEA-1256
μ PD70330	—			—

Related Documents

Document name	Document No.
IE-70320-BX and IE-70330-BX (In-circuit emulator)	EEU-842
RA70116-I (Intertool™ assembler) and SP70116-I (Intertool software package) Assembly language	EEU-861
RA70116-I (Intertool assembler) and SP70116-I (Intertool software package) Operation	EEU-869
LC70116 (Locator)	EEU-834

CONTENTS

CHAPTER 1 GENERAL DESCRIPTION	23
1.1 Features	24
1.2 Differences with V25+™ and V35+™	25
1.3 Ordering Information	26
1.4 Pin Configuration (Top View)	27
1.5 Internal Block Diagram	32
CHAPTER 2 PIN FUNCTIONS	35
2.1 Pin Function Lists	35
2.1.1 μ PD70320	35
2.1.2 μ PD70330	38
2.1.3 Pin state in each mode	40
2.2 Description of Pin Functions	41
2.2.1 P00 to P07 (Port 0)	41
2.2.2 P10 to P17 (Port 1)	42
2.2.3 P20 to P27 (Port 2)	44
2.2.4 PT0 to PT7 (Port with Comparator 0 to 7)	46
2.2.5 V_{TH} (Threshold Voltage)	46
2.2.6 TxD0 and TxD1 (Transmit Data 0 and 1)	46
2.2.7 RxD0 and RxD1 (Receive Data 0 and 1)	46
2.2.8 $\overline{CTS0}$ (Clear to Send 0)	47
2.2.9 $\overline{CTS1}$ (Clear to Send 1)	47
2.2.10 \overline{RESET} (Reset)	47
2.2.11 \overline{EA} (External Access)	47
2.2.12 X1 and X2 (Crystal)	48
2.2.13 D0 to D15 (Data Bus 0 to 15)	48
2.2.14 A0 to A19 (Address Bus 0 to 19)	48
2.2.15 \overline{MREQ} (Memory Request)	49
2.2.16 \overline{MSTB} (Memory Strobe)	49
2.2.17 $\overline{R/\overline{W}}$ (Read/Write Strobe)	49
2.2.18 \overline{REFRQ} (Refresh Request)	49
2.2.19 \overline{IOSTB} (I/O Strobe)	49
2.2.20 V_{DD} (Power Supply)	49
2.2.21 GND (Ground)	50
2.2.22 \overline{UBE} (Upper Byte Enable)	50
2.3 Pin I/O Circuits	51
2.3.1 I/O circuit types	51
2.3.2 I/O circuits	53
2.4 Unused Pin Connections	54

CHAPTER 3 INTERNAL CPU FUNCTIONS	57
3.1 Hardware Configuration	57
3.1.1 PAU (Address calculation unit)	57
3.1.2 EXU (Execution unit)	57
3.1.3 BCU (Bus control unit)	57
3.1.4 CPU pipelines	58
3.2 Registers	61
3.2.1 Register banks	61
3.2.2 General purpose registers (AW, BW, CW, and DW)	63
3.2.3 Pointers (SP and BP) and index registers (IX and IY)	64
3.2.4 Segment registers (PS, SS, DS0, and DS1)	65
3.2.5 Internal data area base register (IDB)	66
3.2.6 Special function registers	66
3.3 Program Counter (PC)	67
3.4 Program Status Word (PSW)	67
3.4.1 CY (Carry Flag)	68
3.4.2 P (Parity Flag)	69
3.4.3 AC (Auxiliary Flag)	69
3.4.4 Z (Zero Flag)	69
3.4.5 S (Sign Flag)	70
3.4.6 V (Overflow Flag)	70
3.4.7 $\overline{\text{IBRK}}$ (I/O Break Flag)	71
3.4.8 BRK (Break Flag)	71
3.4.9 IE (Interrupt Enable Flag)	71
3.4.10 DIR (Direction Flag)	71
3.4.11 RB0 to RB2 (Register Banks 0 to 2 Flag)	71
3.4.12 F0 and F1 (User Flags 0 and 1)	71
3.5 Memory Space	72
3.5.1 Internal data area	74
3.5.2 Internal data area base register (IDB)	77
3.5.3 Special function register area	77
3.5.4 On-chip RAM area	81
3.5.5 Vector table area	82
3.5.6 External memory area	83
3.6 I/O Space	84
CHAPTER 4 INTERRUPT FUNCTIONS	85
4.1 Interrupt Controller	85
4.2 Interrupt Sources	86
4.3 Priority Level Control	88
4.3.1 Multiple interrupt priority level control	88
4.3.2 Priority level control for simultaneously occurring interrupts	90
4.4 Interrupt Requests	91

4.5	Interrupt Response Modes	92
4.5.1	Vectored interrupts	92
4.5.2	Register bank switching function	93
4.5.3	Macro service function	98
4.5.4	Macro service control register	104
4.6	NMI (Non-Maskable Interrupt)	107
4.7	INT (Interrupt)	108
4.8	Interrupt Request Control Register	109
4.9	Interrupt Priority Register (ISPR)	111
4.10	External Interrupts	113
4.10.1	External interrupt mode register (INTM)	113
4.10.2	External interrupt request control registers (EXIC0 to EXIC2)	114
4.10.3	External interrupt macro service control registers (EMS0 to EMS2)	114
4.11	Software Interrupts	115
4.11.1	General software interrupts	115
4.11.2	I/O instruction interrupts	116
4.11.3	FPO instruction interrupt	117
4.12	If Interrupt Requests cannot be Acknowledged	118
4.13	Timing at which an Interrupt cannot be Acknowledged	118
4.14	Interrupt Servicing during Execution of Block Servicing Instruction	118
4.15	How Interrupts are Acknowledged	119
4.16	Hardware Interrupt Response Time	126
4.16.1	V25's interrupt response time (number of system clock cycles)	126
4.16.2	V25's NMI response time (number of system clock cycles)	128
4.16.3	V25's INT response time (number of system clock cycles)	129
4.16.4	V35's interrupt response time (number of system clock cycles)	131
4.16.5	V35's NMI response time (number of system clock cycles)	132
4.16.6	V35's INT response time (number of system clock cycles)	133
CHAPTER 5	BUS CONTROL	135
5.1	Programmable Wait Function	139
5.2	Bus Hold Function	157
5.2.1	Response time from HLDRQ to $\overline{\text{HLDAK}}$ (unit: clock cycles)	157
5.3	Refresh Functions	158
5.3.1	Refresh mode register (RFM)	158
5.3.2	Connection to pseudo SRAM	162
5.3.3	Connection to DRAM	163
5.4	Bus Mastership	164
5.5	Bus Timings	165
5.5.1	Bus timing of V25	165
5.5.2	Bus timing of V35	168

CHAPTER 6 DMA CONTROLLER	173
6.1 Pin Functions	173
6.2 DMA Operation	173
6.3 DMA Control Registers	177
6.3.1 DMA mode registers (DMAM0 and DMAM1)	177
6.3.2 DMA control registers (DMAC0 and DMAC1)	179
6.3.3 DMA interrupt request control registers (DIC0 and DIC1)	180
6.4 DMA Service Channels	181
6.5 DMA Transfer Timings	183
6.5.1 DMA transfer timing of V25	183
6.5.2 DMA transfer timing of V35	185
6.6 DMA Execution Time	189
6.6.1 V25's DMA execution time (number of system clock cycles)	189
6.6.2 V35's DMA execution time (number of system clock cycles)	190
 CHAPTER 7 PORT FUNCTIONS	 191
7.1 Ports 0 to 2	191
7.1.1 Hardware structure	191
7.1.2 Port functions	194
7.2 Port T (PT0 to PT7)	199
7.2.1 Hardware structure	199
7.2.2 Port T mode register (PMT)	200
7.2.3 Port T reception	201
 CHAPTER 8 CLOCK GENERATOR	 203
8.1 Clock Generator Structure	203
8.2 Processor Control Register (PRC)	206
 CHAPTER 9 TIMER UNIT	 207
9.1 Timer Unit Structure and Operation	207
9.2 Timer Control Registers (TMC0 and TMC1)	209
9.3 Timer Unit Interrupt Requests	215
9.3.1 Timer unit interrupt request control registers (TMIC0, TMIC1, and TMIC2)	216
9.3.2 Timer unit macro service control registers (TMMS0, TMMS1, and TMMS2)	216
 CHAPTER 10 TIME BASE COUNTER	 217
10.1 Time Base Counter Structure	217
10.2 Time Base Interval Specification	218
10.3 Time Base Interrupt Request Control Register (TBIC)	219

CHAPTER 11 SERIAL INTERFACE	221
11.1 Serial Interface Configuration	221
11.2 Asynchronous Mode	223
11.2.1 Transmission	223
11.2.2 Transmission completion interrupt	224
11.2.3 Reception	224
11.2.4 Reception completion interrupt	225
11.3 I/O Interface Mode	226
11.3.1 Transmission	226
11.3.2 Transmission completion interrupt	226
11.3.3 Reception	227
11.3.4 Reception error interrupt	227
11.3.5 Serial register clear	227
11.4 Starting Transmission without Using Interrupts	228
11.5 Serial Mode Registers (SCM0 and SCM1)	229
11.6 Baud Rate Generators	233
11.6.1 Serial control registers (SCC0 and SCC1)	235
11.7 Serial Error Handling	236
11.7.1 Serial error registers (SCE0 and SCE1)	236
11.8 Break State Detection Function	238
11.9 Serial Interface Interrupt Requests	239
11.9.1 Interrupt request control registers (SEICn, SRICn, and STICn: n = 0 or 1)	239
11.9.2 Macro service control registers (SRMSn and STMSn: n = 0 or 1)	240
CHAPTER 12 STANDBY FUNCTION	241
12.1 Standby Control Register (STBC)	241
12.2 HALT Mode	242
12.2.1 HALT mode release	242
12.3 STOP Mode	244
12.3.1 STOP mode release	244
CHAPTER 13 RESET FUNCTION	247
CHAPTER 14 ADDRESS GENERATION	249
14.1 Instruction Address	249
14.1.1 Direct addressing	249
14.1.2 Relative addressing	249
14.1.3 Register addressing	250
14.1.4 Register indirect addressing	250
14.1.5 Index addressing	250
14.1.6 Based addressing	251
14.1.7 Based index addressing	251

14.2	Memory Operand Address	252
14.2.1	Register addressing	252
14.2.2	Immediate addressing	252
14.2.3	Direct addressing	253
14.2.4	Register indirect addressing	253
14.2.5	Automatic increment/decrement addressing	253
14.2.6	Index addressing	254
14.2.7	Based addressing	254
14.2.8	Based index addressing	255
14.2.9	Bit addressing	256
14.2.10	Special function register addressing	256
CHAPTER 15	OPERATION STATE TRANSITION	257
APPENDIX A	LIMITATIONS	259
A.1	Limitation on Conflict between Macro Service and INT	259
A.1.1	Description of limitation	259
A.1.2	Avoidance methods	259
A.2	Cautions on Use of the MOVSPA Instruction	260
A.2.1	Device operation description (symptom)	260
A.2.2	Avoidance methods	262
A.3	Limitations on Send Data Missing by Transmission Disable Operation during Serial Transmission	263
A.3.1	Device operation description (symptom)	263
A.3.2	Avoidance methods	264
A.4	Cautions on Interrupt Priority Levels and Servicing Sequence	266
A.5	Limitations on Instruction Slips during HALT Mode	267
A.5.1	Description of limitation	267
A.5.2	Avoidance method	267
A.6	Limitation on CPU Deadlock Due to CLKOUT Output	267
A.6.1	Description of limitation	267
A.6.2	Avoidance method	267
A.7	Caution on Starting Timer Countdown	268
A.7.1	Device operation description (symptom)	268
A.7.2	Avoidance method	268
A.8	Limitations on Macro Service Masks	270
A.8.1	Device operation description (symptom)	270
A.8.2	Avoidance method	271
A.9	Limitations on CVTBD/CVTDB Instructions	272
A.9.1	Device operation description (symptom)	272
A.9.2	Avoidance method	272

APPENDIX B SETTING EXAMPLES	275
B.1 Ports	275
B.2 Programmable Wait, Processor Control, and Refresh Function	276
B.3 Register Bank Switching	279
B.3.1 Register bank switching by interrupt request	279
B.3.2 Register bank switching by instruction (BRKCS or MOVSPA instruction)	281
B.3.3 Register bank switching by instruction (MOVSPB or TSKSW instruction)	282
B.4 Access to Internal Data Area	283
B.5 Timer Unit	284
B.6 I/O Interface Mode	288
B.7 Macro Service	290
B.7.1 Normal mode (serial interface UART transmission)	290
B.7.2 Character search mode (serial interface UART reception)	293
B.8 DMA Controller	296
B.8.1 Demand release mode	296
B.8.2 Single step mode	297
B.8.3 Burst mode	298
APPENDIX C Q & A	299
C.1 Internal CPU Function	304
C.2 Interrupt Function	306
C.3 Bus Control	313
C.4 DMA Controller	318
C.5 Clock Generator	326
C.6 Timer Unit	327
C.7 Serial Interface	328
C.8 Standby Function	332
C.9 Reset Function	333
C.10 Other	335
APPENDIX D REGISTER INDEX (IN ALPHABETICAL ORDER)	337

[MEMO]

LIST OF FIGURES (1/3)

Figure No.	Title	Page
3-1	CPU Pipeline Operation Example	59
3-2	Register Bank Configuration	62
3-3	FLAG Register	71
3-4	Memory Map	73
3-5	Memory Space Access Conditions	75
3-6	On-chip RAM Area Map	81
3-7	I/O Map (64 Kbytes)	84
4-1	Servicing Mode of Interrupts Subject to Multiple Servicing Control	89
4-2	Multiple servicing of Same Interrupts	91
4-3	Interrupt Acknowledge Operation	92
4-4	Register Bank Switching Sequence	94
4-5	Register Bank Return Sequence	95
4-6	Register Bank Switching Sequence by Executing TSKSW Instruction	97
4-7	Interrupt Servicing Efficiency Using Macro Service	99
4-8	Normal Mode Operation Flow	100
4-9	Example of Serial Interface Transmission	101
4-10	Character Search Mode Operation Flow	102
4-11	Example of Serial Interface Reception	103
4-12	Macro Service Control Register	105
4-13	Macro Service Channel Configuration	105
4-14	INT Interrupt Acknowledge Timing	108
4-15	Interrupt Request Control Register	110
4-16	ISPR	111
4-17	ISPR States	112
4-18	INTM	113
4-19	EXIC0, EXIC1, and EXIC2	114
4-20	EMS0, EMS1, and EMS2	114
5-1	Memory Bank Structure	137
5-2	WTC	139
5-3	Wait via READY Pin (V25)	141
5-4	Wait via READY Pin (V35)	149
5-5	Control Circuit Depending on the RFLV Bit Contents	159
5-6	RFM	161
5-7	μ PD42832 Connection Circuit Example	162
5-8	μ PD41256 Connection Circuit Example (when 5 MHz)	163
5-9	Memory Read Cycle	165
5-10	Memory Write Cycle	165
5-11	I/O Read Cycle	165
5-12	I/O Write Cycle	165
5-13	Memory Read Cycle (when one wait state is inserted)	165
5-14	Memory Write Cycle (when two wait states are inserted)	165
5-15	Memory Write Cycle (during READY pin operation)	166
5-16	Refresh Cycle (when one wait state is inserted)	166
5-17	Bus Hold Acknowledgment Release Timing	166
5-18	Refresh Cycle in Hold Mode (0 wait state)	166
5-19	Bus Hold Acknowledgment Release Timing during HALT Mode	167
5-20	Memory Read Cycle	168
5-21	Memory Write Cycle	168
5-22	I/O Read Cycle	168
5-23	I/O Write Cycle	168
5-24	Memory Read Cycle (when one wait state is inserted)	169
5-25	Memory Write Cycle (when two wait states are inserted)	169
5-26	Memory Write Cycle (during READY pin operation)	170

LIST OF FIGURES (2/3)

Figure No.	Title	Page
5-27	Refresh Cycle (when one wait state is inserted)	170
5-28	Bus Hold Acknowledgment Release Timing	171
5-29	Refresh Cycle in Hold Mode	171
5-30	Bus Hold Acknowledgment Release Timing during HALT Mode	172
6-1	DMAM0 and DMAM1	178
6-2	DMAC0 and DMAC1	179
6-3	DIC0 and DIC1	180
6-4	Structure of DMA Service Channel	181
6-5	DMA Address Generation Methods	182
6-6	When in Burst Mode	183
6-7	When in Single-Step Mode (V25)	184
6-8	One Transfer Mode on V25 (memory to I/O with no wait state insertion)	184
6-9	Demand Release Mode (I/O to memory, I/O: one wait state insertion, memory: no wait state insertion)	184
6-10	When in Burst Mode (byte transfer)	185
6-11	When in Single-Step Mode (V35)	186
6-12	One Transfer Mode on V35 (memory to I/O with no wait state insertion)	187
6-13	Demand Release Mode (I/O to memory, I/O: one wait state insertion, memory: no wait state insertion)	188
6-14	Example of IORD and IOWR Signal Generation Circuit	188
7-1	Structure of Ports 0 to 2	191
7-2	Port Set to Output Port Mode	192
7-3	Port Set to Input Port Mode	192
7-4	Port Set to Control Mode	193
7-5	PMC0	194
7-6	PM0	195
7-7	PMC1	196
7-8	PM1	196
7-9	PMC2	197
7-10	PM2	198
7-11	Block Diagram of Port T	199
7-12	PMT	200
8-1	Block Diagram of Clock Generator	203
8-2	Clock Generator External Circuit	204
8-3	PRC	206
9-1	Timer Unit Structure during Interval Timer Mode	207
9-2	Timer Unit Structure during One-shot Timer Mode	208
9-3	Output State of TOUT Pin (during One-shot Timer Mode)	210
9-4	TMC0	213
9-5	TMC1	214
9-6	Interrupt Requests Occurring from Timer Unit	215
9-7	TMIC0, TMIC1, and TMIC2	216
9-8	TMMS0, TMMS1, and TMMS2	216
10-1	Time Base Counter Structure	217
10-2	PRC	218
10-3	TBIC	219
11-1	Serial Interface Function	222
11-2	Format of Send Data	223
11-3	Receive Data Sampling Timing	225

LIST OF FIGURES (3/3)

Figure No.	Title	Page
11-4	SCM0 and SCM1 (when asynchronous mode is set)	231
11-5	SCM0 (when I/O interface mode is set)	232
11-6	SCC0 and SCC1	235
11-7	SCE0 and SCE1	237
11-8	SEICn, SRICn, and STICn (n = 0 or 1)	239
11-9	SRMSn and STMSn (n = 0 or 1)	240
12-1	STBC	241
12-2	HALT Mode Release when Interrupt Request Occurs	243
12-3	Macro Service or DMA Start during HALT Mode	243
12-4	STOP Mode Release by Input to NMI Pin	244
15-1	Operation State Transition Diagram	258
A-1	Normal Operation	260
A-2	Abnormal Operation	261
A-3	Example of Avoidance Process Flow	264

[MEMO]

LIST OF TABLES (1/1)

Table No.	Title	Page
2-1	Operation of Port 0 (n = 0 to 7)	41
2-2	Operation of Port 1 (n = 0 to 7)	42
2-3	Operation of Port 2 (n = 0 to 7)	45
3-1	General Purpose Register Offsets	63
3-2	Pointer and Index Register Offsets	64
3-3	Segment Register Offsets	65
3-4	Special Function Register List	78
4-1	Interrupt Source List	86
4-2	Software Interrupts	115
5-1	Bus Control Pin Functions	135
5-2	V35 Data Access	137
5-3	Address Time-Division Multiplexing Output (V35)	138
5-4	Wait State Settings	140
5-5	Refresh Cycles	158
5-6	Number of Wait States Inserted in Refresh Cycle	159
5-7	REFRQ Signal Output Levels	160
6-1	Transfer Mode Functions	174
7-1	Operation of Port 0 (n = 0 to 7)	194
7-2	Operation of Port 1 (n = 0 to 7)	195
7-3	Operation of Port 2 (n = 0 to 7)	197
8-1	Recommended Ceramic Resonators	205
8-2	Recommended Crystal Resonators	205
9-1	TMn Count Time in Interval Timer Mode (n = 0 or 1)	211
9-2	TM0 and MD0 Count Time in One-shot Timer Mode	212
11-1	Baud Rate Generator Setup Values (for reference)	234
12-1	Operation after HALT Mode Is Released when Interrupt Request Occurs	243
12-2	HALT Mode and STOP Mode	245
13-1	Hardware States after Reset	247
A-1	Instructions Subject to Limitations Concerning CVTBD and CVTDB Instructions	273

[MEMO]

CHAPTER 1 GENERAL DESCRIPTION

The μ PD70320 (or V25) and μ PD70330 (or V35) are single-chip microcontrollers that are software-compatible with the μ PD70108 and 70116 (or V20 and V30, respectively).

The V25 and V35 each contain a 16-bit CPU. The V25 has an 8-bit external data bus and the V35 has a 16-bit external data bus.

The V25 and V35 feature powerful interrupt functions that are important for control applications. Their on-chip peripheral functions include an interval timer, serial interface, and DMA controller.

Both of these products are applicable as main controllers in systems where large amounts of data are processed and many devices are controlled.

In particular, these devices are applicable for controlling systems that handle mass data and devices such as printers, word processors, and other terminals.

1.1 Features

- o Internal 16-bit architecture
- ★ o External data bus width
 - μ PD70320: 8 bits
 - μ PD70330: 16 bits
- o Software compatible with μ PD70108, 70116 in native mode (additional instructions are provided)
- o 3-stage pipeline system
- o Minimum instruction cycle
 - μ PD70320: 400 ns (with external clock of 10 MHz)
 - ★ • μ PD70320-8, 70330-8: 250 ns (with external clock of 16 MHz)
- ★ o Internal memory RAM: 256 bytes
- o Memory space: 1 Mbyte
- o I/O space: 64 Kbytes
- o Register banks (memory-mapped): 8 banks
- o On-chip peripheral hardware mapped in memory (special function registers)
- o Input port with comparator (port T): 8 bits
- o I/O lines Input port: 4 bits
I/O ports: 20 bits
- o Serial interface: 2 channels
 - On-chip dedicated baud rate generator
 - Asynchronous mode, I/O interface mode
- o Interrupt controller
 - Eight programmable priority levels
 - Three interrupt response modes
 - Vectored interrupt function
 - Register bank switching function
 - Macro service function
- o DRAM, pseudo SRAM refresh function
- o DMA controller: 2 channels
 - Four DMA transfer modes
- o 16-bit timer: 2 channels
- o Time base counter (20 bits): 1 channel
- o Programmable wait function
- o Standby function (STOP or HALT)
- o CMOS
- o Packages
 - 84-pin plastic QFJ (Quad Flat J-leaded Package)
 - 94-pin plastic QFP (Quad Flat Package)

1.2 Differences with V25+™ and V35+™

		V25	V35	V25+	V35+
		μPD70320	μPD70330	μPD70325	μPD70335
DMA function	Transfer processing method	Microprogram		Dedicated hardware	
	Maximum transfer rate (@ 8 MHz)	0.6 Mbytes/s	0.8 Mbytes/s	4 Mbytes/s	5.3 Mbytes/s
	DMA request sampling timing	Between instruction execution cycles		Between bus cycles	
	DMA service channel	In on-chip RAM area		In special function register area	
	Transfer address specification method	Segment		Linear	
	Execution form in single step mode	One DMA transfer/one instruction execution		One DMA transfer/one bus cycle	
	Interrupt request during DMA transfer (demand release mode)	Only NMI is acknowledged		Not acknowledged	
	No. of required wait cycles for DMARQ stop control (demand release mode)	Not required		Two wait cycles	
	Transfer processing units	Bytes or words	Bytes or words	Bytes	Bytes or words
	TC (Terminal Counter) setup value	DMA transfer count		(DMA transfer count) – 1	
	Terminal count generation timing	TC = 0		TC = FFFFH	
	TC output low level width	Fixed		Widened by wait insertion	
Serial interface	Transmission clock output (channel 0) in asynchronous mode	Disabled		Enabled ($\overline{\text{SCK0}}$ pin)	
	Serial error register	Available		Serial status register	
	Receive buffer full flag	Not available		In serial status register	
	Transmit buffer empty flag	Not available		In serial status register	
	All sent flag	Not available		In serial status register	
Interrupt function	Interrupt cause register	Not available		Available	
Maximum operation frequency		8 MHz		10 MHz	

★ 1.3 Ordering Information

Part number	External data bus (bits)	Package	Maximum operation frequency (MHz)
μ PD70320L	8	84-pin plastic QFJ (1150 × 1150 mil)	5
μ PD70320L-8	8	84-pin plastic QFJ (1150 × 1150 mil)	8
μ PD70320GJ-5BG	8	94-pin plastic QFP (20 × 20 mm)	5
μ PD70320GJ-8-5BG	8	94-pin plastic QFP (20 × 20 mm)	8
μ PD70330L-8	16	84-pin plastic QFJ (1150 × 1150 mil)	8
μ PD70330GJ-8-5BG	16	94-pin plastic QFP (20 × 20 mm)	8

Remark Plastic QFJ is a new name for PLCC (Plastic Leaded Chip Carrier).

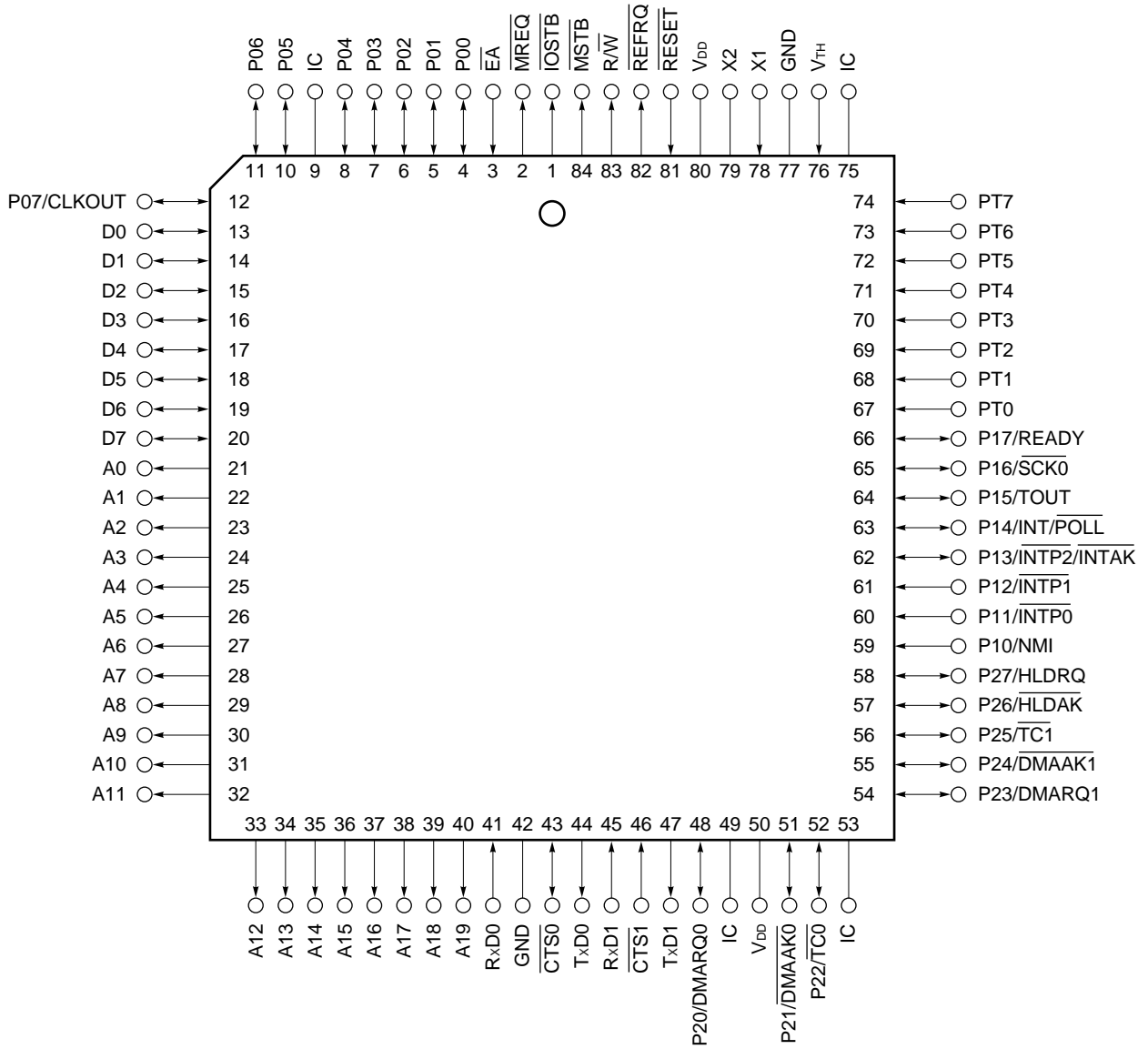
★ 1.4 Pin Configuration (Top View)

★ (1) 84-pin plastic QFJ (1150 × 1150 mil)

(a) V25

μPD70320L

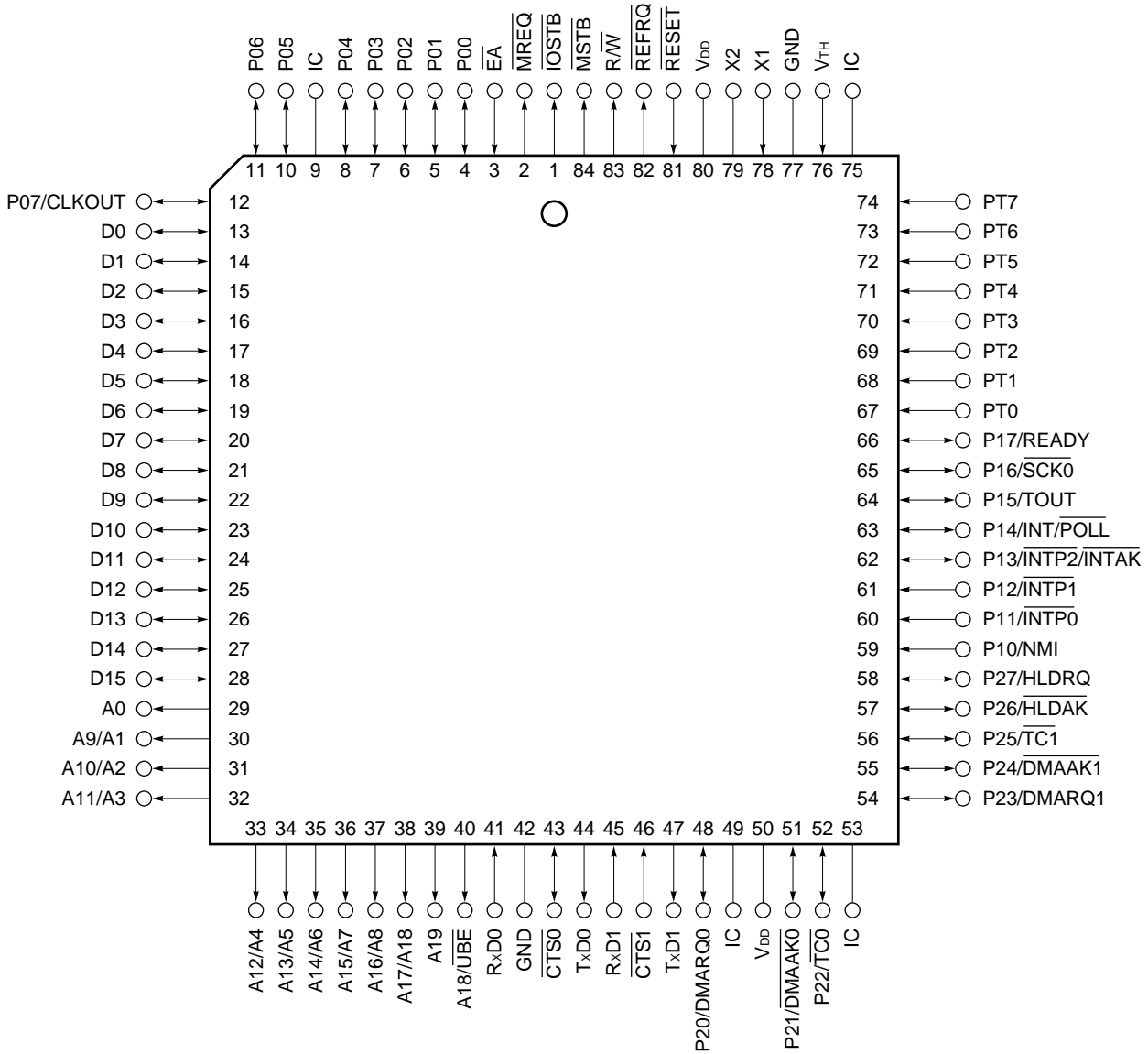
μPD70320L-8



IC : Internally Connected

- Cautions**
1. Connect IC pin to V_{DD} individually via a resistor (3 to 10 kΩ).
 2. Connect the EA pin to GND via a resistor (3 to 10 kΩ).

★ (b) V35
 μPD70330L-8



IC : Internally Connected

- Cautions**
1. Connect IC pin to V_{DD} individually via a resistor (3 to 10 kΩ), except for pin 9, which should be connected to GND via a resistor (3 to 10 kΩ).
 2. Connect the \overline{EA} pin to GND via a resistor (3 to 10 kΩ).

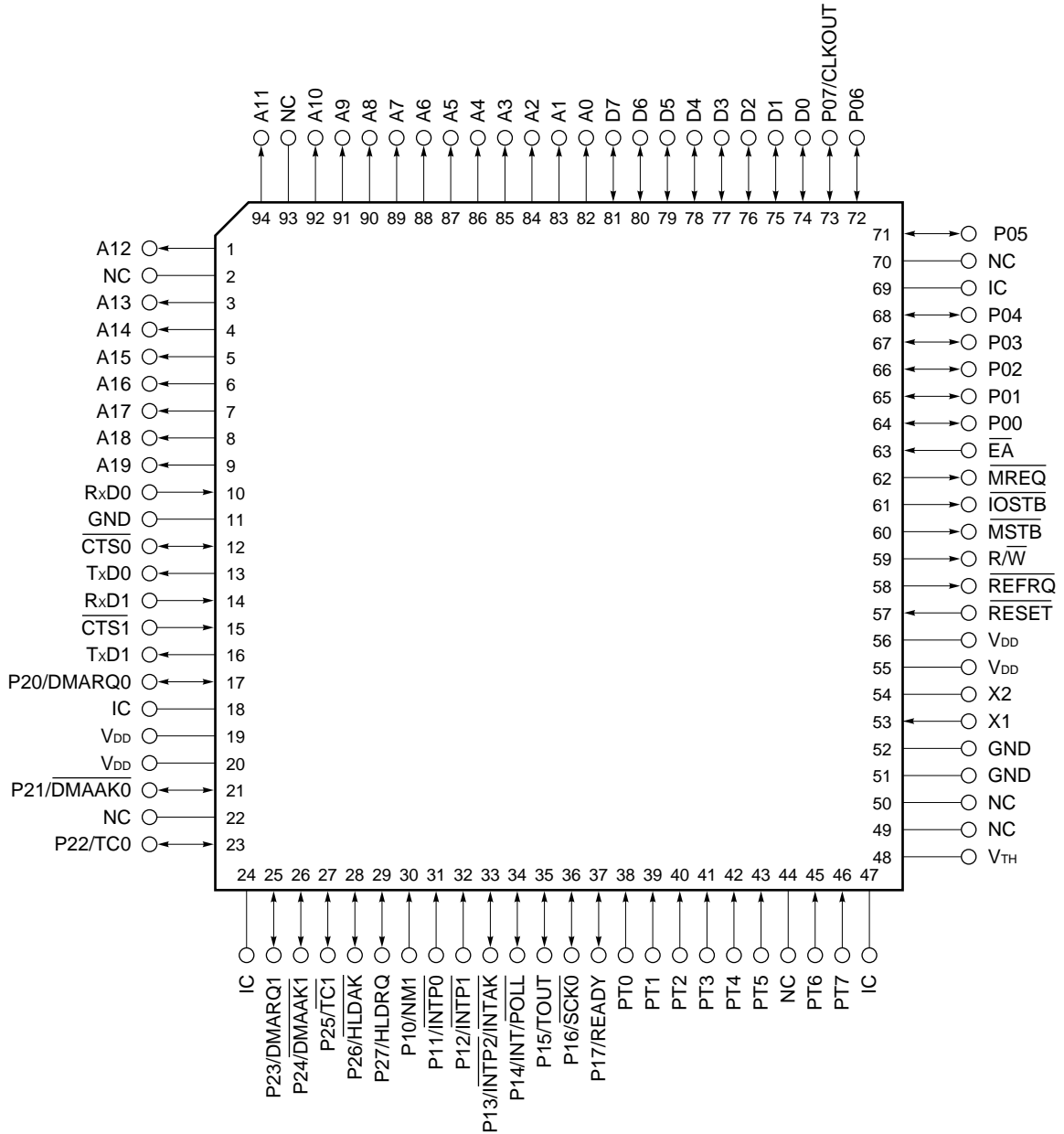
(2) 94-pin plastic QFP (20 × 20 mm)

★

(a) V25

μPD70320GJ-5BG

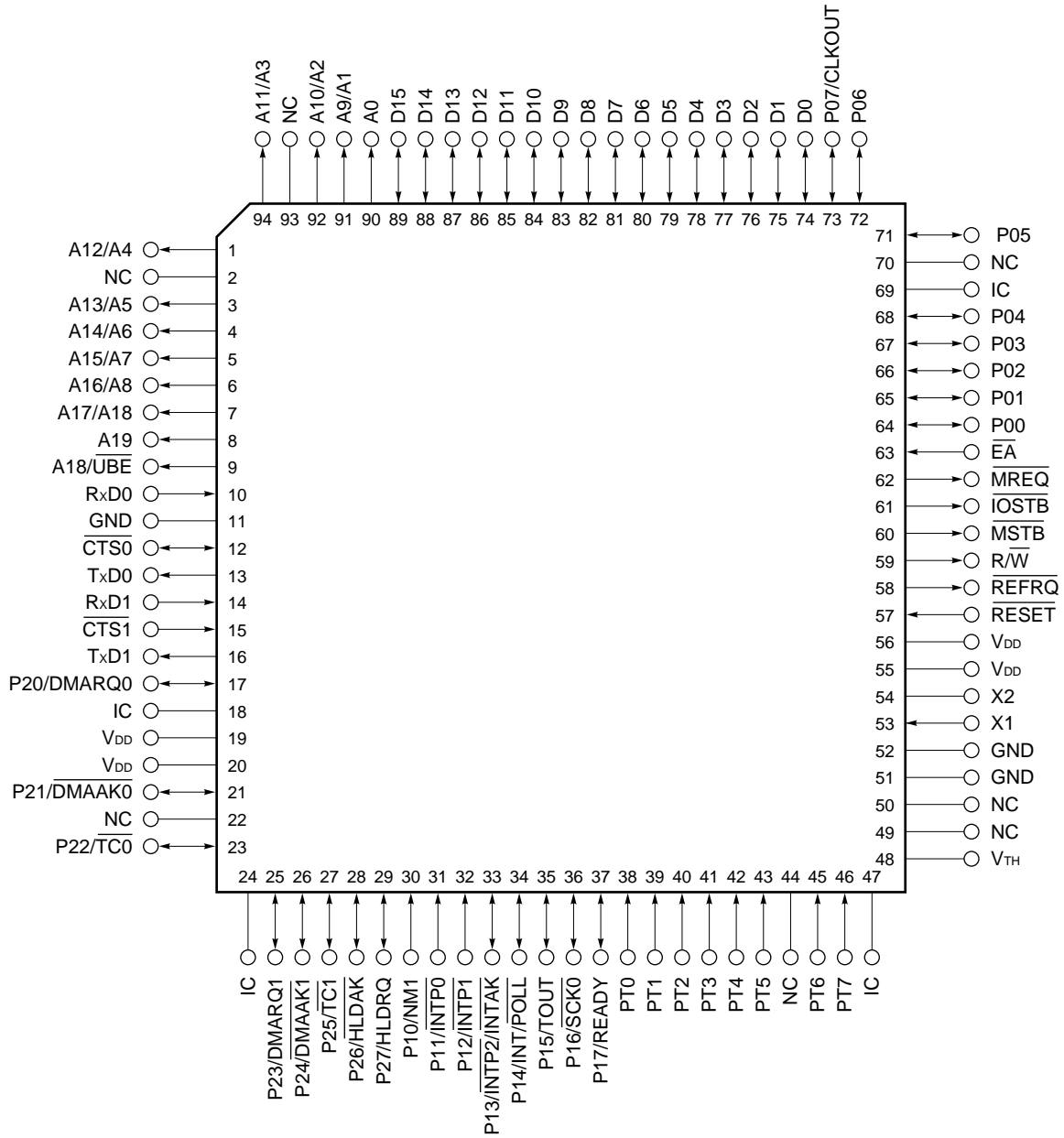
μPD70320GJ-8-5BG



IC : Internally Connected
 NC : Non-Connection

- Cautions**
1. Connect IC pin to V_{DD} individually via a resistor (3 to 10 kΩ).
 2. Connect the \overline{EA} pin to GND via a resistor (3 to 10 kΩ).

★ (b) V35
 μPD70330GJ-8-5BG



IC : Internally Connected
 NC : Non-Connection

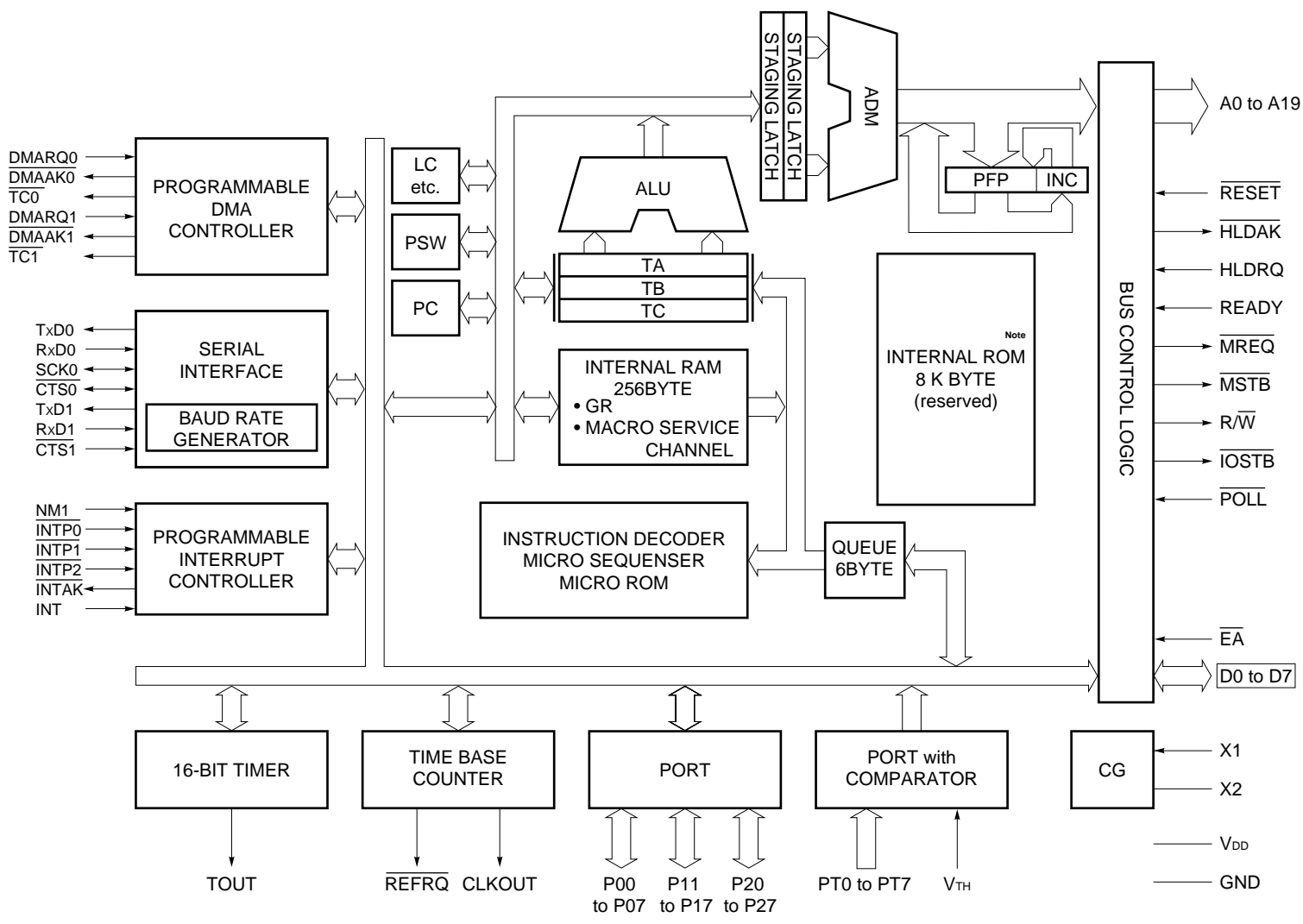
- Cautions**
1. Connect IC pin to V_{DD} individually via a resistor (3 to 10 kΩ).
 2. Connect the $\overline{\text{EA}}$ pin to GND via a resistor (3 to 10 kΩ).

★ **Pin Identifications**

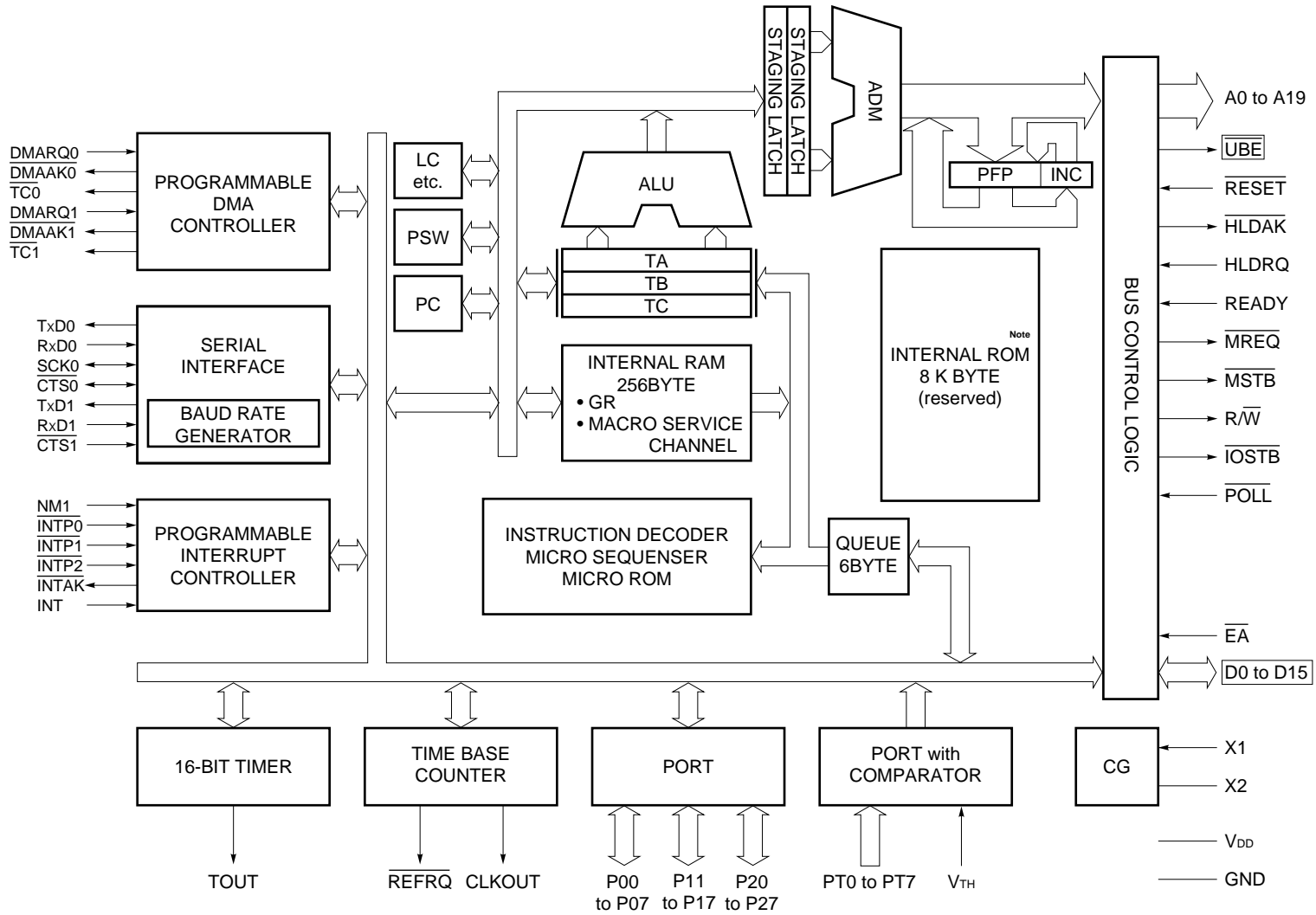
$\overline{\text{IOSTB}}$: I/O Strobe
$\overline{\text{MREQ}}$: Memory Request
P00 to P07	: Port 0
P11 to P17	: Port 1
P20 to P27	: Port 2
CLKOUT	: Clock Out
D0 to D15	: Data Bus
A0 to A19	: Address Bus
RxD0, RxD1	: Receive Data
$\overline{\text{CTS0}}$, $\overline{\text{CTS1}}$: Clear To Send
TxD0, TxD1	: Transfer Data
DMARQ0, DMARQ1	: DMA Request
$\overline{\text{DMAAK0}}$, $\overline{\text{DMAAK1}}$: DMA Acknowledge
$\overline{\text{TC0}}$, $\overline{\text{TC1}}$: Terminal Count
HLDKAK	: Hold Acknowledge
HLDKAK	: Hold Request
NMI	: Non-Maskable Interrupt Request
$\overline{\text{INTP0}}$ to $\overline{\text{INTP2}}$: Interrupt From Peripherals
$\overline{\text{INTAK}}$: Interrupt Acknowledge
INT	: Interrupt Request
$\overline{\text{POLL}}$: Polling
TOUT	: Timer Output
$\overline{\text{SCK0}}$: Serial Clock
READY	: Ready
PT0 to PT7	: Port T
X1, X2	: Crystal
$\overline{\text{RESET}}$: Reset
$\overline{\text{REFRQ}}$: Refresh Request
$\overline{\text{R/W}}$: Read/Write
$\overline{\text{MSTB}}$: Memory Strobe
$\overline{\text{EA}}$: External Access
V _{TH}	: Threshold Voltage
V _{DD}	: Power Supply
GND	: Ground

★ 1.5 Internal Block Diagram

★ (1) V25



★ **Note** User can not use.



★ **Note** User can not use.

[MEMO]

CHAPTER 2 PIN FUNCTIONS

★ 2.1 Pin Function Lists

2.1.1 μ PD70320

(1) Ports

Pin name	I/O	Shared by	Function
P00 to P06	I/O	–	8-bit I/O port for which the input or output mode can be specified bit-wise
P07		CLKOUT	
P10	I	NMI	Cannot be used as a general purpose port (Nonmaskable interrupt request input)
P11		$\overline{\text{INTP0}}$	Cannot be used as a general purpose port (External interrupt request input)
P12		$\overline{\text{INTP1}}$	
P13		$\overline{\text{INTP2/INTAK}}$	
P14	I/O	$\overline{\text{POLL/INT}}$	I/O port for which the input or output mode can be specified bit-wise
P15		TOUT	
P16		$\overline{\text{SCK0}}$	
P17		READY	
P20	I/O	DMARQ0	8-bit I/O port for which the input or output mode can be specified bit-wise
P21		$\overline{\text{DMAAK0}}$	
P22		$\overline{\text{TC0}}$	
P23		DMARQ1	
P24		$\overline{\text{DMAAK1}}$	
P25		$\overline{\text{TC1}}$	
P26		$\overline{\text{HLDK}}$	
P27		HLDRQ	
PT0 to PT7	I	–	Input port with an 8-bit comparator

Remark After reset is released, the port pins become input port pins. To use P13/ $\overline{\text{INTP2/INTAK}}$ as an $\overline{\text{INTAK}}$ pin, be sure to connect a pull-up resistor to this pin to avoid external interrupt controller malfunction after reset is released.

★ (2) Non-port pins

Pin name	I/O	Function	Shared by
$\overline{\text{IOSTB}}$	O	I/O read or I/O write strobe output	–
$\overline{\text{MREQ}}$		Output indicating that memory bus cycle has started	
CLKOUT		System clock output	
D0 to D7	I/O	8-bit data bus	–
A0 to A19			
RxD0	I	Serial data input	
RxD1			
$\overline{\text{CTS0}}$	I/O	Asynchronous mode: CTS input	
		I/O interface mode: reception clock I/O	
$\overline{\text{CTS1}}$	I	CTS input	
TxD0	O	Serial data output	
TxD1			
DMARQ0	I	DMA request input (CH0)	P20
DMARQ1		DMA request input (CH1)	P23
$\overline{\text{DMAAK0}}$	O	DMA acknowledge output (CH0)	P21
$\overline{\text{DMAAK1}}$		DMA acknowledge output (CH1)	P24
$\overline{\text{TC0}}$		DMA termination output (CH0)	P22
$\overline{\text{TC1}}$		DMA termination output (CH1)	P25
$\overline{\text{HLDAK}}$		Hold acknowledge output	P26
HLDRQ	I	Hold request input	P27
NMI		Nonmaskable interrupt request input	P10
$\overline{\text{INTP0}}$		External interrupt request input	P11
$\overline{\text{INTP1}}$			P12
$\overline{\text{INTP2}}$			P13/ $\overline{\text{INTAK}}$
$\overline{\text{INTAK}}$	O	INT acknowledge signal output	P13/ $\overline{\text{INTP2}}$
$\overline{\text{POLL}}$	I	$\overline{\text{POLL}}$ input	P14/INT
INT		External interrupt request input	P14/ $\overline{\text{POLL}}$
TOUT	O	Timer output	P15
$\overline{\text{SCK0}}$		Serial clock output	P16
READY	I	Ready input	P17
X1		Crystal or ceramic resonator connection pin for system clock oscillation. (External clock can be input to the X1 pin, and inverted clock to the X2 pin.)	–
X2			
$\overline{\text{RESET}}$	I	Reset signal input	

Pin name	I/O	Function	Shared by
REFRQ	O	DRAM refresh pulse output	-
$\overline{R/W}$		Read or write cycle identification signal output	
\overline{MSTB}		Memory read or memory write strobe output	
V_{TH}	I	Comparator reference voltage input	
V_{DD}	-	Positive power supply pin (Connect all V_{DD} pins to power supply)	
GND		GND pin (Connect all GND pin to ground)	
IC		Internally connected (Fix high via an external pull-up resistor)	
\overline{EA}	I	External memory access (Connect to GND via a resistor)	

2.1.2 μ PD70330

(1) Ports

Pin name	I/O	Shared by	Function
P00 to P06	I/O	–	8-bit I/O port for which the input or output mode can be specified bit-wise
P07		CLKOUT	
P10	I	NMI	Cannot be used as a general purpose port (Nonmaskable interrupt request input)
P11		$\overline{\text{INTP0}}$	Cannot be used as a general purpose port (External interrupt request input)
P12		$\overline{\text{INTP1}}$	
P13		$\overline{\text{INTP2}}/\overline{\text{INTAK}}$	
P14	I/O	$\overline{\text{POLL}}/\overline{\text{INT}}$	I/O port for which the input or output mode can be specified bit-wise
P15		TOUT	
P16		$\overline{\text{SCK0}}$	
P17		READY	
P20	I/O	DMARQ0	8-bit I/O port for which the input or output mode can be specified bit-wise
P21		$\overline{\text{DMAAK0}}$	
P22		$\overline{\text{TC0}}$	
P23		DMARQ1	
P24		$\overline{\text{DMAAK1}}$	
P25		$\overline{\text{TC1}}$	
P26		$\overline{\text{HLDAK}}$	
P27		HLDRQ	
PT0 to PT7	I	–	Input port with an 8-bit comparator

Remark After reset is released, the port pins become input port pins. To use P13/ $\overline{\text{INTP2}}/\overline{\text{INTAK}}$ as an $\overline{\text{INTAK}}$ pin, be sure to connect a pull-up resistor to this pin to avoid external interrupt controller malfunction after reset is released.

★ (2) Non-port pins

Pin name	I/O	Function	Shared by
$\overline{\text{IOSTB}}$	O	I/O read or I/O write strobe output, and low-order address strobe output	–
$\overline{\text{MREQ}}$		Output indicating that memory bus cycle or I/O bus cycle has started, and high-order address strobe output	
CLKOUT		System clock output	P07
D0 to D15	I/O	16-bit data bus	–
A0	O	LSB of address outputs selecting the low-order memory bank	
A9/A1 to A16/A8, A17/A18, A19		19-bit address output by multiplexer	
$\overline{\text{UBE}}$	O	Used for address's bit 18 output and selection of high-order memory bank	A18
RxD0	I	Serial data input	
RxD1			
$\overline{\text{CTS0}}$	I/O	Asynchronous mode: CTS input I/O interface mode: reception clock I/O	
$\overline{\text{CTS1}}$	I	CTS input	
TxD0	O	Serial data output	
TxD1			
DMARQ0	I	DMA request input (CH0)	P20
DMARQ1		DMA request input (CH1)	P23
$\overline{\text{DMAAK0}}$	O	DMA acknowledge output (CH0)	P21
$\overline{\text{DMAAK1}}$		DMA acknowledge output (CH1)	P24
$\overline{\text{TC0}}$		DMA termination output (CH0)	P22
$\overline{\text{TC1}}$		DMA termination output (CH1)	P25
$\overline{\text{HLDAK}}$		Hold acknowledge output	P26
HLDRQ	I	Hold request input	P27
NMI		Nonmaskable interrupt request input	P10
$\overline{\text{INTP0}}$		External interrupt request input	P11
$\overline{\text{INTP1}}$			P12
$\overline{\text{INTP2}}$			P13/ $\overline{\text{INTAK}}$
$\overline{\text{INTAK}}$	O	INT acknowledge signal output	P13/ $\overline{\text{INTP2}}$
$\overline{\text{POLL}}$	I	$\overline{\text{POLL}}$ input	P14/ $\overline{\text{INT}}$
INT		External interrupt request input	P14/ $\overline{\text{POLL}}$

Pin name	I/O	Function	Shared by
TOUT	O	Timer output	P15
$\overline{\text{SCK0}}$		Serial clock output	P16
READY	I	Ready input	P17
X1		Crystal or ceramic resonator connection pin for system clock oscillation. (External clock can be input to the X1 pin, and inverted clock to the X2 pin.)	-
X2	-		
$\overline{\text{RESET}}$	I	Reset signal input	
$\overline{\text{REFRQ}}$	O	DRAM refresh pulse output	
$\overline{\text{R/W}}$		Read or write cycle identification signal output	
$\overline{\text{MSTB}}$		Memory read or memory write strobe output, and low-order address strobe output	
V _{TH}	I	Comparator reference voltage input	
V _{DD}	-	Positive power supply pin (Connect all V _{DD} pins to power supply)	
GND		GND pin (Connect all GND pins to ground)	
IC		Internally connected (Fix high via an external pull-up resistor)	
$\overline{\text{EA}}$	I	External memory access (Connect to GND via a resistor)	

2.1.3 Pin state in each mode

Pin		Operation state	Hold	HALT	STOP	During reset	Other
Port	During output in port mode		Held	Held	Held	Hi-z	Continued
	During output in control mode		Continued	Continued	Held	Hi-z	Continued
TxD0, TxD1			Continued	Continued	Held	Hi-z	Continued
$\overline{\text{CTS0}}$			Continued	Continued	Held	Hi-z	Continued
D0 to D15			Hi-z	Hi-z	Hi-z	Hi-z	Continued
A0 to A19, $\overline{\text{UBE}}$			Hi-z	Held ^{Note}	Held ^{Note}	Hi-z	Continued
$\overline{\text{MREQ}}$			Hi-z	High	High	Hi-z	Continued
$\overline{\text{MSTB}}$			Hi-z	High	High	Hi-z	Continued
$\overline{\text{R/W}}$			Hi-z	High	High	Hi-z	Continued
$\overline{\text{REFRQ}}$			Hi-z	Continued	Held	Hi-z	Continued
$\overline{\text{IOSTB}}$			Hi-z	High	High	Hi-z	Continued

Continued: Specified operation is continued.

Held: The state just before the mode transition is held.

Hi-Z: High impedance

Note Address data is held, but output address is undefined.

★ **2.2 Description of Pin Functions**

The V25 and V35 include both dedicated control pins and pins that have dual functions as ports. For details of the setup method for pins that have dual functions as ports and for information on the hardware configuration, see **CHAPTER 7 PORT FUNCTIONS**.

2.2.1 P00 to P07 (Port 0) ... 3-state input/output

P00 to P07 are 8-bit I/O pins for port 0 (8-bit I/O port with output latch). The P07 pin also serves as CLKOUT output.

The P00 to P06 pins operate only in the port mode. The port or control mode can be selected for the P07 pin by setting the port 0 mode control register (PMC0). (See **Table 2-1**.)

Table 2-1. Operation of Port 0 (n = 0 to 7)

	PMC0n = 1	PMC0n = 0	
		PM0n = 1	PM0n = 0
P00	—	Input port	Output port
P01	—	Input port	Output port
P02	—	Input port	Output port
P03	—	Input port	Output port
P04	—	Input port	Output port
P05	—	Input port	Output port
P06	—	Input port	Output port
P07	CLKOUT output	Input port	Output port

(1) Port mode

The input or output mode can be selected for the P00 to P07 pins bit-wise by setting the port 0 mode register (PM0). However, the P07 pin is set to the port mode by setting a value of 0 for PMC0 register bit 7.

(2) Control mode

When the P07 pin is set to the control mode by setting a value of 1 for the PMC0 register bit 7, the pin serves as a CLKOUT output.

(a) CLKOUT (Clock Out) ... output also used for P07

CLKOUT is a system clock (CLK) output pin that supplies various clocks to the CPU and peripheral hardware at the frequency set by the clock generator.

In the hold or standby mode, the pins set as output ports hold the immediately preceding data. When the P07 pin is set to CLKOUT, it continues clock output in the hold or HALT mode and holds the immediately preceding data in the STOP mode.

When $\overline{\text{RESET}}$ is input, the P00 to P07 pins operate as an input port (high impedance output).

2.2.2 P10 to P17 (Port 1) ... 3-state I/O

P10 to P17 are 8-bit I/O pins used for both port 1 (8-bit I/O with output latch) and for various control signals.

The port or control mode can be selected for the P10 to P17 pins bit-wise by setting the port 1 mode control register (PMC1). (See **Table 2-2.**) However, the P11 and P12 pins operate only in the port mode and also serve as interrupt request input. The P10 pin serves only as an NMI input.

Table 2-2. Operation of Port 1 (n = 0 to 7)

	PMC1n = 1	PMC1n = 0	
		PM1n = 1	PM1n = 0
P10	—	NMI input	—
P11	—	$\overline{\text{INTP0}}$ input	—
P12	—	$\overline{\text{INTP1}}$ input	—
P13	$\overline{\text{INTAK}}$ output	$\overline{\text{INTP2}}$ input	—
P14	INT input	Input port ($\overline{\text{POLL}}$ input)	Output port
P15	TOUT output	Input port	Output port
P16	$\overline{\text{SCK0}}$ output	Input port	Output port
P17	READY input	Input port	Output port

(1) Port mode

When the P10 to P17 pins are set to the port mode by setting the PMC1 register (PMC1n = 0; n = 0 to 7), the input or output mode can be selected for the pins bit-wise by setting the port 1 mode register (PM1). The P10 to P13 pins can operate only as an input port and can also serve as interrupt request input. When the P14 pin is set to an input port, the pin also serves as a $\overline{\text{POLL}}$ input.

(a) NMI (Non-Maskable Interrupt) ... P10 input

NMI is an input pin for nonmaskable interrupt requests; this pin cannot be masked by software.

Nonmaskable interrupts are acknowledged by the CPU at any time, and they take precedence over any other interrupt.

NMI input is edge-triggered, with the valid edge specified in the external interrupt mode register (INTM).

The input is sampled in each clock cycle. NMI is acknowledged when the pin level is changed from inactive to active level and continues the active level for a certain period or more. When NMI is acknowledged, an interrupt of vector number 2 occurs after the instruction being executed terminates. The NMI input is also used to release the CPU standby mode. The NMI input state can be monitored by reading P10.

(b) $\overline{\text{INTP0}}$ to $\overline{\text{INTP2}}$ (Interrupt from Peripheral 0 to 2) ... input also used for P11 to P13

These are external interrupt request input pins that can be masked by software.

$\overline{\text{INTPn}}$ ($n = 0$ to 2) is edge-triggered, with the valid edge specified in the external interrupt mode register (INTM). The input is sampled in each clock cycle. INTPn ($n = 0$ to 2) is acknowledged when the pin level is changed from inactive to active level and continues the active level for a certain period or more.

The $\overline{\text{INTPn}}$ input is also used to release the HALT mode.

(c) $\overline{\text{POLL}}$ (Poll) ... input also used for P14/INT

$\overline{\text{POLL}}$ input is checked by the POLL instruction. If it is low, the next instruction is executed. If it is high, $\overline{\text{POLL}}$ input is checked at every fifth clock cycle until it goes low.

These functions are used to synchronize the CPU program with external device operations.

Caution The $\overline{\text{POLL}}$ pin functions when the input mode (input port, INT input) has been selected for P14. Otherwise, it is assumed to be low while the POLL instruction is being executed.

(2) Control mode

Pins P13 to P17 can be set to control mode bit-wise by setting the PMC1 register.

(a) $\overline{\text{INTAK}}$ (Interrupt Acknowledge) ... output also used for P13/ $\overline{\text{INTP2}}$

$\overline{\text{INTAK}}$ is an acknowledge signal output pin for interrupt request input (INT) that can be masked by software.

When the CPU acknowledges the INT signal, the $\overline{\text{INTAK}}$ signal is output low. In synchronization with the signal, the external device sends an interrupt vector to the CPU via the data bus (D0 to D15).

$\overline{\text{INTAK}}$ is used in tandem with the INT pin to connect to an interrupt controller such as the $\mu\text{PD71059}$.

(b) INT (Interrupt) ... input also used for P14/ $\overline{\text{POLL}}$

INT is an interrupt request input pin that can be masked by software.

This request input, which is active high, is sampled in the last clock cycle of an instruction. By receiving the $\overline{\text{INTAK}}$ signal that is output by the CPU, the external device recognizes that the INT interrupt request is acknowledged. The INT signal must be held high at least until the first $\overline{\text{INTAK}}$ signal is output.

The INT pin is used in tandem with the $\overline{\text{INTAK}}$ pin to connect to an interrupt controller such as the $\mu\text{PD71059}$.

The INT input is also used to release the HALT mode.

(c) TOUT (Timer Output) ... output also used for P15

TOUT is an output pin from the timer unit (timer 0).

(d) $\overline{\text{SCK0}}$ (Serial Clock 0) ... output also used for P16

$\overline{\text{SCK0}}$ is a serial interface (channel 0) transmission clock output pin.

(e) READY (Ready) ... input also used for P17

READY is an input pin used to control wait state insertion during external bus cycles (except memory refresh cycles).

The READY pin is level-sensitive, and wait states are inserted while the READY pin is low.

Caution The bus cycle always becomes ready when the pin is set to P17.

The pins set as an output port hold the immediately preceding data in the hold or standby mode. Control signal output pins that have been set to the control mode continue operation when in the hold or HALT mode, and hold the immediately preceding data when in the STOP mode.

When RESET is input, the P00 to P17 pins operate as an input port (high impedance output).

2.2.3 P20 to P27 (Port 2) ... 3-state I/O

P20 to P27 are 8-bit I/O pins used for both port 2 (8-bit I/O port with output latch) and for various control signals.

The port or control mode can be selected as the operation mode for P20 to P27 bit-wise by setting the port 2 mode control register (PMC2). (See **Table 2-3**.)

Table 2-3. Operation of Port 2 (n = 0 to 7)

	PMC2n = 1	PMC2n = 0	
		PM2n = 1	PM2n = 0
P20	DMARQ0 input	Input port	Output port
P21	$\overline{\text{DMAAK0}}$ output	Input port	Output port
P22	$\overline{\text{TC0}}$ output	Input port	Output port
P23	DMARQ1 input	Input port	Output port
P24	$\overline{\text{DMAAK1}}$ output	Input port	Output port
P25	$\overline{\text{TC1}}$ output	Input port	Output port
P26	$\overline{\text{HLDAK}}$ output	Input port	Output port
P27	HLDRQ input	Input port	Output port

(1) Port mode

When P20 to P27 are set to the port mode by setting the PMC2 register (PMC2n = 0; n = 0 to 7), the pins can be set bit-wise to input or output port mode by setting the port 2 mode register (PM2).

(2) Control mode

P20 to P27 can be set to control mode bit-wise by setting the PMC2 register (PMC2n = 1; n = 0 to 7).

(a) DMARQ0 and DMARQ1 (DMA Requests 0 and 1) ... input also used for P20 and P23

DMARQ0 and DMARQ1 are DMA request input pins from DMA controllers (channels 0 and 1). These signals are active high.

(b) $\overline{\text{DMAAK0}}$ and $\overline{\text{DMAAK1}}$ (DMA Acknowledge 0 and 1) ... output also used for P21 and P24

$\overline{\text{DMAAK0}}$ and $\overline{\text{DMAAK1}}$ are DMA acknowledge output pins to DMA controllers (channels 0 and 1). However, the signals are not output during DMA transfer between memories (burst mode or single step mode).

These signals are active low.

(c) $\overline{\text{TC0}}$ and $\overline{\text{TC1}}$ (Terminal Count 0 and 1) ... output also used for P22 and P25

$\overline{\text{TC0}}$ and $\overline{\text{TC1}}$ are DMA completion signal output pins to DMA controllers (channels 0 and 1). The signals are output when $\overline{\text{TC0}}$ and $\overline{\text{TC1}}$ in the DMA service channels are set to 0.

These signals are active low.

(d) $\overline{\text{HLD}}\text{AK}$ (Hold Acknowledge) ... output also used for P26

$\overline{\text{HLD}}\text{AK}$ is an acknowledge signal output pin indicating that the V25 or V35 has acknowledged the hold request signal (HLDRQ) and has set the bus for high impedance.

While the signal is active (low), the address bus, data bus, and control bus are placed into high-impedance.

(e) HLDRQ (Hold Request) ... input also used for P27

HLDRQ is a signal input pin for an external device that requests the V25 or V35 to release the address bus, data bus, and control bus.

The HLDRQ input signal is active high.

The pins set as an output port hold the immediately preceding data in the hold or standby mode. Control signal output pins that have been set to the control mode continue operation when in the hold or HALT mode, and hold the immediately preceding data when in the STOP mode.

When $\overline{\text{RESET}}$ is input, the P20 to P27 pins operate as an input port (high impedance output).

2.2.4 PT0 to PT7 (Port with Comparator 0 to 7) ... input

Port T (PT0 to PT7) consists of 8-bit input pins with a comparator, for which the threshold voltage (reference voltage) can be selected from among 16 levels.

2.2.5 V_{TH} (Threshold Voltage) ... input

V_{TH} is a reference voltage input pin for port T.

2.2.6 TxD0 and TxD1 (Transmit Data 0 and 1) ... output

TxD0 and TxD1 are serial data output pins from the serial interface (channels 0 and 1).

In the asynchronous mode, transmit data is transmitted with one data frame consisting of start, character, parity, and stop bits, starting at the least significant bit (LSB). When transmission is disabled or when the serial register does not contain the transmit data, the TxD0 and TxD1 pins become the mark state (1).

In the I/O interface mode (TxD0 pin only), transmit data is fixed to eight bits and is transmitted starting at the most significant bit (MSB).

The TxD0 and TxD1 pins continue operation in the hold or HALT mode and hold the immediately preceding data in the STOP mode. When $\overline{\text{RESET}}$ is input, the pins become high impedance.

2.2.7 RxD0 and RxD1 (Receive Data 0 and 1) ... input

RxD0 and RxD1 are serial data input pins to the serial interface (channels 0 and 1).

In the asynchronous mode, when RxD0 or RxD1 input is detected low in the reception enable state, it is recognized as a start bit and reception operation is performed.

In the I/O interface mode (RxD0 pin only), receive data is input to the serial register in synchronization with the reception clock's rising edge.

2.2.8 $\overline{\text{CTS0}}$ (Clear to Send 0) ... I/O

$\overline{\text{CTS0}}$ is a serial interface (channel 0) CTS pin.

In the asynchronous mode, the $\overline{\text{CTS0}}$ pin is an active low input pin that enables transmission (clear to send). In the I/O interface mode, it becomes an I/O pin for the reception clock.

The $\overline{\text{CTS0}}$ pin continues operation in the hold or HALT mode and holds the immediately preceding data in the STOP mode. When $\overline{\text{RESET}}$ is input, $\overline{\text{CTS0}}$ becomes an input pin in the I/O interface mode (high impedance output).

2.2.9 $\overline{\text{CTS1}}$ (Clear to Send 1) ... input

$\overline{\text{CTS1}}$ is a serial interface (channel 1) CTS pin.

The $\overline{\text{CTS1}}$ pin is an active-low input pin that enables transmission (clear to send) when in asynchronous mode.

2.2.10 $\overline{\text{RESET}}$ (Reset) ... input

$\overline{\text{RESET}}$ is an active-low reset input pin.

$\overline{\text{RESET}}$ input is an asynchronous input. When a signal having a given low level width is input independently of the operation clock, system reset takes precedence over any other operation that is to be performed.

In addition to normal initialization and start, the $\overline{\text{RESET}}$ pin is also used for standby (STOP or HALT) mode release.

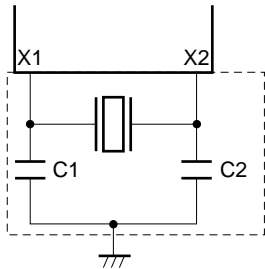
2.2.11 $\overline{\text{EA}}$ (External Access) ... input

$\overline{\text{EA}}$ is an input pin for memory access. Connect this pin to GND via a resistor (3 to 10 k Ω).

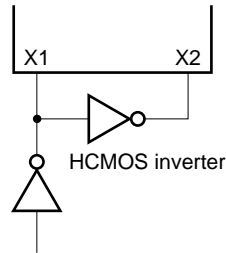
2.2.12 X1 and X2 (Crystal)

X1 and X2 are crystal or ceramic resonator connection pins for system clock generation. An external clock can also be input.

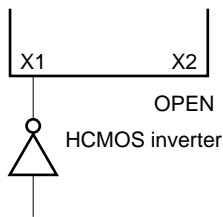
(a) Crystal or ceramic oscillation Note



(b) External clock input



(c) External clock input (X2: Open)



- Cautions**
1. Place the oscillator as close as possible to X1 and X2.
 2. Do not pass any of the signal lines through the section in the broken line.

Note When considering a match for this device, use a ceramic resonator and external capacitors recommended by NEC Corporation for better device matching.

2.2.13 D0 to D15 (Data Bus 0 to 15) ... I/O

D0 to D15 are data bus I/O pins.

In the hold or standby mode or when $\overline{\text{RESET}}$ is input, all of these pins are placed into high impedance.

2.2.14 A0 to A19 (Address Bus 0 to 19) ... output

A0 to A19 are 20-bit address bus output pins.

In the standby mode, address data is held but the output address is undefined. In the hold mode or when $\overline{\text{RESET}}$ is input, A0 to A19 are placed into high impedance.

2.2.15 $\overline{\text{MREQ}}$ (Memory Request) ... output

$\overline{\text{MREQ}}$ is an active low output pin which indicates that a memory bus cycle has started and that the memory address on the address bus is valid.

In the standby mode, $\overline{\text{MREQ}}$ is fixed high. In the hold mode or when $\overline{\text{RESET}}$ is input, $\overline{\text{MREQ}}$ is placed into high impedance.

2.2.16 $\overline{\text{MSTB}}$ (Memory Strobe) ... output

The $\overline{\text{MSTB}}$ pin is an output pin for controlling memory access in combination with the $\overline{\text{MREQ}}$ and $\text{R}/\overline{\text{W}}$ signals.

This pin indicates that the data output on the data bus is valid during memory write. $\overline{\text{MSTB}}$ output differs from the $\overline{\text{MREQ}}$ signal only in its fall timing.

In the standby mode, $\overline{\text{MSTB}}$ is fixed high. In the hold mode or when $\overline{\text{RESET}}$ is input, $\overline{\text{MSTB}}$ is placed into high impedance.

2.2.17 $\text{R}/\overline{\text{W}}$ (Read/Write Strobe) ... output

$\text{R}/\overline{\text{W}}$ is a signal output pin for identifying memory read or memory write cycles when a memory bus cycle has started.

For $\text{R}/\overline{\text{W}}$, output high indicates a memory read cycle and output low indicates a memory write cycle.

In the standby mode, $\text{R}/\overline{\text{W}}$ is fixed high. In the hold mode or when $\overline{\text{RESET}}$ is input, $\text{R}/\overline{\text{W}}$ is placed into high impedance.

2.2.18 $\overline{\text{REFRQ}}$ (Refresh Request) ... output

$\overline{\text{REFRQ}}$ is a refresh pulse output pin.

$\overline{\text{REFRQ}}$ pin output is controlled by the refresh mode register (RFM) contents. In the HALT mode, the refresh operation is continued. In the STOP mode, the immediately preceding data is held. In the hold mode or when $\overline{\text{RESET}}$ is input, $\overline{\text{REFRQ}}$ is placed into high impedance.

2.2.19 $\overline{\text{IOSTB}}$ (I/O Strobe) ... output

$\overline{\text{IOSTB}}$ is an active-low output pin indicating that an I/O bus cycle has started.

When the $\overline{\text{IOSTB}}$ pin is low, it indicates that the I/O address output to the A0 to A15 pins is valid.

In the standby mode, the pin is fixed high. In the hold mode or when $\overline{\text{RESET}}$ is input, $\overline{\text{IOSTB}}$ is placed into high impedance.

2.2.20 V_{DD} (Power Supply)

V_{DD} is a positive power supply pin.

Apply the rated power supply voltage to both V_{DD} pins.

2.2.21 GND (Ground)

GND is a GND potential pin.

Fix all GND pins to the GND potential.

2.2.22 $\overline{\text{UBE}}$ (Upper Byte Enable) ... input

$\overline{\text{UBE}}$ is a high-order memory bank selection pin on the V35. It is also used for A18.

This signal becomes active during the following bus cycles.

- Byte access to an odd address, or first byte access to an odd address for word data access
- Word data access to an even address

Access		$\overline{\text{UBE}}$	A0
Word access to even address		0	0
Word access to odd address	First	0	1
	Second	1	0
Byte access to even address		1	0
Byte access to odd address		0	1

The operation described above is also effective in the I/O cycle.

2.3 Pin I/O Circuits

Section 2.3.1 lists the I/O circuit types. Section 2.3.2 shows the circuits for each type.

2.3.1 I/O circuit types

(1) Ports

Pin name	Type	Pin name	Type
P00 to P06	5	P20/DMARQ0	5
P07/CLKOUT	5	P21/ $\overline{\text{DMAAK0}}$	5
P10/NMI	2	P22/ $\overline{\text{TC0}}$	5
P11/ $\overline{\text{INTP0}}$	1	P23/DMARQ1	5
P12/ $\overline{\text{INTP1}}$	1	P24/ $\overline{\text{DMAAK1}}$	5
P13/ $\overline{\text{INTP2}}/\overline{\text{INTAK}}$	5	P25/ $\overline{\text{TC1}}$	5
P14/INT/ $\overline{\text{POLL}}$	5	P26/ $\overline{\text{HLDAK}}$	5
P15/TOOUT	5	P27/HLDRQ	5
P16/ $\overline{\text{SCK0}}$	5	PT0 to PT7	7
P17/READY	5		

★ (2) Non-port pins

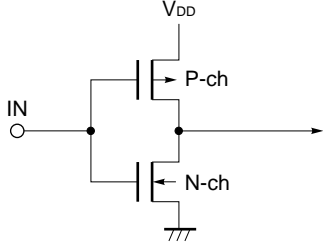
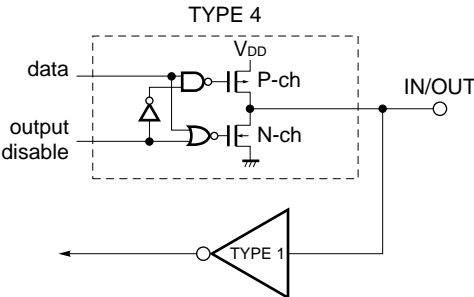
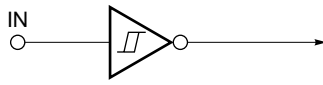
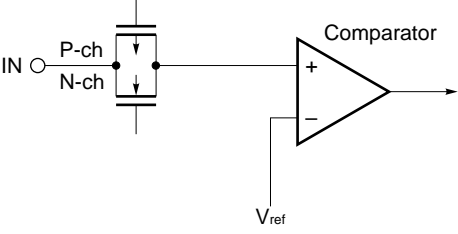
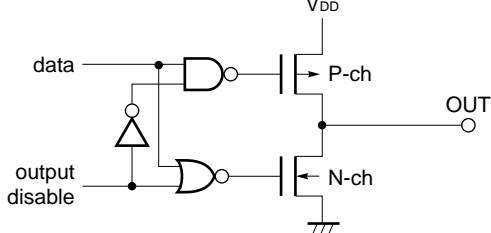
(a) $\mu\text{PD70320}$

Pin name	Type	Pin name	Type
TxD0	4	$\overline{\text{EA}}$	1
TxD1	4	D0 to D7	5
RxD0	1	A0 to A19	4
RxD1	1	$\overline{\text{MREQ}}$	4
CTS0	5	$\overline{\text{MSTB}}$	4
$\overline{\text{CTS1}}$	1	R/ $\overline{\text{W}}$	4
$\overline{\text{REFRQ}}$	4	$\overline{\text{IOSTB}}$	4
$\overline{\text{RESET}}$	2		

(b) μ PD70330

Pin name	Type	Pin name	Type
TxD0	4	A0	4
TxD1	4	A9/A1 to A16/A8	4
RxD0	1	A17/A18	4
RxD1	1	A19	4
$\overline{\text{CTS}}_0$	5	A18/ $\overline{\text{UB}}\overline{\text{E}}$	4
$\overline{\text{CTS}}_1$	1	$\overline{\text{MREQ}}$	4
$\overline{\text{REFRQ}}$	4	$\overline{\text{MSTB}}$	4
$\overline{\text{RESET}}$	2	R/ $\overline{\text{W}}$	4
$\overline{\text{EA}}$	1	$\overline{\text{IOSTB}}$	4
D0 to D15	5		

2.3.2 I/O circuits

<p>TYPE 1</p> 	<p>TYPE 5</p>  <p>I/O circuit consisting of type-4 push-pull output and type-1 input buffer.</p>
<p>TYPE 2</p>  <p>Schmitt trigger input having hysteresis characteristics^{Note}.</p>	<p>TYPE 7</p>  <p>(Threshold voltage)</p>
<p>TYPE 4</p>  <p>Push-pull output that can be set to high-impedance output (both P-ch and N-ch off).</p>	

Note These circuits have been designed for light hysteresis characteristics. However, if the characteristics required for input to these pins are not satisfied, be sure to connect externally to a device that has the required hysteresis characteristics.

2.4 Unused Pin Connections

When connecting to V_{DD} or GND via a resistor, use 3 to 10 k Ω of resistance. Recommended connection methods for unused pins are listed below.

(1) Ports

Pin name	Recommended connection	
P00 to P06	Input state: Connect to V_{DD} individually via a resistor.	
P07/CLKOUT	Output state: Leave open	
P10/NMI	Connect to GND via a resistor.	
P11/ $\overline{\text{INTP0}}$	Connect to V_{DD} or GND via a resistor.	
P12/ $\overline{\text{INTP1}}$		
P13/ $\overline{\text{INTP2}}$ / $\overline{\text{INTAK}}$	Input state: Connect to V_{DD} individually via a resistor.	
P14/INT/ $\overline{\text{POLL}}$	Output state: Leave open	
P15/TOUT		
P16/ $\overline{\text{SCK0}}$		
P17/READY		
P20/DMARQ0		
P21/ $\overline{\text{DMAAK0}}$		
P22/ $\overline{\text{TC0}}$		
P23/DMARQ1		
P24/ $\overline{\text{DMAAK1}}$		
P25/ $\overline{\text{TC1}}$		
P26/ $\overline{\text{HLDAK}}$		
P27/ $\overline{\text{HLDRQ}}$		
PT0 to PT7		Connect to GND via a resistor.

Remark After reset is released, the port pins become input port pins.

(2) Non-port pins

(a) V25

Pin name	Recommended connection
TxD0	Leave open
TxD1	
RxD0	Connect to V _{DD} or GND via a resistor.
RxD1	
$\overline{\text{CTS0}}$	Input state: Connect to V _{DD} individually via a resistor. Output state: Leave open
$\overline{\text{CTS1}}$	Connect to V _{DD} or GND via a resistor.
$\overline{\text{REFRQ}}$	Leave open
V _{TH}	Connect to GND via a resistor.
D0 to D7	Input state: Connect to V _{DD} individually via a resistor. Output state: Leave open
A0 to A19	Leave open
$\overline{\text{MREQ}}$	
$\overline{\text{MSTB}}$	
R/ $\overline{\text{W}}$	
$\overline{\text{IOSTB}}$	

(b) V35

Pin name	Recommended connection
TxD0	Leave open
TxD1	
RxD0	Connect to V _{DD} or GND via a resistor.
RxD1	
$\overline{\text{CTS0}}$	Input state: Connect to V _{DD} individually via a resistor. Output state: No connection
$\overline{\text{CTS1}}$	Connect to V _{DD} or GND via a resistor.
$\overline{\text{REFRQ}}$	Leave open
V _{TH}	Connect to GND via a resistor.
D0 to D15	Input state: Connect to V _{DD} individually via a resistor. Output state: Leave open
A0	Leave open
A9/A1 to A16/A8	
A17/A18	
A19	
A18/ $\overline{\text{UBE}}$	
$\overline{\text{MREQ}}$	
$\overline{\text{MSTB}}$	
R/ $\overline{\text{W}}$	
$\overline{\text{IOSTB}}$	

CHAPTER 3 INTERNAL CPU FUNCTIONS

The V25 and V35 have a 16-bit CPU which is software-compatible with the V20 and V30 CPU in the native mode.

3.1 Hardware Configuration

The internal CPU in the V25 and V35 can be broadly divided into the three functional units described below. These units function together to perform processing in pipeline mode, which makes for efficient bus usage and high-speed instruction execution.

- PAU (Address calculation unit)
- EXU (Execution unit)
- BCU (Bus control unit)

3.1.1 PAU (Address calculation unit)

The PAU generates a 20-bit physical address from information provided by the EXU and then requests the BCU to be started.

3.1.2 EXU (Execution unit)

The EXU performs basic processing related to instruction execution, such as arithmetic and logical operations and data transfer. A pipeline stage for the EXU consists of 2 to 4 clock cycles, and each instruction is executed in one or several pipeline stages.

★ 3.1.3 BCU (Bus control unit)

The BCU starts a required bus cycle based on the 20-bit physical address generated by the PAU. At the same time, it executes bus control, such as for READY, hold, and refresh.

When the PAU does not send a bus cycle start request to the BCU, the BCU generates an opcode prefetch address and prefetches an instruction. The prefetched opcode is stored in the prefetch queue.

A pipeline stage for the BCU is performed in one bus cycle.

Prefetch queue

The V25 and V35 have a six-byte instruction queue (FIFO) that can store a maximum of six bytes of the opcodes prefetched by the BCU.

The opcode stored in the queue is fetched by the EXU for execution.

When a branch, call, return, or break instruction is executed or when external interrupt servicing is performed, the queue contents are cleared and an instruction at a new location is prefetched.

Normally, an opcode is prefetched when the queue contains one or more bytes of free space.

If the average execution time of consecutively executed instructions exceeds the number of clock cycles required to prefetch the opcode of each instruction to some degree, when the EXU terminates execution of one instruction it can immediately execute the opcode stored in the queue. The fetch time from memory can thus be excluded from the instruction execution time.

However, the queue's effectiveness is reduced in the following cases.

- When a number of queue clear instructions are executed, such as when a branch instruction is executed.
- When instructions having short execution times are executed consecutively.

3.1.4 CPU pipelines

The V25 and V35 CPU uses a synchronous pipeline structure. Accordingly, after processing of one pipeline stage is completed by each of the units (PAU, EXU, and BCU) that have been started simultaneously, the next stage is started. Figure 3-1 shows an example of pipeline operations.

Example Pipeline operation when the following instructions are executed.

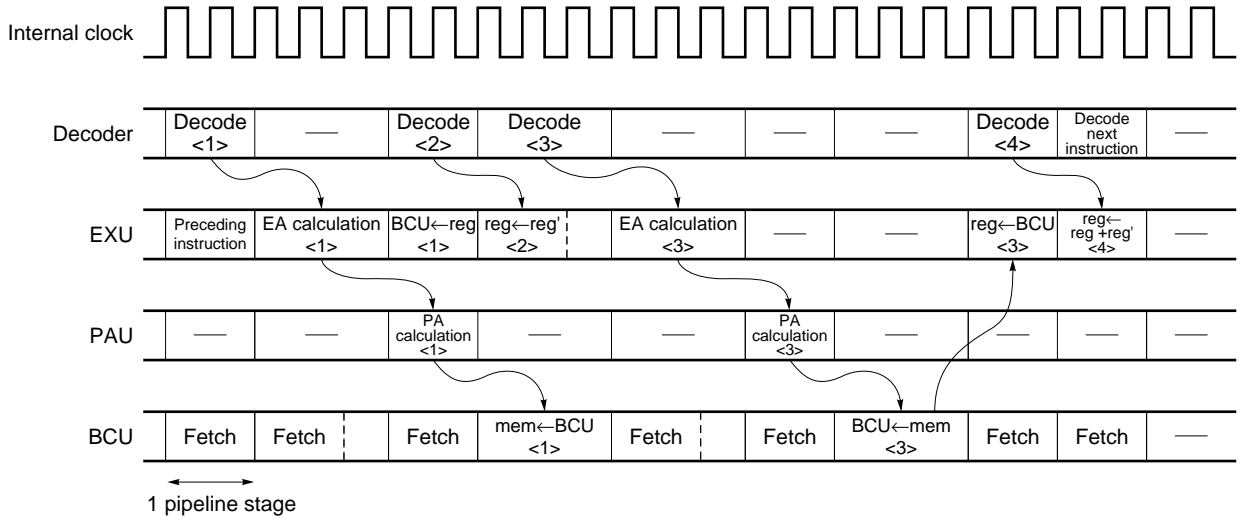
:			
:			
MOV	mem, reg	... <1>	
MOV	reg, reg'	... <2>	
MOV	reg, mem	... <3>	
ADD	reg, reg'	... <4>	
:			
:			

This example assumes the following conditions.

- Prefetch cycle: 2 clocks (0 wait)
- Memory read/write: 3 clocks (1 wait)

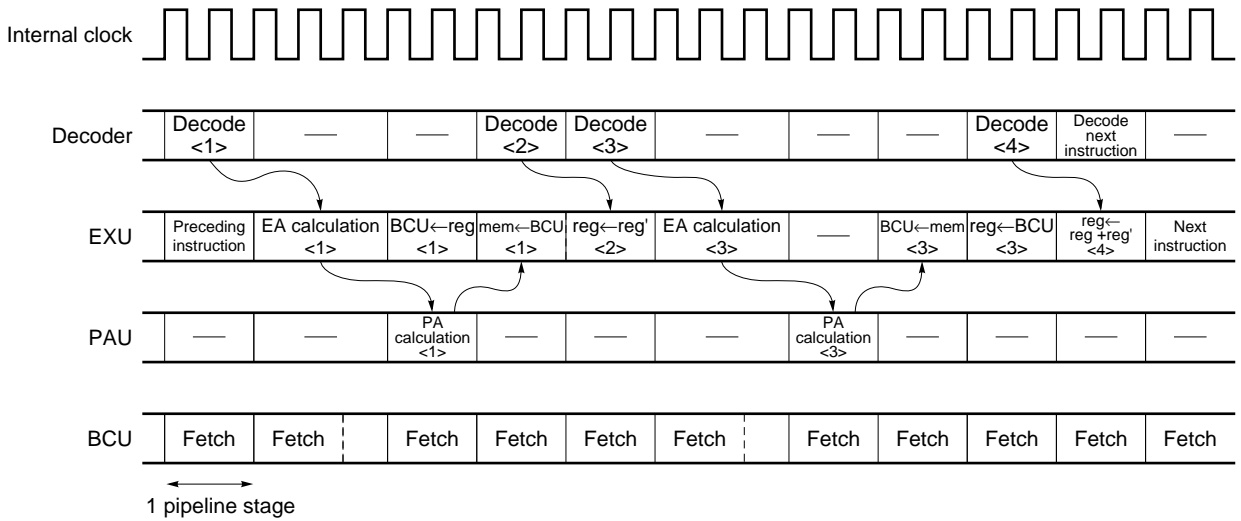
Figure 3-1. CPU Pipeline Operation Example (1/2)

(a) When internal RAM access is disabled



Remark The next instruction is executed even during the BCU's memory write operation.

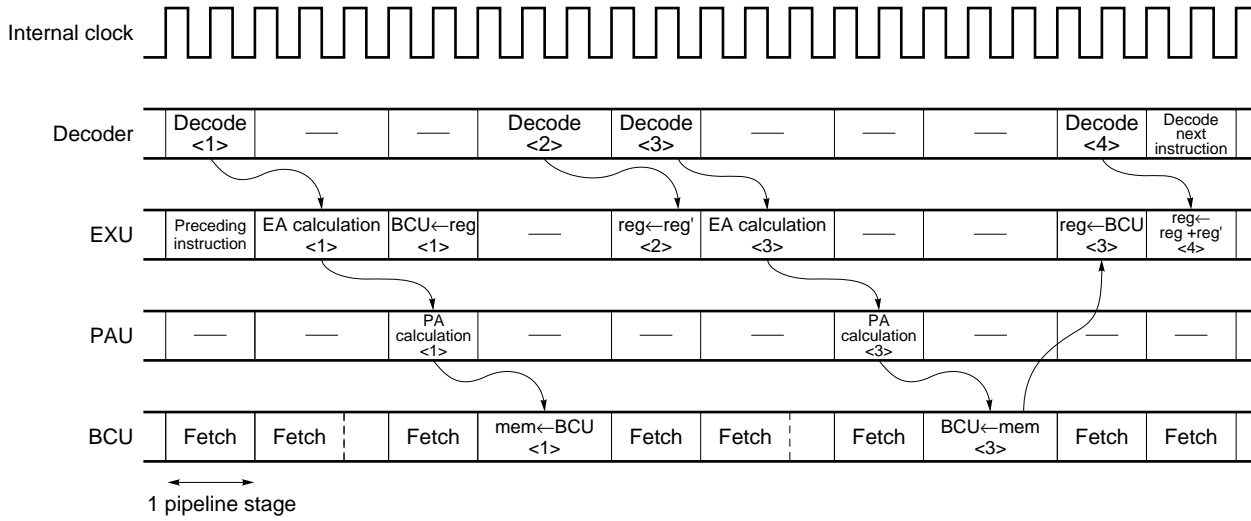
(b) When internal RAM access is enabled (access to internal RAM)



Remark The EXU executes memory write and memory read access to internal RAM. During this operation, the BCU executes program fetch.

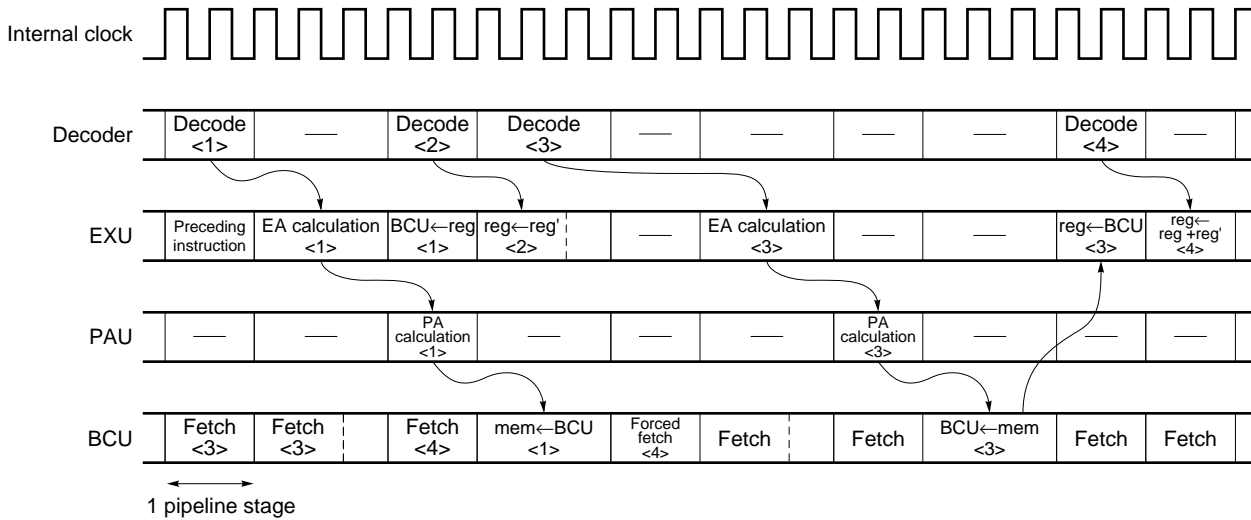
Figure 3-1. CPU Pipeline Operation Example (2/2)

(c) When internal RAM access is enabled (access to external RAM)



Remark When internal RAM access is enabled, the EXU does not go to execution of the next instruction during BCU memory write.

(d) When a forced prefetch cycle is entered



Remark A forced prefetch cycle is started when there are less than two bytes of code stored in the prefetch queue.

Forced prefetch cycle

The V25 and V35 have a six-byte prefetch queue. Normally, prefetch is executed when there is one or more bytes of free space in the queue.

When the queue contains one byte or less of opcode, execution of the next pipeline stage is halted, and forced prefetch cycles are started until at least two bytes of opcode have been stored in the prefetch queue.

If the addresses for instructions such as branch or call instructions are not consecutive, the queue contents are cleared, and two bytes of opcode at a new location are fetched for execution of the instruction.

3.2 Registers

The CPU in the V25 and V35 has a general purpose register set that is compatible with that in the V20 and V30. It also has various special function registers to control the on-chip peripheral hardware. All of the registers are mapped in memory space. In particular, the general purpose register set can also be used for on-chip RAM, and a maximum of eight banks of the register set can be arranged on the on-chip RAM.

The addresses of these registers can be relocated in 4-Kbyte units and are specified by setting the internal data area base register (IDB), which is a special function register. (See section 3.5.2.)

3.2.1 Register banks

The general purpose register set is mapped in the on-chip RAM area. This general purpose register set uses the bank format, and up to eight banks can be set, with 32 bytes used per bank. Of these eight banks, banks 0 and 1 are also used as macro service channels (see section 4.5.4) and can also be accessed as data memory (see section 3.5.4).

Normally, the CPU uses register bank 7 to execute programs. When an interrupt occurs, switching to another register bank is performed automatically. To return from the current register bank to the previous one, the return instruction RETRBI (an added instruction from the V20 and V30) from the interrupt must be executed. (See section 4.5.2.)

Figure 3-2 shows the configuration of register banks. In each register bank, (+00H) and (+01H) are reserved areas when the register bank is used. The general purpose register set is mapped in the area of (+08H) to (+1FH), which are offset from the start address of each register bank. The area from (+02H) to (+07H) is used during register bank switching, and is therefore not available for other use.

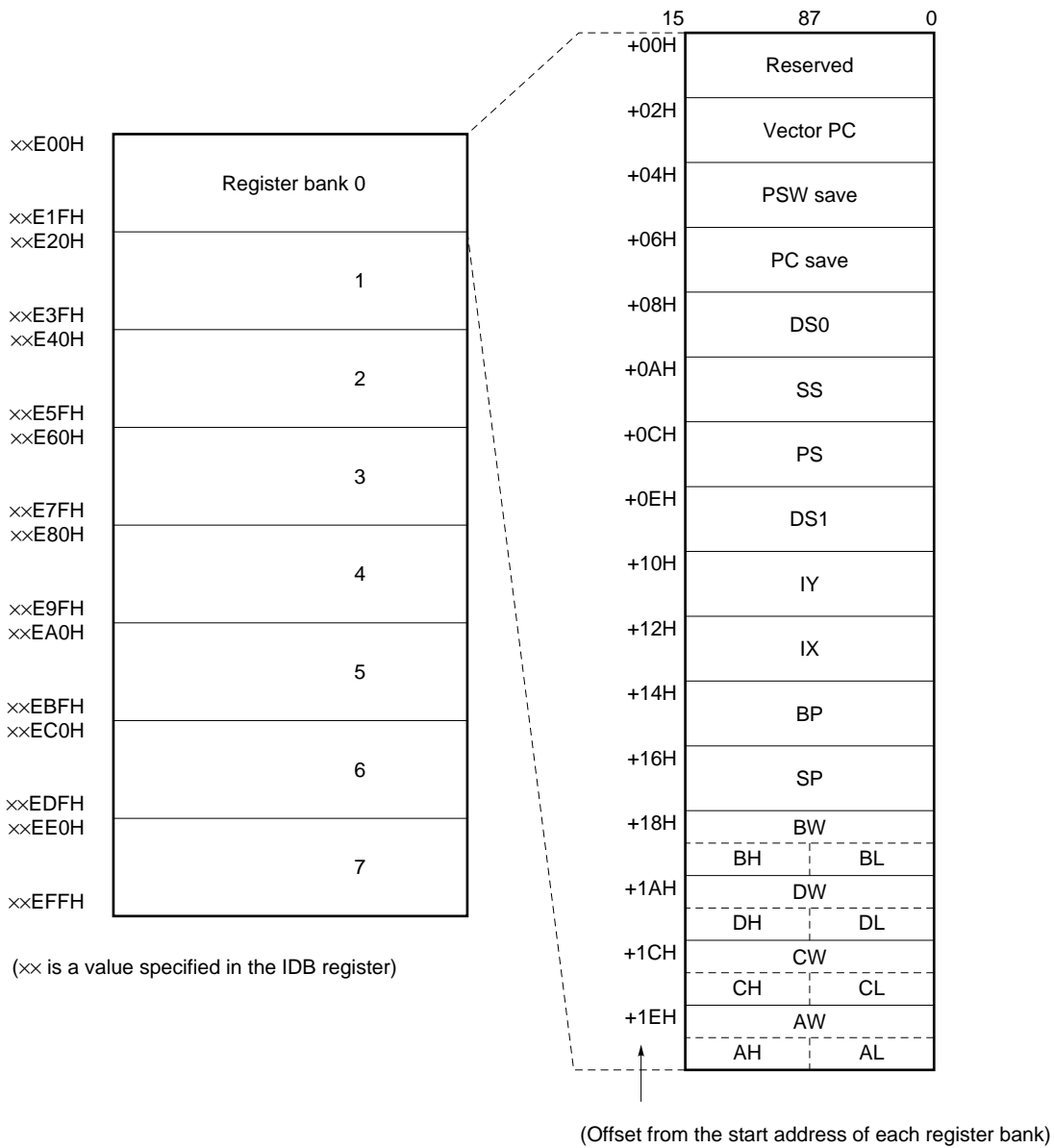
The value loaded into the program counter (PC) during register bank switching, that is, the offset value for the interrupt service routine's starting address, is set in the (+02H) area.

(+04H) is an area for saving the program status word (PSW) during register bank switching.

(+06H) is an area for saving the PC during register bank switching. (See section 4.5.2.)

After reset, register bank 7 is automatically selected. Only the segment registers (see section 3.2.4) in register bank 7 are initialized at reset.

Figure 3-2. Register Bank Configuration



3.2.2 General purpose registers (AW, BW, CW, and DW)

The general purpose registers contain four 16-bit registers. In addition to being accessed as 16-bit registers, these registers can also be accessed as 8-bit registers by dividing them into high-order and low-order halves (AH, AL, BH, BL, CH, CL, DH, DL).

The registers are used as 8-bit or 16-bit registers for various instructions, including transfer, arithmetic, and logical operation instructions.

These registers are also used as default registers for the types of processing specified below.

AW : Word multiplication/division, word input/output, and data conversion

AL : Byte multiplication/division, byte input/output, translation, BCD rotation, and data conversion

AH : Byte multiplication/division

BW : Translation

CW : Loop control branch and repeat prefix

CL : Shift instruction, rotation instruction, and BCD operation

DW : Word multiplication/division and indirect addressing input/output

These registers are mapped in the on-chip RAM. The addresses can be found using the following expression: (IDB register **Note** value \times 4096) + (0E00H) + (register bank number \times 32) + (offset for each register)

Note See section 3.5.2 regarding the IDB register.

Table 3-1. General Purpose Register Offsets

Register	Offset	Register	Offset
AW	1EH	AL	1EH
		AH	1FH
BW	18H	BL	18H
		BH	19H
CW	1CH	CL	1CH
		CH	1DH
DW	1AH	DL	1AH
		DH	1BH

3.2.3 Pointers (SP and BP) and index registers (IX and IY)

SP, BP, IX, and IY are 16-bit registers that are used as base pointers or index registers when memory is being accessed in an addressing mode such as based addressing (BP), indexed addressing (IX and IY), or based indexed addressing (BP, IX, and IY). The SP is also used as a pointer when a stack is handled. Like general purpose registers, they are used for instructions such as transfer and arithmetic operations, but they cannot be used as 8-bit registers for the instructions. These registers are also used as default registers for the types of processing specified below.

SP : Stack manipulation

IX : Block transfer and BCD string operation source

IY : Block transfer and BCD string operation destination

These registers are mapped in the on-chip RAM. The addresses can be found using the following expression.
(IDB register **Note** value \times 4096) + (0E00H) + (register bank number \times 32) + (offset for each register)

Note See section 3.5.2 regarding the IDB register.

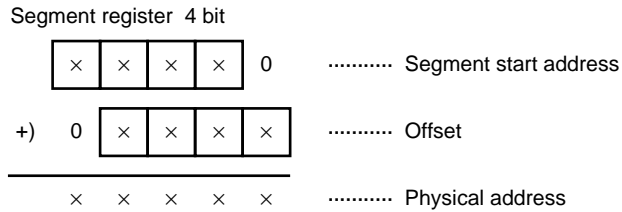
Table 3-2. Pointer and Index Register Offsets

Register	Offset
SP	16H
BP	14H
IX	12H
IY	10H

3.2.4 Segment registers (PS, SS, DS0, and DS1)

The CPU divides memory space into 64-Kbyte logical segments. The start address of each segment is specified in the segment register. Offset from the start address is specified in another register or by effective address.

Accordingly, the physical address is generated as shown below.



The segment registers are PS (Program Segment), SS (Stack Segment), DS0 (Data Segment 0), and DS1 (Data Segment 1). The segments are used for the following.

- PS : Program fetch
- SS : Stack manipulate instruction and addressing with BP as the base register
- DS0: General variables access and source block data access (block transfer instruction, etc.)
- DS1: Destination block data access (block transfer instruction, etc.)

However, a segment other than DS0 can be used if a segment override prefix is used. Similarly, another segment can be used instead of SS in the addressing mode with BP as a base register.

When reset, register bank 7's PS is initialized to FFFFH and SS, DS0, and DS1 are initialized to 0000H. These registers are mapped in the on-chip RAM. The addresses can be found using the following expression.
 (IDB register ^{Note} value × 4096) + (0E00H) + (register bank number × 32) + (offset for each register)

Note See section 3.5.2 regarding the IDB register.

Table 3-3. Segment Register Offsets

Register	Offset
DS0	08H
DS1	0EH
SS	0AH
PS	0CH

3.2.5 Internal data area base register (IDB)

The internal data area base register (IDB) is an 8-bit register used to determine the address of the internal data area (see section 3.5.1), which consists of on-chip RAM (also used for general purpose registers) and a special function register area for on-chip peripheral hardware control, etc. The register can be referenced at two addresses: FFFFFH and (IDB register value \times 4096 + FFFH). (See section 3.5.2.)

3.2.6 Special function registers

The V25 and V35 have special function registers to control on-chip peripheral hardware. These registers are mapped in the special function register area in the internal data area and are read/written using the same method as normal memory access. (See section 3.5.3).

Execution of the additional instruction BTCLR is valid only for the special function registers.

3.3 Program Counter (PC)

The program counter (PC) is a 16-bit binary counter which retains the offset of the program memory address having the program to be executed by the CPU.

The PC is incremented each time an instruction byte is fetched from the instruction queue. A new location is loaded into the PC whenever an instruction such as a branch, call, return, or break instruction is executed.

When reset, 0000H is loaded into the PC. Because the PS is initialized to FFFFH when reset, after reset the CPU starts program execution at address FFFF0H.

3.4 Program Status Word (PSW)

The program status word (PSW) consists of six status flags, five control flags, and two user flags.

- o Status flags
 - V (Overflow)
 - S (Sign)
 - Z (Zero)
 - AC (Auxiliary Carry)
 - P (Parity)
 - CY (Carry)
- o Control flags
 - RB0 to RB2 (Register Bank0 to 2)
 - DIR (Direction)
 - IE (Interrupt Enable)
 - BRK (Break)
 - $\overline{\text{IBRK}}$ (I/O Break)
- o User flags
 - F0 (user Flag0)
 - F1 (user Flag1)

The status flags are automatically set to 1 or reset to 0 according to the instruction execution result (data values). The CY flag can also be directly set, reset, or inverted by executing an instruction.

The control flags are set or reset by executing an instruction for CPU operation control. The IE and BRK flags are reset whenever interrupt servicing is started.

The user flags that are available to the user can be set, reset, or tested by executing instructions.

The PSW bit configuration is shown below.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	RB2	RB1	RB0	V	DIR	IE	BRK	S	Z	F1	AC	F0	P	$\overline{\text{IBRK}}$	CY

The PSW contents can be saved and restored from the stack by executing PUSH and POP instructions. However, when the PSW contents are restored from the stack by executing the POP PSW instruction, the contents of bits 12 to 14 (RB0 to RB2) are not restored to the PSW.

The PSW's low-order eight bits can also be saved to and restored from the AH register by executing the MOV instruction.

When an interrupt occurs, the PSW contents are automatically saved in the stack before IE and BRK are reset. When $\overline{\text{RESET}}$ is asserted, RB0 to RB2, $\overline{\text{IBRK}}$, and bit 15 are all set to 1 and other bits are reset to 0.

Caution Be sure to set PSW's bit 15 to 1.

3.4.1 CY (Carry Flag)

(1) Binary addition and subtraction

The CY (Carry Flag) is set when a byte operation results in carrying into or borrowing from bit 7. Otherwise, it is reset.

The CY is set when a word operation results in carrying into or borrowing from bit 15. Otherwise, it is reset. Increment and decrement instructions do not affect the CY flag.

(2) Logical operations

The CY flag is reset regardless of the operation result.

(3) Binary multiplication

If AH is 0 as a result of an unsigned byte operation, the CY flag is reset. If AH is not 0, the CY flag is set. If AH is an AL sign extension as a result of a signed byte operation, the CY flag is reset. Otherwise, the CY flag is set.

If DW is 0 as a result of an unsigned word operation, the CY flag is reset. If DW is not 0, the CY flag is set. If DW is an AW sign extension as a result of a signed word operation, the CY flag is reset. Otherwise, the CY flag is set.

When the product of an 8-bit immediate operation is within 16 bits, the CY flag is reset. When the product exceeds 16 bits, the CY flag is set.

(4) Binary division

Undefined

(5) Shift and rotate

When a shift or rotate instruction that also manipulates the CY flag is executed, if the bit shifted to the CY flag is 1, the CY flag is set. If this bit is 0, the CY flag is reset.

3.4.2 P (Parity Flag)

(1) Binary addition and subtraction, logical operation, and shift

When there is an even number of “1” bits among the low-order eight bits of the operation result, the P flag is set. When there is an odd number of “1” bits, the P flag is reset.

When the result is all zeros, the P flag is set.

(2) Binary multiplication and division

Undefined

3.4.3 AC (Auxiliary Flag)

(1) Binary addition and subtraction

The AC flag is set when a byte operation generates either a carry from the low-order four bits to the high-order four bits or a borrow from the high-order four bits to the low-order four bits. Otherwise, the AC flag is reset.

Word operations are performed on low-order bytes in the same way as byte operations.

(2) Logical operation, binary multiplication and division, and shift and rotate

Undefined

3.4.4 Z (Zero Flag)

(1) Binary addition and subtraction, logical operation, and shift and rotate

The Z flag is set when eight bits in the result of a byte operation or 16 bits in the result of a word operation are all zeros. Otherwise, the Z flag is reset.

(2) Binary multiplication and division

Undefined

3.4.5 S (Sign Flag)

(1) Binary addition and subtraction, logical operation, and shift and rotate

The S flag is set when bit 7 of a byte operation result is 1. When bit 7 is 0, the S flag is reset.

The S flag is set when bit 15 of a word operation result is 1. When bit 15 is 0, the S flag is reset.

(2) Binary multiplication and division

Undefined

3.4.6 V (Overflow Flag)

(1) Binary addition and subtraction

When a byte operation is performed, the V flag is set if the carry values from bit 6 and bit 7 differ. If they are the same, the V flag is reset.

When a word operation is performed, the V flag is set if the carry values from bit 14 and bit 15 differ. If they are the same, the V flag is reset.

(2) Logical operation

The V flag is reset regardless of the operation result.

(3) Binary multiplication

If AH is 0 as a result of an unsigned byte operation, the V flag is reset. If AH is not 0, the V flag is set.

If AH is an AL sign extension as a result of a signed byte operation, the V flag is reset. Otherwise, the V flag is set.

If DW is 0 as a result of an unsigned word operation, the V flag is reset. If DW is not 0, the V flag is set.

If DW is an AW sign extension as a result of a signed word operation, the V flag is reset. Otherwise, the V flag is set.

When the product of an 8-bit immediate operation is within 16 bits, the V flag is reset. When the product exceeds 16 bits, the V flag is set.

(4) Binary division

The V flag is reset regardless of the operation result.

(5) Shift and rotate

As a result of one-bit left shift or rotate instruction,

When CY = most significant bit: V is reset

When CY • most significant bit: V is set

As a result of one-bit right shift or rotate instruction,

When most significant bit = second most significant bit: V is reset

When most significant bit • second most significant bit: V is set

When shifting or rotating by multiple bits, the V flag becomes undefined.

3.4.7 $\overline{\text{IBRK}}$ (I/O Break Flag)

$\overline{\text{IBRK}}$ controls the occurrence of software interrupts when an I/O instruction is executed.

When $\overline{\text{IBRK}} = 0$, a software interrupt (interrupt vector 19) occurs whenever execution of an I/O instruction is attempted. Therefore, I/O instructions can be emulated by software.

When $\overline{\text{IBRK}} = 1$, a software interrupt does not occur even when an I/O instruction is executed.

3.4.8 BRK (Break Flag)

If the BRK flag is set, a software interrupt (interrupt vector 1) occurs each time one instruction is executed. Therefore, instructions can be traced one at a time.

The BRK flag is set by executing a memory manipulate instruction only when it is saved in the stack as part of the PSW. When the BRK flag is restored to the PSW after it is set, the flag becomes valid.

3.4.9 IE (Interrupt Enable Flag)

When an IE flag is set by executing an EI instruction, interrupts are enabled. When an IE flag is reset by executing a DI instruction, interrupts are disabled.

3.4.10 DIR (Direction Flag)

The DIR flag is set by executing the SET1 DIR instruction and is reset by executing the CLR1 DIR instruction.

If a DIR flag is set, processing is performed from a high-order address to a low-order address when a block transfer or I/O instruction is executed. If the DIR flag is reset, processing is performed from a low-order address to a high-order address.

3.4.11 RB0 to RB2 (Register Banks 0 to 2 Flag)

RB0 to RB2 indicate the current register bank among the eight register banks set in the on-chip RAM.

RB0 to RB2 are not restored from the stack when the POP PSW instruction is executed.

Caution Do not change the values for RB0 to RB2 that are saved in the stack and PSW save register by an interrupt service routine.

3.4.12 F0 and F1 (User Flags 0 and 1)

F0 and F1 are flags that the user can use as desired.

F0 and F1 can be set and reset by executing a PSW-related instruction. They can also be set, reset, and tested by setting the FLAG register (a special function register).

Figure 3-3 shows the FLAG register's bit configuration.

Figure 3-3. FLAG Register

7	6	5	4	3	2	1	0
–	–	F1	–	F0	–	–	–

3.5 Memory Space

The V25 and V35 have 1 Mbyte of memory space. Figure 3-4 shows a memory map.

00000H to 003FFH : Vector area (if not used as a vector area, this area can be used for another purpose)

××E00H to ××FFFH : Internal data area (the location of this area can be changed in 4-Kbyte units)

FFFFCH to FFFFEH : Reserved area

FFFFFFH : IDB register

Remark ××: IDB register value

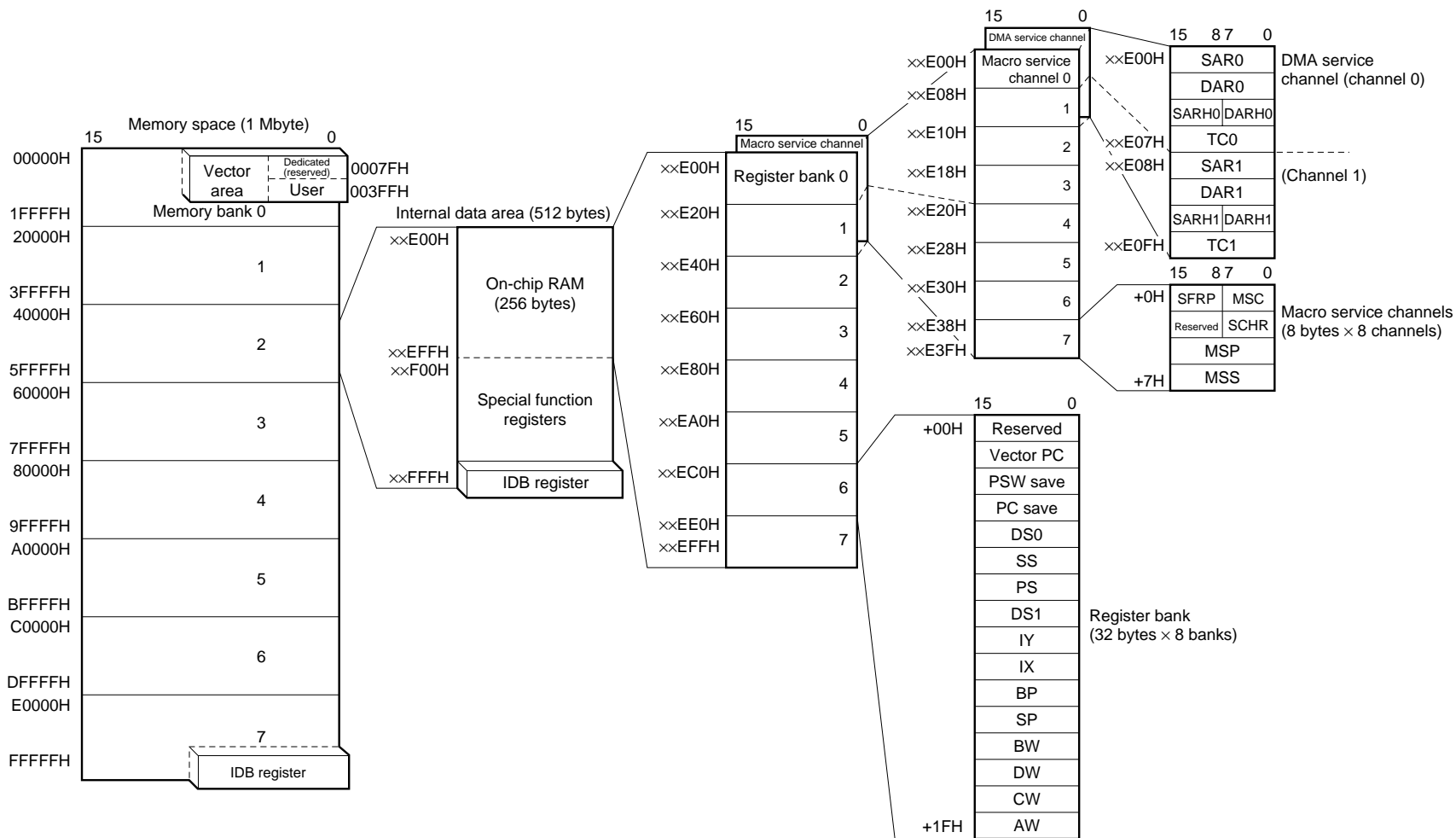
When the memory space is accessed, wait cycle insertion every 128 Kbytes is programmable.

Do not access spaces that has not been memory-mapped.

See section 3.2.4 for the physical addresses.

- Cautions**
1. If a word reference is made with the offset within the segment at address FFFFH, the second byte reference address is FFFFH + 1 rather than 0000H in the same segment (it is 0000H in the V20 and V30).
 2. If the combination of the segment value and offset value exceeds the physical address FFFFFFFH, the next address to be generated becomes 00000H.

★ Figure 3-4. Memory Map



- Remarks**
1. ×× is the IDB register value.
 2. +□□ H is the address's offset value. Adding the register bank's or macro service channel's start address to this value gives the actual address.
 3. Macro service channels are assigned in duplicate to register banks 0 and 1 and DMA service channels are assigned in duplicate to macro service channels 0 and 1.

3.5.1 Internal data area

The internal data area is a 512-byte area consisting of the on-chip RAM area and the special function register area. It can be relocated in 4-Kbyte units within the 1-Mbyte memory space. The IDB register (internal data area base register) sets the base address of the internal data area. The high-order eight bits of the 20-bit internal data area base address are specified in the IDB register and the low-order 12 bits are fixed to E00H.

When $\overline{\text{RESET}}$ is asserted, the IDB register is initialized to FFH, and therefore the internal data area is located between FFE00H and FFFFFH.

★ (1) Access to internal data area

As shown in Figure 3-4, the internal data area overlaps with the external memory area. The overlapping memory areas to be accessed are divided according to the memory access conditions. Figure 3-5 (1) and (2) shows the memory access conditions and memory area to be accessed.

Data in the internal data area is fetched in two clock cycles.

When an internal data area is accessed, control signals such as $\overline{\text{MREQ}}$, $\overline{\text{MSTB}}$, and $\overline{\text{IOSTB}}$ are not externally output.

(2) On-chip RAM area

The on-chip RAM area consists of the low-order 256 bytes of the internal data area ($\times\times\text{E00H}$ to $\times\times\text{EFFH}$, where $\times\times$ is the IDB register value). In addition to normal use as RAM, the on-chip RAM is subject to functional assignments of register banks and macro service channels (see section 3.5.4). Bit 6 (RAMEN) of the processor control register (PRC), a special function register, can be reset to 0 to disable access to the on-chip RAM as normal RAM.

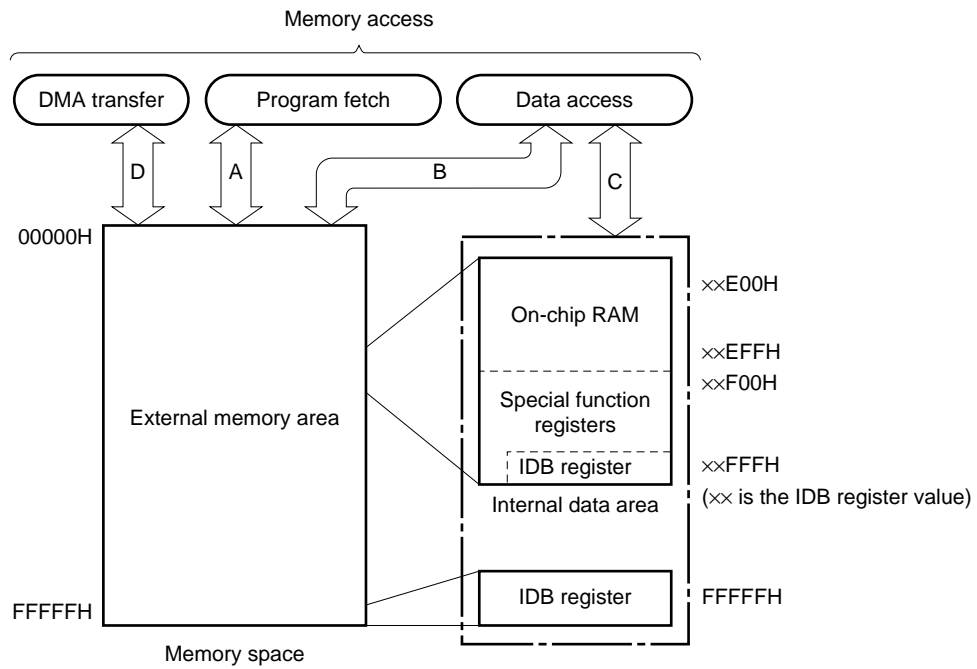
The macro service function and DMA transfer operate normally even when the RAMEN bit is reset to 0. However, at this point, neither macro service nor DMA can be reset. To set or reset macro service or DMA, set the RAMEN bit to 1.

(3) Special function register area

The high-order 256 bytes ($\times\times\text{F00H}$ to $\times\times\text{FFFH}$, where $\times\times$ is the IDB register value) of the internal data area are the special function register area. The special function registers, such as the on-chip peripheral hardware mode registers and control registers, are mapped in the special function register area (see section 3.5.3).

★

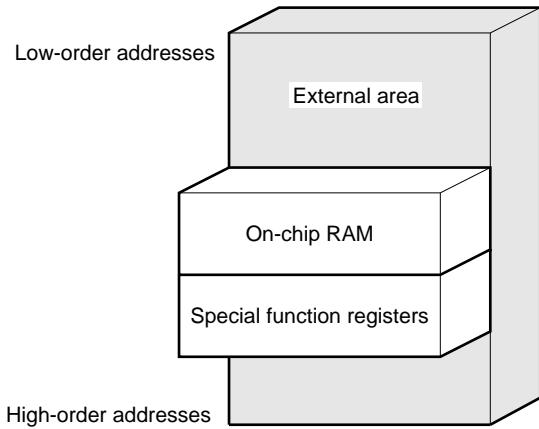
Figure 3-5. Memory Space Access Conditions



- A : For program fetch, an area other than the internal data area is accessed.
- B : Data access to an area other than the internal data area, or data access to an address corresponding to the on-chip RAM area when on-chip RAM access is disabled.
- C : If condition B is not satisfied, the internal data area takes precedence in being accessed.
- D : For DMA transfer, the external memory area is accessed.

★ (1) Program fetch

External area is accessed

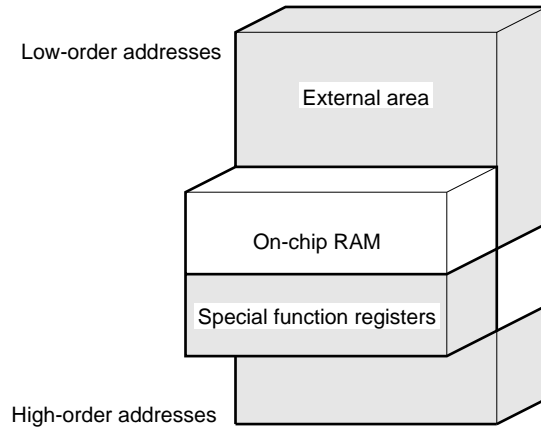


★ (2) Data access

(a) When on-chip RAM access is disabled

RAMEN = 0 (PRC)

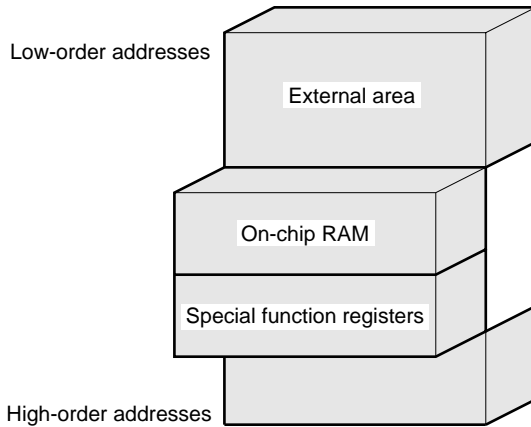
Special function registers take precedence in being accessed.



(b) When on-chip RAM access is enabled

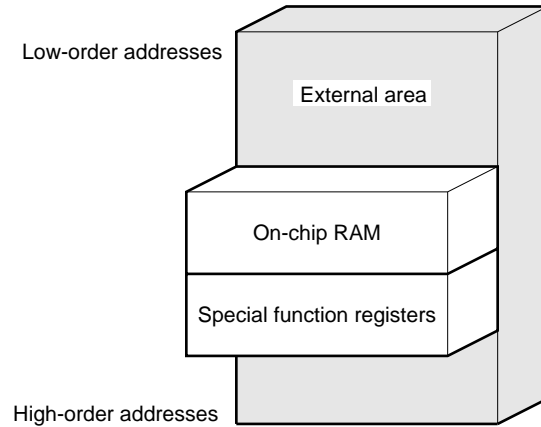
RAMEN = 1 (PRC)

Internal data area takes precedence in being accessed.



(c) DMA transfer

Only external areas are accessed.



Shaded areas are accessed

3.5.2 Internal data area base register (IDB)

The IDB register determines the physical addresses of the internal data area (on-chip RAM area and special function register area described below). The high-order eight bits of the internal data base address are specified in the IDB register. The low-order 12 bits are fixed to E00H. Access this area only as IDB.

The IDB register is allocated two addresses: $\times\times\text{FFFH}$ (where $\times\times$ is the IDB register value) in the special function register area and fixed address FFFFFH. The IDB register can be changed or referenced by accessing one of these addresses.

When reset, the IDB register is set to FFH. Consequently, the internal data area base address becomes FFE00H.

3.5.3 Special function register area

Special function registers, such as on-chip peripheral hardware mode registers and control registers, are mapped in the area $\times\times\text{F00H}$ to $\times\times\text{FFFH}$ (where $\times\times$ is the IDB register value).

Programs cannot be fetched from the special function register area.

The special function registers are manipulated by accessing memory.

Table 3-4 lists the special function registers. The columns in the table have the following meanings.

- Symbol The name of the special function register is represented by a symbol. It corresponds to the instruction operand.
- R/W Indicates whether or not the special function register can be read and/or written.
 - R/W : Both read and write are enabled.
 - R : Read only is enabled.
 - W : Write only is enabled.
- Access units Indicates the bit units in which the special function register can be manipulated (16 bits, 8 bits, or 1 bit).
- When reset Indicates the register state when $\overline{\text{RESET}}$ is asserted.

$\times\times$ in the high-order eight bits of each address is the IDB register value.

Addresses not listed in the table are reserved; a read to the reserved areas returns an undefined value, and a write to these areas is ignored.

Table 3-4. Special Function Register List (1/3)

Type	Address	Special function register name	Symbol	R/W	Access units (bits)	When reset	
Port	xxF00H	Port 0	P0	R/W	8/1	Undefined	
	xxF01H	Port 0 mode register	PM0	W	8	FFH	
	xxF02H	Port 0 mode control register	PMC0			00H	
	xxF08H	Port 1	P1	R/W	8/1	Undefined	
	xxF09H	Port 1 mode register	PM1	W	8	FFH	
	xxF0AH	Port 1 mode control register	PMC1			00H	
	xxF10H	Port 2	P2	R/W	8/1	Undefined	
	xxF11H	Port 2 mode register	PM2	W	8	FFH	
	xxF12H	Port 2 mode control register	PMC2			00H	
	xxF38H	Port T	PT	R	8	Undefined	
	xxF3BH	Port T mode register	PMT	R/W	8/1	00H	
Interrupt control	xxF40H	External interrupt mode register	INTM	R/W	8/1	00H	
	xxF44H	External interrupt macro service control register 0 Note 4	EMS0			Undefined	
	xxF45H	External interrupt macro service control register 1 Note 4	EMS1				
	xxF46H	External interrupt macro service control register 2 Note 4	EMS2				
	xxF4CH	External interrupt request control register 0 Note 4	EXIC0				47H
	xxF4DH	External interrupt request control register 1 Note 4	EXIC1				
	xxF4EH	External interrupt request control register 2 Note 4	EXIC2				
	xxFFCH	Interrupt priority register	ISPR				R
Serial interface channel 1	xxF60H	Receive buffer register 0	RxB0	R	8	Undefined	
	xxF62H	Transmit buffer register 0	TxB0	W	8	Undefined	
	xxF65H	Serial reception macro service control register 0 Note 4	SRMS0	R/W	8/1	Undefined	
	xxF66H	Serial transmission macro service control register 0 Note 4	STMS0				
	xxF68H	Serial mode register 0	SCM0				
	xxF69H	Serial control register 0	SCC0			00H	
	xxF6AH	Baud rate generator register 0	BRG0				
	xxF6BH	Serial error register 0 Note 6	SCE0	R	8	00H	
	xxF6CH	Serial error interrupt request control register 0 Note 4	SEIC0	R/W	8/1	47H	
	xxF6DH	Serial reception interrupt request control register 0 Note 4	SRIC0				
	xxF6EH	Serial transmission interrupt request control register 0 Note 4	STIC0				

Table 3-4. Special Function Register List (2/3)

Type	Address	Special function register name	Symbol	R/W	Access units (bits)	When reset
Serial interface channel 2	xxF70H	Receive buffer register 1	RxB1	R	8	Undefined
	xxF72H	Transmit buffer register 1	TxB1	W		
	xxF75H	Serial reception macro service control register 1 Note 4	SRMS1	R/W	8/1	Undefined
	xxF76H	Serial transmission macro service control register 1 Note 4	STMS1			Undefined
	xxF78H	Serial mode register 1	SCM1			00H
	xxF79H	Serial control register 1	SCC1			
	xxF7AH	Baud rate generator register 1	BRG1	R	8	00H
	xxF7BH	Serial error register 1 Note 6	SCE1			
	xxF7CH	Serial error interrupt request control register 1 Note 4	SEIC1	R/W	8/1	47H
	xxF7DH	Serial reception interrupt request control register 1 Note 4	SRIC1			
	xxF7EH	Serial transmission interrupt request control register 1 Note 4	STIC1			
Timer unit	xxF80H	Timer register 0 Note 5	TM0	R/W	16	Undefined
	xxF82H	Modulo/timer register 0 Note 5	MD0			
	xxF88H	Timer register 1 Note 5	TM1			
	xxF8AH	Modulo/timer register 1 Note 5	MD1			
	xxF90H	Timer control register 0 Note 5	TMC0	R/W	8/1	00H
	xxF91H	Timer control register 1 Note 5	TMC1	R/W	8/1	Undefined
	xxF94H	Timer unit macro service control register 0 Note 4	TMMS0			
	xxF95H	Timer unit macro service control register 1 Note 4	TMMS1			
	xxF96H	Timer unit macro service control register 2 Note 4	TMMS2			
	xxF9CH	Timer unit interrupt request control register 0 Note 4	TMIC0			
	xxF9DH	Timer unit interrupt request control register 1 Note 4	TMIC1			
xxF9EH	Timer unit interrupt request control register 2 Note 4	TMIC2				
DMA controller	xxFA0H	DMA control register 0	DMAC0	R/W	8/1	Undefined
	xxFA1H	DMA mode register 0	DMAM0			00H
	xxFA2H	DMA control register 1	DMAC1			Undefined
	xxFA3H	DMA mode register 1	DMAM1			00H
	xxFACH	DMA interrupt request control register 0 Note 4	DIC0	R/W	8/1	47H
	xxFADH	DMA interrupt request control register 1 Note 4	DIC1			

Table 3-4. Special Function Register List (3/3)

Type	Address	Special function register name	Symbol	R/W	Access units (bits)	When reset
System control	xxFE0H	Standby control register	STBC	R/W ^{Note 1}	8/1	Hold ^{Note 2}
	xxFE1H	Refresh mode register	RFM	R/W	8/1	FCH
	xxFE8H	Wait control register	WTC	R/W	16/8	FFFFH
	xxFEAH	User flag register ^{Note 3}	FLAG	R/W	8/1	00H
	xxFEBH	Processor control register	PRC	R/W	8/1	4EH
	xxFECH	Time base interrupt request control register ^{Note 4}	TBIC			47H
	xxFFFH	Internal data area base register ^{Note 4}	IDB	R/W	8/1	FFH

- Notes**
1. The standby control register can be set to 1 by executing a given instruction, but cannot be cleared to 0 (W: only "1" can be written).
 2. When power-on reset: 00H
 3. Bit manipulation of the user flag register (FLAG) has no meaning except for bits 3 and 5. The contents of FLAG register user flags 0 and 1 (F0 and F1) can also be affected by manipulating F0 and F1 in PSW (see section 3.4.12).
 4. One wait cycle is inserted in the access cycle of these registers.
 5. A maximum of six wait cycles may be inserted in the access cycle of these registers.
 6. Bit 7 indicates the RxD pin's level.

3.5.4 On-chip RAM area

The 256-byte on-chip RAM area is contained in the area from $\times\times E00H$ to $\times\times EFFH$ (where $\times\times$ is the IDB register value).

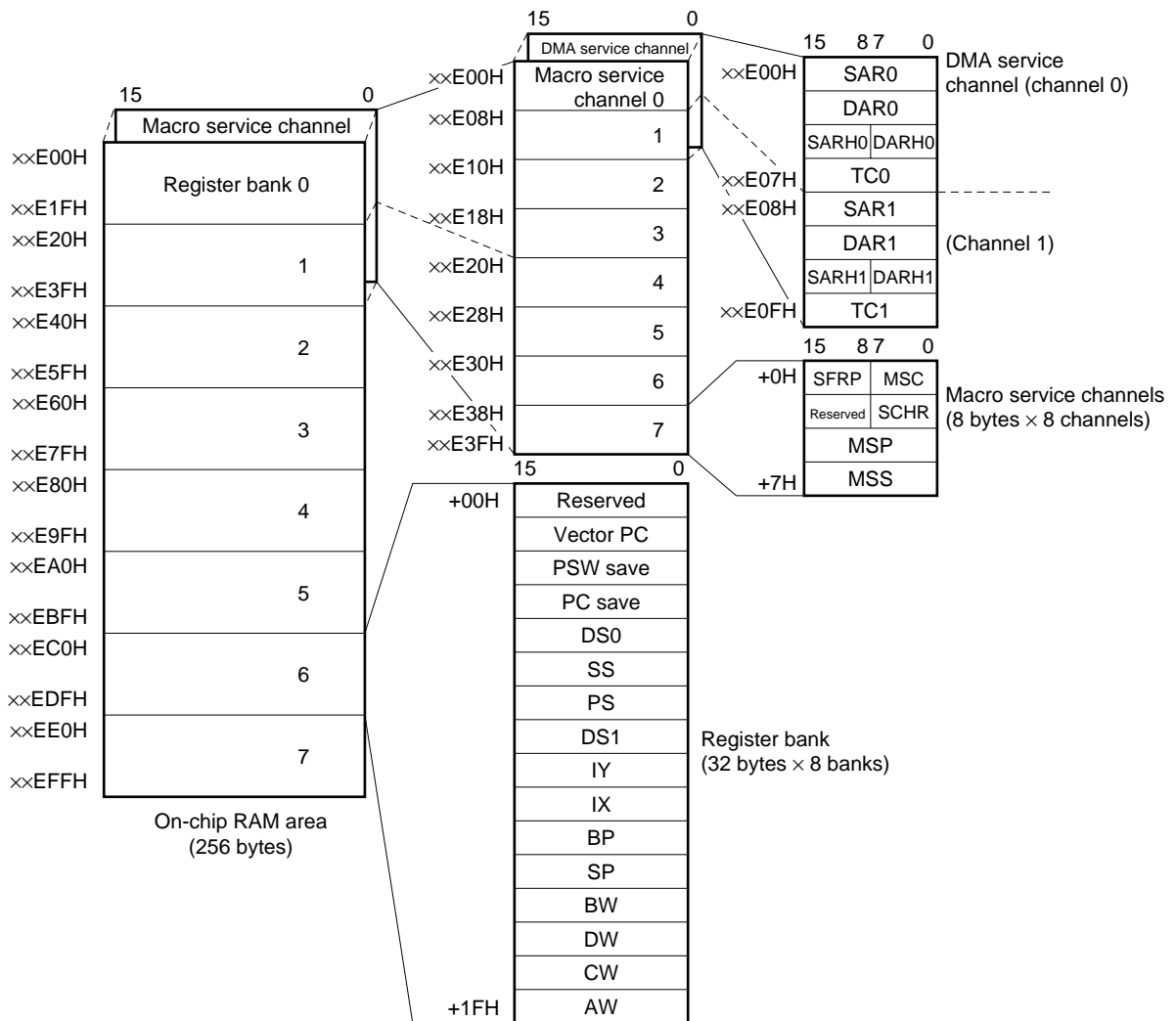
Eight register banks are mapped in the on-chip RAM. In addition, macro service channel registers are also mapped in duplicate in the on-chip RAM.

When the on-chip RAM is being accessed, the current address and data being accessed are output to the address bus and data bus as they are. The R/\overline{W} signal holds its previous state. The \overline{MREQ} and \overline{MSTB} signals become inactive.

Memory access to the on-chip RAM can be disabled by resetting bit 6 (RAMEN) of the processor control register (PRC) to 0. Programs cannot be fetched from the on-chip RAM. When a memory access is disabled, the on-chip RAM is accessed only as a register.

Access to the on-chip RAM uses two clock cycles; the same as no-wait memory access.

Figure 3-6. On-chip RAM Area Map



3.5.5 Vector table area

This area, from 00000H to 003FFH (1 Kbyte), contains 256 vectors (4 bytes per vector) for vectored interrupts. The interrupt routine start addresses corresponding to interrupt requests, break instructions, etc., are allocated to the vector table.

Vector 0 (00000H)	:	Divide error	
Vector 1 (00004H)	:	Single step	
Vector 2 (00008H)	:	NMI input	
Vector 3 (0000CH)	:	BRK 3 instruction	
Vector 4 (00010H)	:	BRKV instruction	
Vector 5 (00014H)	:	CHKIND instruction	
Vector 6 (00018H)	:	Reserved	
Vector 7 (0001CH)	:	FPO instruction	
Vector 8 (00020H)	:	Reserved	
Vector 11 (0002CH)	:	Reserved	
Vector 12 (00030H)	:	INTSER0	
Vector 13 (00034H)	:	INTSR0	
Vector 14 (00038H)	:	INTST0	
Vector 15 (0003CH)	:	Reserved	
Vector 16 (00040H)	:	INTSER1	
Vector 17 (00044H)	:	INTSR1	
Vector 18 (00048H)	:	INTST1	
Vector 19 (0004CH)	:	I/O instruction	
Vector 20 (00050H)	:	INTD0	
Vector 21 (00054H)	:	INTD1	
Vector 22 (00058H)	:	Reserved	
Vector 23 (0005CH)	:	Reserved	
Vector 24 (00060H)	:	INTP0	
Vector 25 (00064H)	:	INTP1	
Vector 26 (00068H)	:	INTP2	
Vector 27 (0006CH)	:	Reserved	
Vector 28 (00070H)	:	INTTU0	
Vector 29 (00074H)	:	INTTU1	
Vector 30 (00078H)	:	INTTU2	
Vector 31 (0007CH)	:	INTTB	
Vector 32 (00080H)	:	} User area	
			• BRK imm8 instruction
Vector 255 (003FCH)	:		• INT input

Vectors 0 to 31 are used for designated interrupt sources (part of this area is reserved) and cannot be used for other purposes.

Vectors 32 to 255 can be used for general purposes, specifically two-byte break instructions and INT input. Unused sections can be used for purposes other than vectors.

Each vector consists of four bytes. When an interrupt is acknowledged, the high-order two bytes are loaded into the program segment (PS) and the low-order two bytes are loaded into the program counter.

Example Vector 0	000H	001H	PC ← (001H, 000H)
	002H	003H	PS ← (003H, 002H)

★ 3.5.6 External memory area

External memory such as RAM is connected to the area from 00000H to FFFFEH in the μ PD70320 and 70330. However, the areas from FFF00H to FFFEFH and from FFFFCH to FFFFEH are reserved.

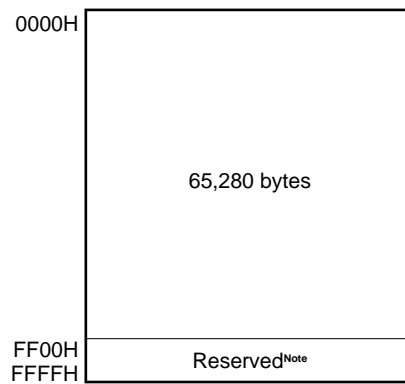
The address bus (A0 to A19), data bus (D0 to D15), and $\overline{\text{MREQ}}$, $\overline{\text{MSTB}}$, $\text{R}/\overline{\text{W}}$, and $\overline{\text{UBE}}$ signals are used to access external memory. A refresh pulse output pin ($\overline{\text{REFRQ}}$) is provided to refresh pseudo SRAM so that pseudo SRAM can be easily connected. An automatic refresh address output function is also provided to refresh DRAM so that DRAM can be easily connected (see section 5.3). In addition, wait cycles can be inserted into memory cycles in 128-Kbyte units by software (see section 5.1).

3.6 I/O Space

The V25 and V35 have a 64-Kbyte I/O space in addition to their 1-Mbyte memory space. Figure 3-7 shows an I/O space map.

Use the address bus (A0 to A15), data bus (D0 to D7 (V25) and D0 to D15 (V35)), and $\overline{\text{IOSTB}}$, $\text{R}/\overline{\text{W}}$, $\overline{\text{DMAAK0}}$, and $\overline{\text{DMAAK1}}$ signals to access the I/O space. Zeros are output from the unused high-order four bits (A16 to A19) of the address buses. Wait cycles can be inserted into I/O cycles by software (see section 5.1).

Figure 3-7. I/O Map (64 Kbytes)



Note Address, data, and control signals are also output when the reserved area is accessed.

CHAPTER 4 INTERRUPT FUNCTIONS

4.1 Interrupt Controller

The V25 and V35 contain an interrupt controller that can control multiple interrupts from 17 sources. The interrupt controller divides a total of 17 interrupt sources—five external and 12 internal sources—into groups for management, and can perform programmable multiprocessing control in group units. Also, one of three interrupt servicing functions; vectored interrupt, register bank switching, and macro service, can be selected according to the interrupt source property.

The number of external interrupt sources can be easily increased by connecting an external μ PD71059 or an external interrupt controller equivalent to the μ PD71059.

The interrupt request control registers and macro service control registers are used to set up the interrupt controller. Registers are provided for each interrupt source. The interrupt control instructions are listed below.

- DI (Disable Interrupt)
- EI (Enable Interrupt)
- RETI (Return from Interrupt)
- RETRBI (Return from Register Bank Switching Interrupt)
- FINT (Finish Interrupt), which informs the internal interrupt controller of interrupt servicing completion

An interrupt priority register is also provided to recognize the interrupt request acknowledgment state.

4.2 Interrupt Sources

There is a total of 17 interrupt sources (five external and 12 internal) for the V25 and V35. Interrupts from these 17 sources are divided into eight groups and are controlled by the interrupt controller. The configuration of these groups is fixed by hardware. Five of these eight groups (all except NMI, INT, and INTTB) can be specified priority levels from 0 to 7 (0 being the highest) by software. Functions supported by the interrupt controller vary depending upon the interrupt source. Table 4-1 lists these interrupt sources.

Table 4-1. Interrupt Source List (1/2)

Group	Interrupt source	External/ internal	Vector	Macro service	Bank switching	Priority level ^{Note}			Multiprocessing control
						Setting	Inter-group	Intra-group	
NMI	NMI (Non Maskable Interrupt)	External	2	Not available	Not available	Cannot be set	0	–	Not controlled
External INT	INT (INTerrupt)	External	External input	Not available	Not available	Cannot be set	7	–	Not controlled
Timer interrupt	INTTU0 (INTerrupt from Timer Unit0)		28	Available	Available	Can be set	1	1	Controlled
	INTTU1 (INTerrupt from Timer Unit1)		29					2	
	INTTU2 (INTerrupt from Timer Unit2)		30					3	
DMA	INTD0 (INTerrupt from DMA channel0)	Internal	20	Not available	Available	Can be set	2	1	Controlled
	INTD1 (INTerrupt from DMA channel1)		21					2	
External	INTP0 (INTerrupt from Peripheral#0)	External	24	Available	Available	Can be set	3	1	Controlled
	INTP1 (INTerrupt from Peripheral#1)		25					2	
	INTP2 (INTerrupt from Peripheral#2)		26					3	

Note The inter-group and intra-group priority levels indicate the acknowledgment sequence when interrupt requests having the same priority level assignments occur at the same time.

Table 4-1. Interrupt Source List (2/2)

Group	Interrupt source	External/ internal	Vector	Macro service	Bank switching	Priority level ^{Note}			Multiprocessing control
						Setting	Inter-group	Intra-group	
Serial interface channel 0	INTSER0 (INTerrupt from Serial ERror of channel0)	Internal	12	Not available	Available	Can be set	4	1	Controlled
	INTSR0 (INTerrupt from Serial Receiver of channel0)		13	Available				2	
	INTST0 (INTerrupt from Serial Transmitter of channel0)		14	Available				3	
Serial interface channel 1	INTSER1 (INTerrupt from Serial ERror of channel1)	Internal	16	Not available	Available	Can be set	5	1	Controlled
	INTSR1 (INTerrupt from Serial Receiver of channel1)		17	Available				2	
	INTST1 (INTerrupt from Serial Transmitter of channel1)		18	Available				3	
Time base	INTTB (INTerrupt from Time Base counter)	Internal	31	Not available	Not available	Cannot be set (fixed to 7)	6	–	Controlled

Note The inter-group and intra-group priority levels indicate the acknowledgment sequence when interrupt requests having the same priority level assignments occur at the same time.

4.3 Priority Level Control

4.3.1 Multiple interrupt priority level control

Multiple interrupt priority level control is performed in group units for all interrupts except NMI and INT.

Multiple interrupt servicing control is performed during the EI state. Therefore, set the EI using a given interrupt service routine before performing multiple servicing. However, multiprocessing control can also be performed in the DI state for interrupt responses by macro service.

(1) Multiprocessing control

Under multiprocessing control, any interrupt having a higher priority level than the interrupt being serviced is acknowledged, the interrupt being serviced is stopped, and the higher-priority interrupt is then serviced. Any interrupt that has a lower priority level than the interrupt being serviced is held pending. The pending interrupt is acknowledged when the interrupt being serviced terminates as long as the interrupt mask bit in the interrupt control register (provided for each interrupt source) has not been set and the interrupt request flag has not been reset by the current interrupt service routine. Multiple interrupts from interrupt sources having the same priority level or from within the same group cannot be serviced.

(2) Interrupt responses to all interrupts except NMI, INT, and software interrupts

For interrupt responses to these interrupts, the FINT instruction must be executed at the end of a given interrupt service routine to inform the interrupt controller of the interrupt service routine's completion. If the FINT instruction is not executed, the only subsequent interrupts to be acknowledged are those that have a priority level higher than the interrupt for which the FINT instruction was not executed (see section 4.9 **Interrupt Priority Register (ISPR)**).

Figure 4-1 shows the servicing mode for interrupts that are subject to multiprocessing control.

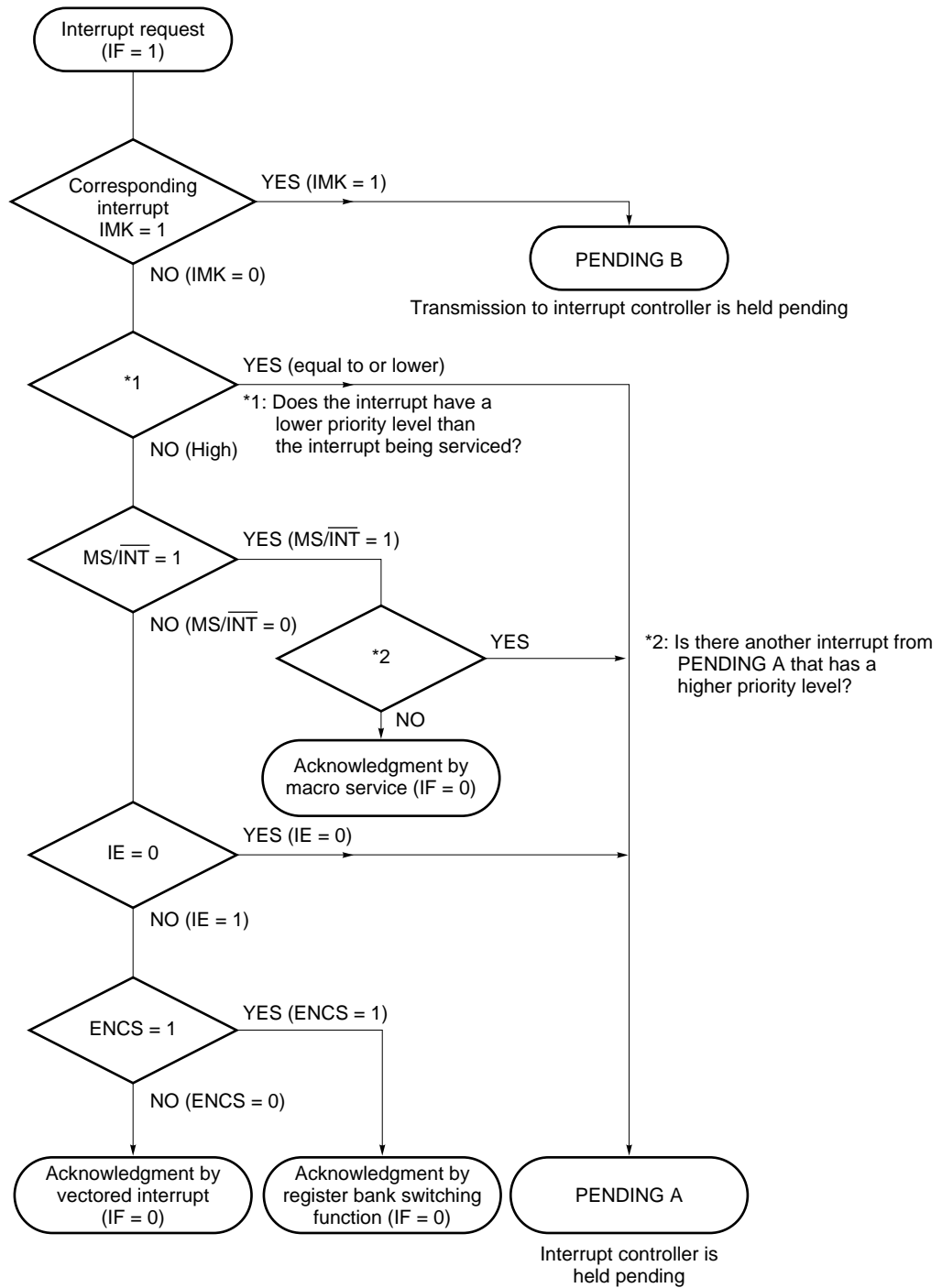
(3) Interrupt responses from NMI and INT

NMI and INT interrupt responses are not subject to multiprocessing control. INT is acknowledged when interrupts are enabled (NMI is always acknowledged).

(4) Setting of priority levels

Eight priority levels from 0 to 7 (0 being the highest) can be set as desired for each interrupt group. The priority level also indicates the number of the selected register bank when using the register bank switching function described below. The priority level is set via the three bits PR0 to PR2 in the interrupt control register that is provided for each interrupt source (the set priority level will not change as long as the settings for PR0 to PR2 remain the same). However, the priority level setting is effective only in the interrupt control register of the interrupt source that has the highest priority level in the interrupt group. This setting is ignored if it is written to any other interrupt control register. When read, the return value is fixed to 7. When a reset signal is input, all priority levels are initialized to 7. See the **Cautions on Interrupt Priority Levels and Servicing Sequence** in section A.4.

Figure 4-1. Servicing Mode of Interrupts Subject to Multiple Servicing Control



4.3.2 Priority level control for simultaneously occurring interrupts

When two or more interrupts occur simultaneously, NMI has the highest priority level for acknowledgment and INT has the lowest. The priority levels of interrupts other than NMI or INT are the same as those of multiple interrupts. Hardware-fixed priority levels are applied to groups that are specified the same priority level. Similarly, priority levels in each group are applied to the interrupts in the group.

Examples of priority level control are listed below.

- Examples**
1. If INTSR0 (specified priority level 3) and INTTU2 (specified priority level 6) occur at the same time, INTSR0 is acknowledged first.
 2. If INTP0 and INTP1 occur at the same time, INTP0 is acknowledged first.
 3. If NMI and INTD1 occur at the same time, NMI is acknowledged first.
 4. If INTTB and INT occur at the same time, INTTB is acknowledged first.
 5. If INTSER1 and INTTU1 (both specified priority level 4) occur at the same time, INTTU1 is acknowledged first.

4.4 Interrupt Requests

(1) Occurrence of interrupt requests

When an interrupt request occurs, IF (bit 7) is set to 1 in the corresponding interrupt request control register. Setting IF to 1 indicates the occurrence of an interrupt request. Interrupt requests also occur when IF is set by software.

(2) Acknowledgment of interrupt request

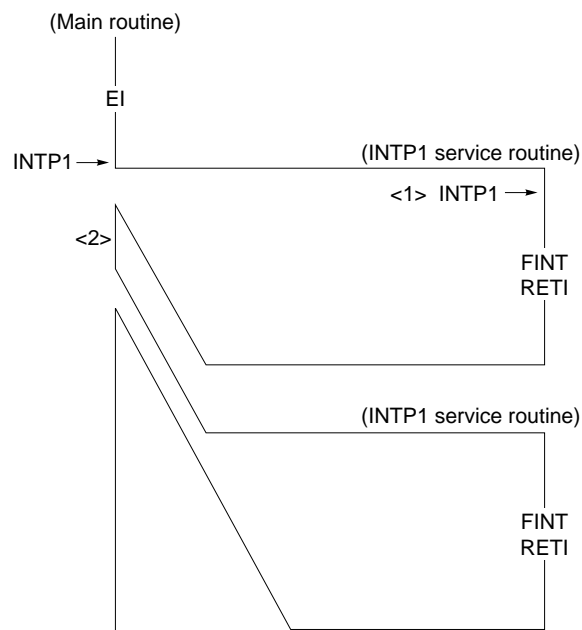
When an interrupt request is acknowledged, IF (bit 7) is reset to 0 in the corresponding interrupt request control register. Accordingly, if the same interrupt request occurs again during interrupt servicing, IF is set to 1 and the new interrupt is held pending.

(3) Holding of interrupt requests

Only one interrupt request of the same source can be held pending (see **Figure 4-2**). This means that when IF (bit 7) has been set to 1, if an interrupt request from the same source occurs, the interrupt request is ignored.

Interrupt requests that occur when IF is set to 1 are held pending. Any pending interrupt requests are canceled if the software resets IF to 0.

Figure 4-2. Multiple servicing of Same Interrupts



<1> If another INTP1 for the same interrupt occurs while INTP1 is being serviced or while a FINT instruction is being executed, the second INTP1 is held pending. However, only one INTP1 can be held pending.

<2> When servicing of the first INTP1 is completed, servicing starts for the INTP1 held pending.

(4) Polling of interrupt requests

To determine the timing of interrupt request occurrences without having an interrupt acknowledged, set the IMK bit in the interrupt control register to 1, mask the interrupt request, and poll the IF bit to detect the occurrence of interrupt requests. However, after detecting the occurrence of interrupt requests, the IF bit must be reset to 0 by software.

4.5 Interrupt Response Modes

The V25 and V35 each have three interrupt response modes; a vectored interrupt function, register bank switching function, and macro service function. Each of these functions can be selected according to the purpose of the interrupt. The interrupt controller responds to a given interrupt request according to the response mode set in the interrupt control register.

If an interrupt is acknowledged by the vectored interrupt function or register bank switching function, the contents of PC, PS, and PSW are saved using the method corresponding to the selected function. After the PSW is saved, the IE and BRK flags are reset and the DI state holds. Consequently, all interrupts except for those with NMI or macro service responses and single-step interrupts are disabled (all software interrupts except single-step interrupts can occur). (See section 4.10.)

4.5.1 Vectored interrupts

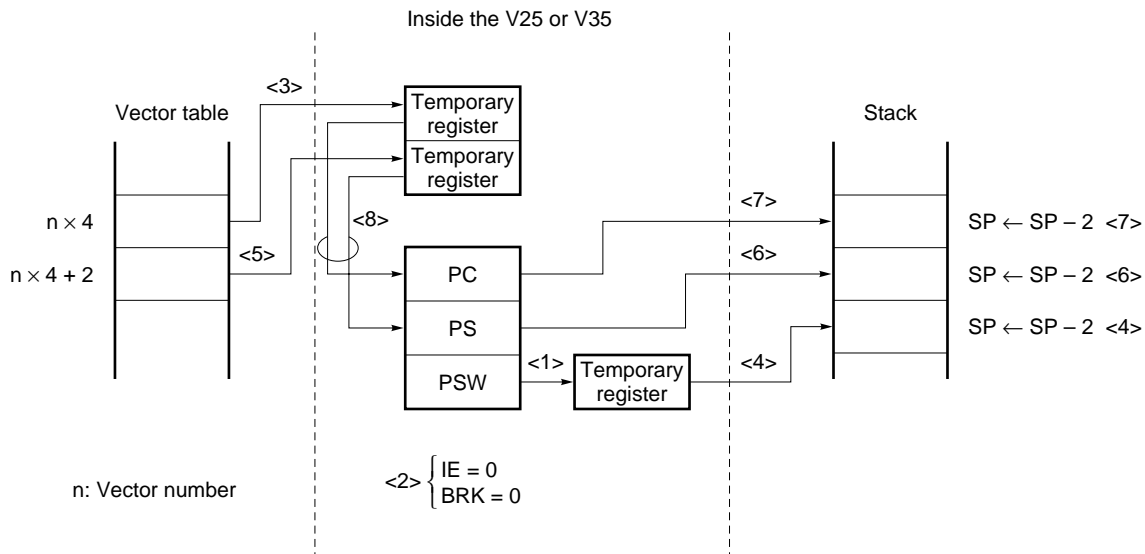
When a vectored interrupt is acknowledged, the current contents of PSW, PC, and PS are saved on a stack, then one vector is selected from a vector table and the program is executed as an interrupt service routing starting at the address indicated by the vector. All vectors other than INT interrupts are fixed. When an INT interrupt occurs, an interrupt acknowledge cycle is generated and an interrupt vector is read from the data bus (see section 4.7 INT). Table 4-1 lists interrupt vectors other than INT.

A return from an interrupt is made by executing an RETI instruction. However, an FINT instruction must be executed to make a return from any interrupt except NMI and INT. Whenever a return is made from an interrupt, PC, PS, and PSW are restored from the stack.

The vectored interrupt sequence is shown below (see also Figure 4-3).

- <1> Write the PSW value into a temporary register.
- <2> Clear the IE and BRK flags.
- <3> Read the PC value at the vector address into an internal temporary register.
- <4> $SP \leftarrow SP - 2$. Write the PSW value onto a stack when an interrupt occurs.
- <5> Read the PS value at the vector address into an internal temporary register.
- <6> $SP \leftarrow SP - 2$. Write the PS value onto a stack when an interrupt occurs.
- <7> $SP \leftarrow SP - 2$. Write the PC value onto a stack when an interrupt occurs.
- <8> Write the values read from the vector address into the PC and PS for a branch.

Figure 4-3. Interrupt Acknowledge Operation



4.5.2 Register bank switching function

The V25 and V35 map a general purpose register set in internal RAM and can have up to eight register banks. Automatic register bank switching during execution of the BRKCS or TSKSW instruction or during an interrupt response eliminates the need for software's conventional save processing of registers on a stack. This enables high-speed switching of program execution environments.

(1) Register bank switching when an interrupt request occurs

(a) Settings

To use the register bank switching function during an interrupt response, set a value of 1 for the ENCS bit in the interrupt control register provided for each interrupt source. One register bank can be specified for each interrupt group. The register bank number is the same as the multiple interrupt priority level and is specified in bits PR0 to PR2 in the interrupt control register. The register bank number matches the interrupt priority level.

PS and vector PC in the new register bank must be initialized before using the register bank switching function. Initialize SS and SP either before or after register bank switching. If the MOVSPA instruction is executed for initialization after switching, the value before switching is set in SS and SP, and the stack can be continuously used since before register bank switching. Initialize other registers as required. However, do not change PS within the interrupt service routine.

(b) Switching sequence

The register bank switching sequence is executed as shown below (see also **Figure 4-4**).

- <1> Save the contents of PSW to a temporary register.
- <2> Perform register bank switching.
- <3> Set IE to 0 and BRK to 0.
- <4> Save the PC contents and the PSW contents that were saved to a temporary register in the PC save area and PSW save area in the new register bank.
- <5> Load the interrupt service routine's start address offset into PC from the vector PC area in the register bank.

The register bank switching is now complete and the interrupt service routine is executed.

(c) Return from register bank switching

To return from a register bank switching interrupt, first execute the FINT instruction (because use of the register bank switching function is limited to interrupts subject to multiple interrupt control) and then execute the RETRBI instruction. When the RETRBI instruction is executed, PC and PSW are restored from the PC and PSW save areas in the register banks as shown in Figure 4-5. (The register banks are restored by the RETRBI instruction. Because they are not restored by the RETI instruction, they normally cannot be returned to the main routine.)

The register bank switching function can be used only for one interrupt in an interrupt group that is specified the same priority level (see section **4.8**).

When register bank switching is performed for several interrupt response modes within an interrupt group specified the same priority level, all must be switched to the same register bank.

After register bank switching, it is not possible to detect which interrupt source was used for switching. However, this can be detected in the V25+ and V35+ (see the **V25+, V35+ User's Manual**).

Figure 4-4. Register Bank Switching Sequence

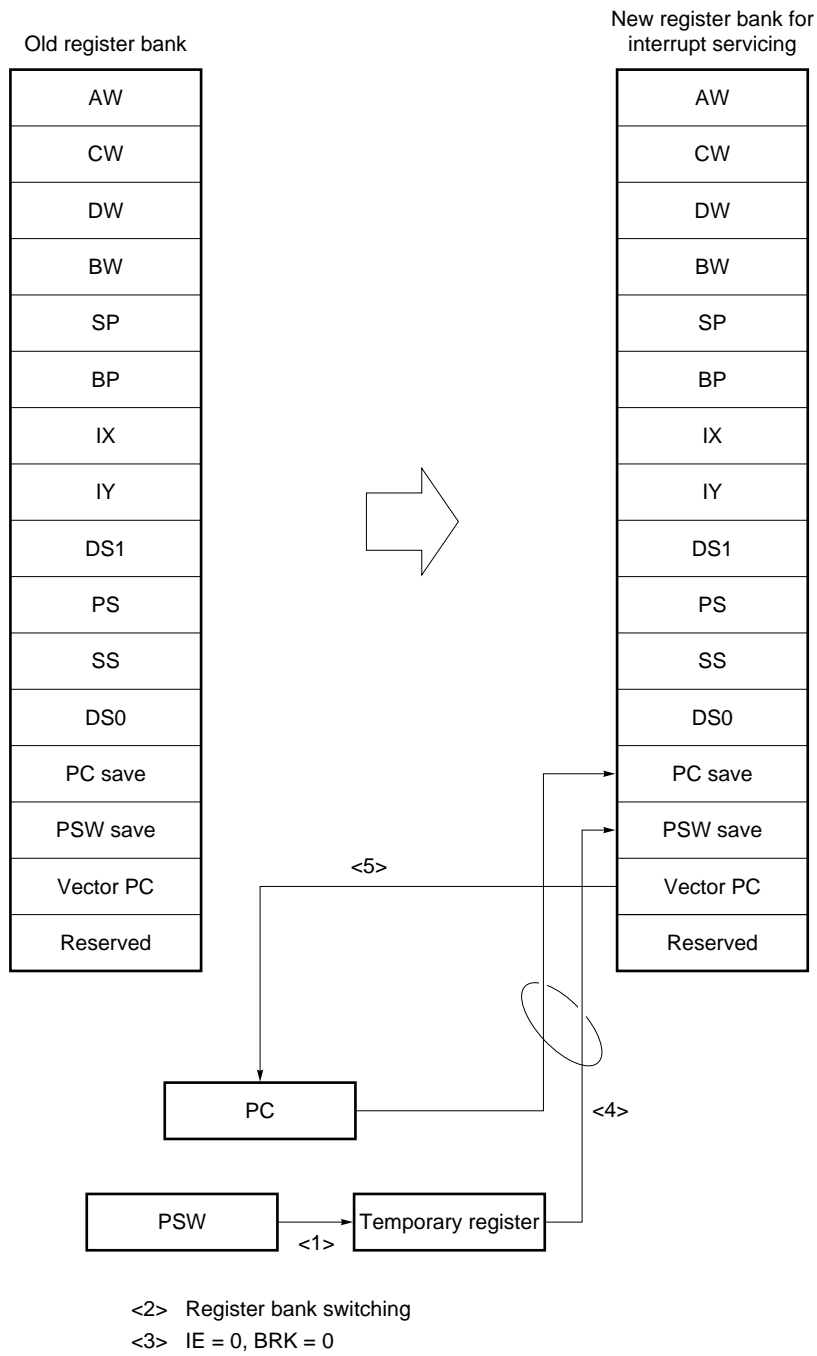
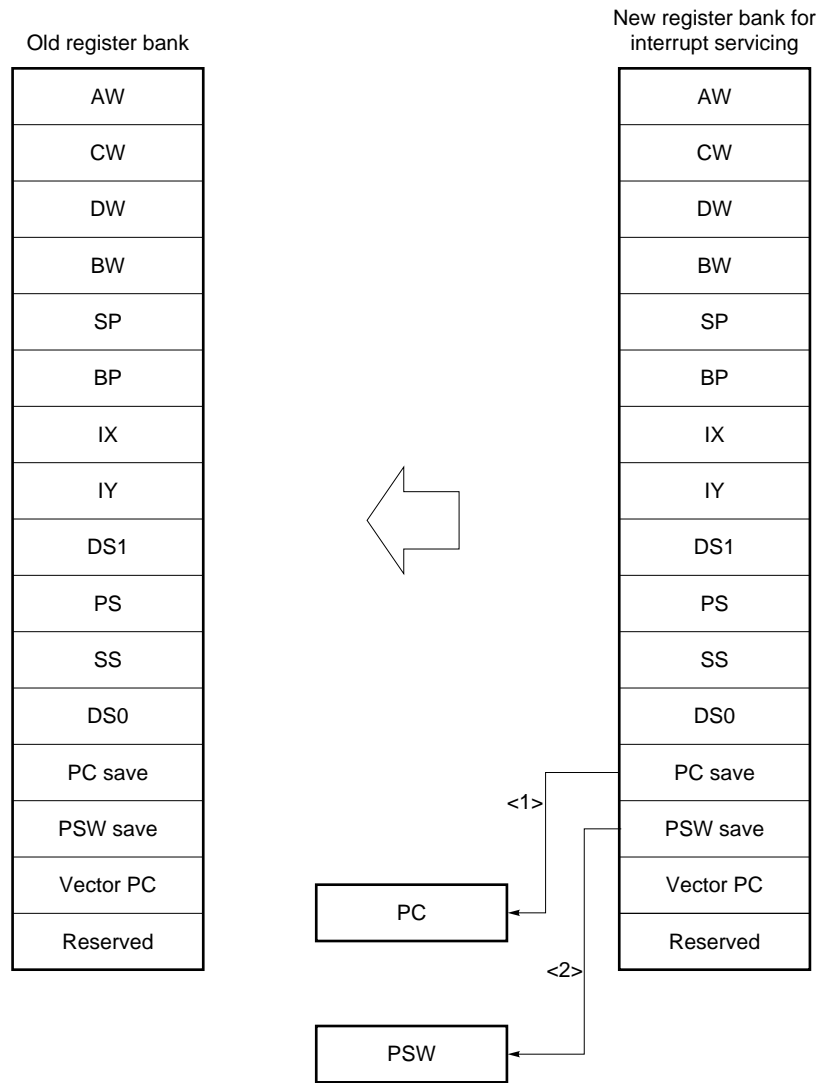


Figure 4-5. Register Bank Return Sequence



(2) Register bank switching by software (BRKCS instruction)

The BRKCS instruction can be executed to perform register bank switching. The BRKCS instruction can also be used as a high-speed subroutine call.

The number of the register bank to be switched to is specified in the low-order three bits of the 16-bit register described in the operand. Next, the PC contents and PSW contents are saved in the PC save area and PSW save area in the new register bank. Then the vector PC that was previously stored in the register bank is loaded into the PC for a branch.

(a) Settings

To execute the BRKCS instruction, the PS and vector PC in the register bank to be selected must be previously initialized. Initialize SS and SP either before or after switching. If the MOVSPA or MOVSPB instruction is executed for initialization, the value before switching is set in SS and SP in the new register bank, and the stack can be used continuously since before register bank switching.

Execute the RETRBI instruction to return from the new register bank. In this case, it is not necessary to use the FINT instruction.

(b) Switching and restore sequence

The register bank switching and restore sequence is the same as for register bank switching when an interrupt request occurs. However, the FINT instruction does not have to be executed for restoring. Restore by executing only the RETRBI instruction.

(3) Register bank switching by software (TSKSW instruction)

The TSKSW instruction can be executed to perform register bank switching. The TSKSW instruction is used for high-speed task switching.

During execution of the TSKSW instruction, the PC contents and PSW contents are first saved in the PC save area and PSW save area in the current register bank (the register bank prior to switching). Next, the number of the register bank to be switched to is specified in the low-order three bits of the 16-bit register described in the operand. Then the PC save area contents that were previously stored in the new register bank is loaded into the PC for a branch.

(a) Settings

To execute the TSKSW instruction, the PS, PC save area, SS, SP, and PSW save area in the register bank to be selected must be previously initialized. If the MOVSPB instruction is executed to initialize SS and SP, the value before switching is set in SS and SP, and the stack can be continuously used.

(b) Switching sequence

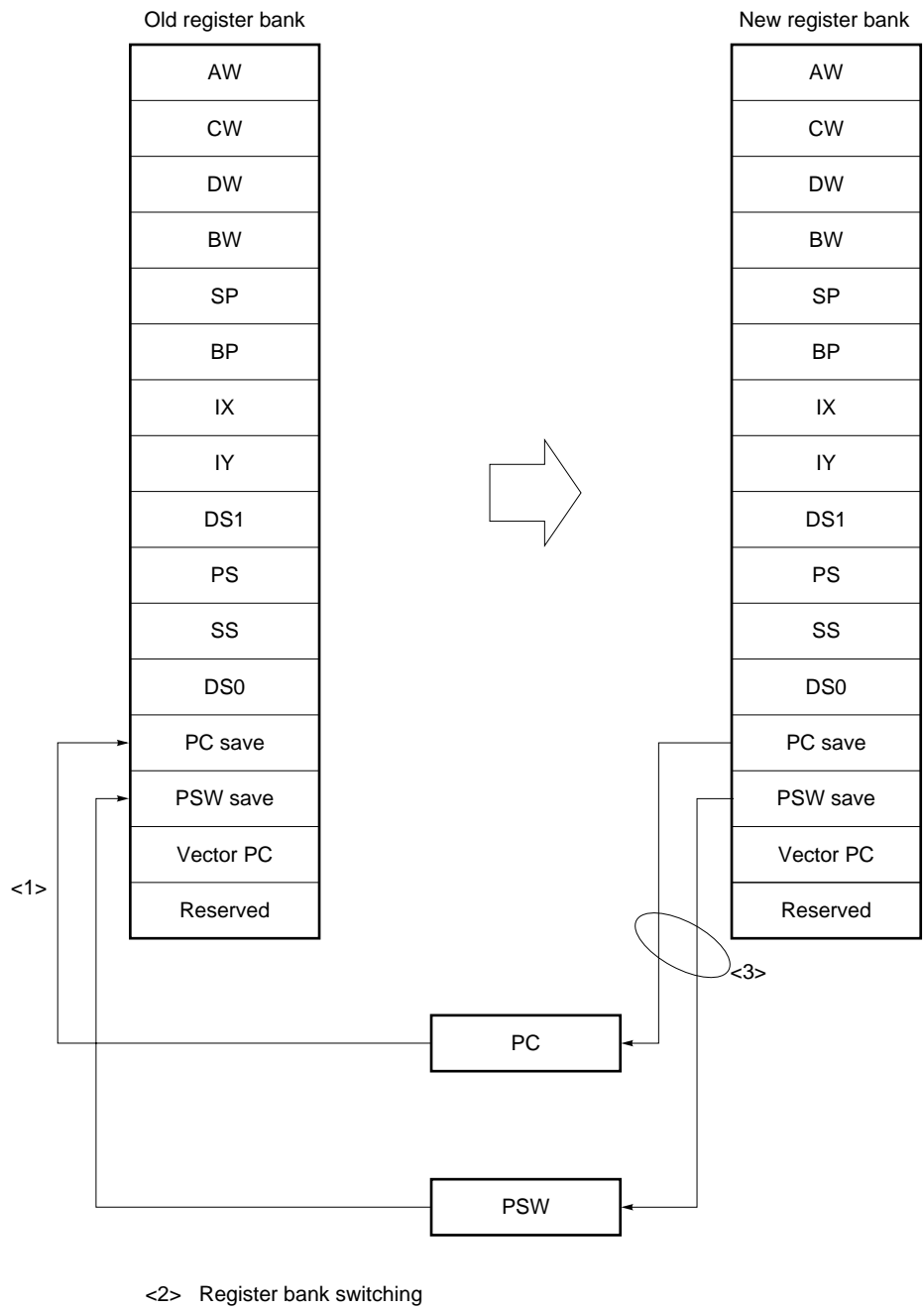
The register bank switching sequence is executed as shown below (see also **Figure 4-6**).

<1> Save PC and PSW in the PC save area and PSW save area in the current register bank (before switching).

<2> Perform register bank switching.

<3> Load the value in the PSW save area in the new register bank into the PSW and the value in the PC save area into the PC for a branch.

Figure 4-6. Register Bank Switching Sequence by Executing TSKSW Instruction



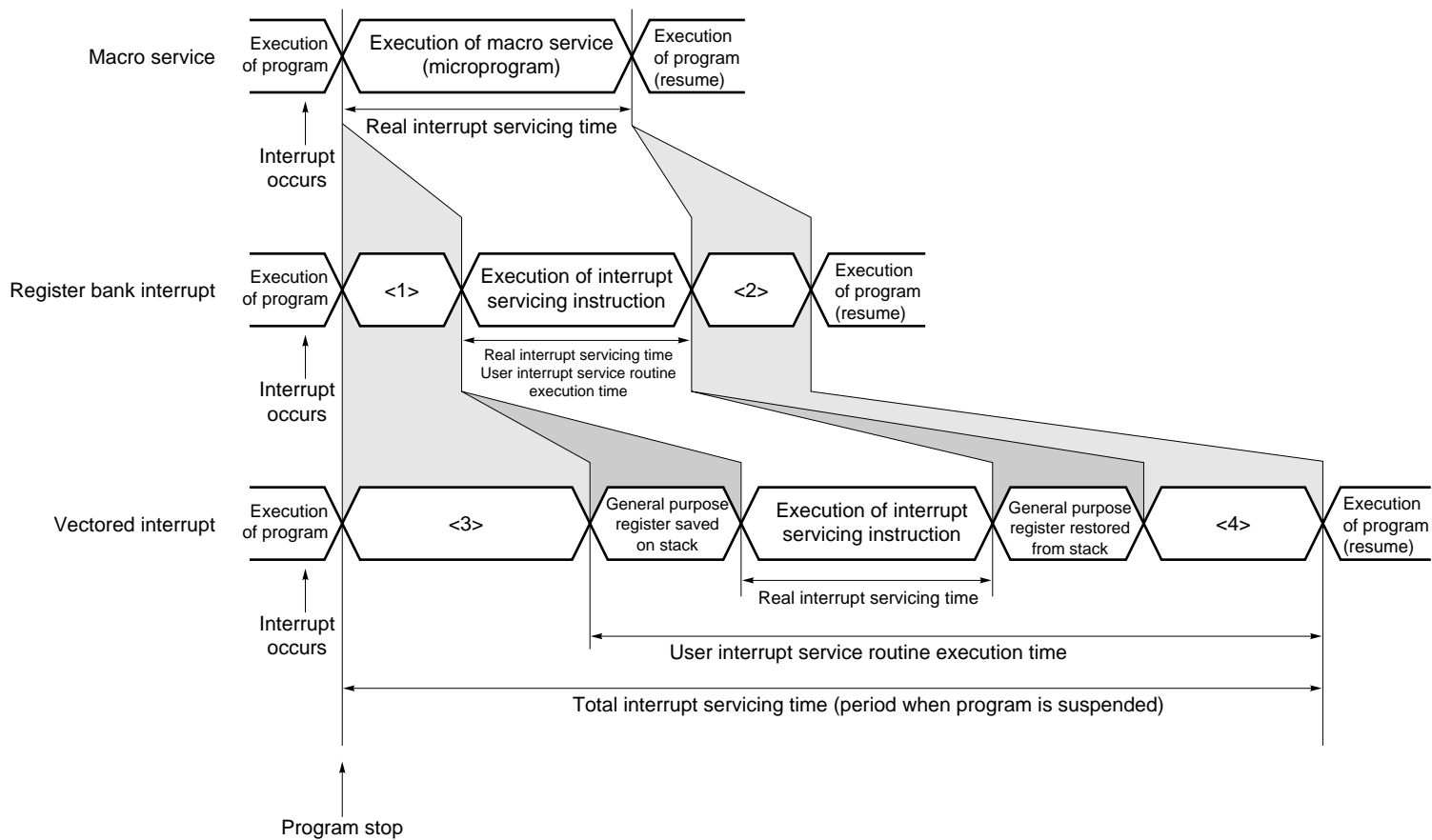
4.5.3 Macro service function

The macro service function transfers data in byte or word units between the special function register area and external memory space when an interrupt request occurs. This function enables simple processing such as simple data transfer without software interrupt servicing and can reduce interrupt servicing overhead (operations such as register saving, initialization, and restore). Processing performed by the macro service function need not be considered by software. Data that is conventionally processed by software in byte units can be processed as one unit of data, thereby enabling efficient programming.

Unlike other interrupt response modes, if the IMK bit (interrupt mask bit) of the interrupt control register provided for each interrupt source is reset and if the MS/ $\overline{\text{INT}}$ bit (macro service enable bit) is set, the macro service function serves regardless of the EI or DI state (see section 4.8). However, control based on the interrupt priority levels is effective.

When macro service has been executed as many times as specified in the macro service counter (MSC), or when the SFRP value matches the transferred data, the MS/ $\overline{\text{INT}}$ bit is reset and a macro service completion interrupt is generated. This macro service completion interrupt is held pending in the DI state.

Figure 4-7. Interrupt Servicing Efficiency Using Macro Service



<1> Save PC and PSW to a register file, then read register number from vector table for a branch.

<2> Execute RETRBI instruction. Restore PC and PSW from register file for a branch.

<3> Save PC, PS, and PSW on a stack. Read PC and PS from vector table for a branch.

<4> Execute RETI instruction. Restore PC, PS, and PSW from stack for a branch.

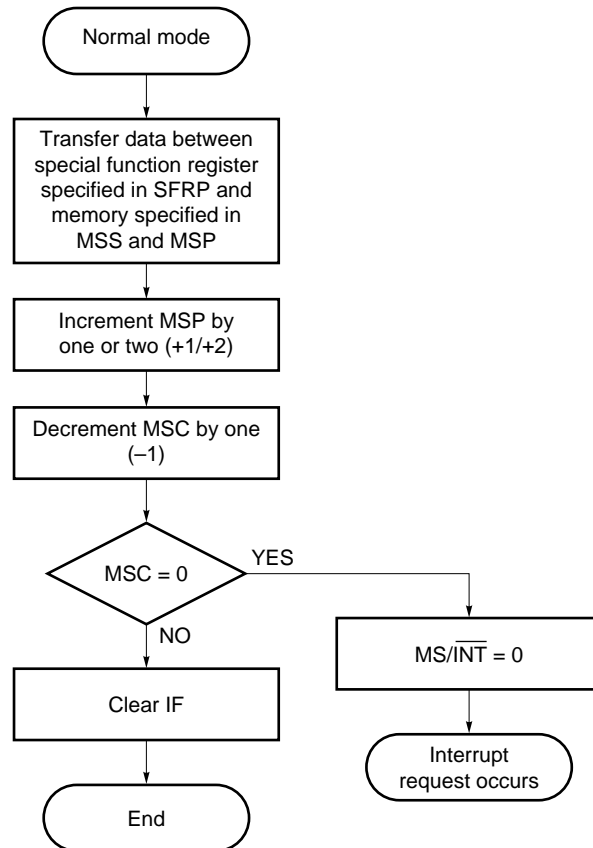
The macro service function contains the following two operation modes.

(1) Normal mode

Whenever an interrupt request occurs, one-byte or one-word data is transferred as many times as specified in the macro service counter (MSC).

Figure 4-8 shows the operation flow in normal mode.

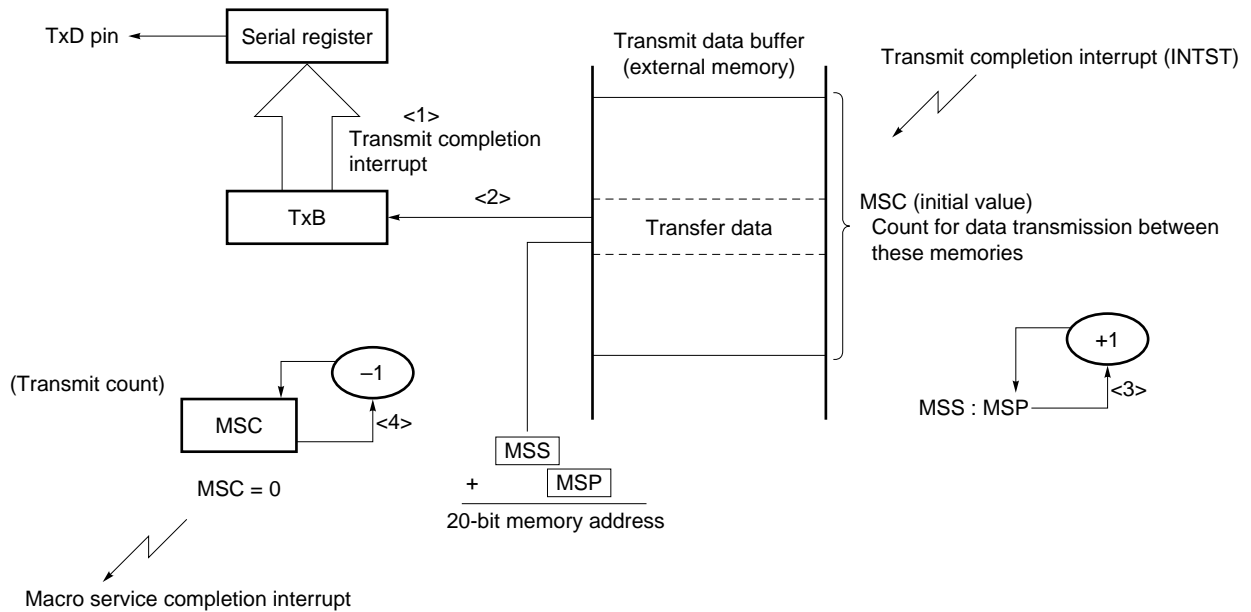
Figure 4-8. Normal Mode Operation Flow



Remark See section 4.5.4 for SFRP, MSS, MSP, and MSC.
See section 4.8 for MS/INT̄.

An example of serial interface transmission is shown below.

Figure 4-9. Example of Serial Interface Transmission



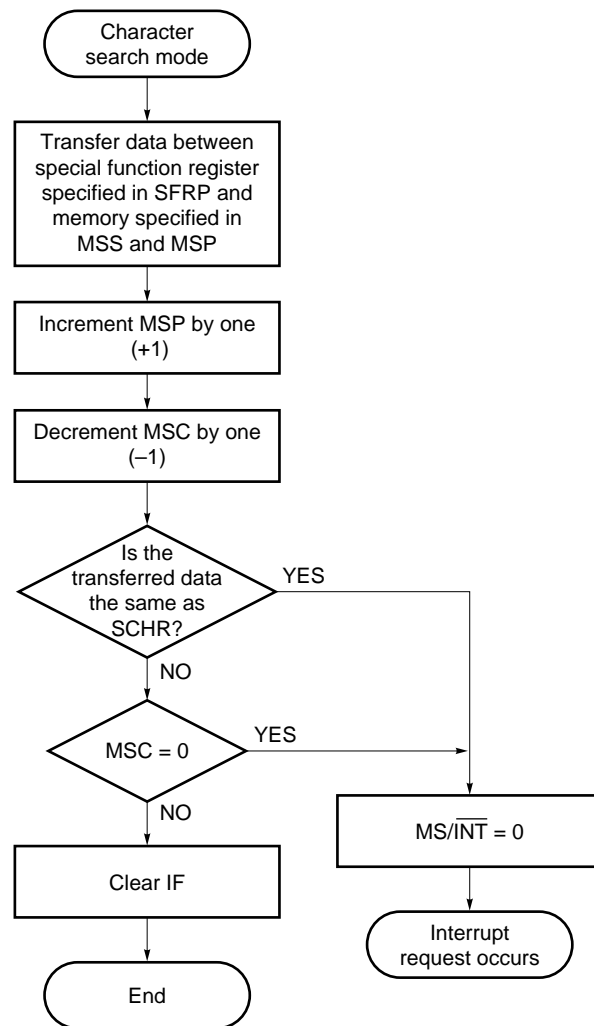
- <1> The TxB register contents are transferred to the serial register, and then a transmit completion interrupt occurs.
- <2> Data is transferred to TxB from the address indicated by "MSS:MSP".
- <3> MSP is incremented.
- <4> MSC is decremented. When MSC is 0, a macro service completion interrupt occurs.

(2) Character search mode

Whenever an interrupt request occurs, one-byte data is transferred the number of times specified in the macro service counter (MSC) or until the transferred data matches the 8-bit data previously specified as the SCHR character data.

Figure 4-10 shows the operation flow in character search mode.

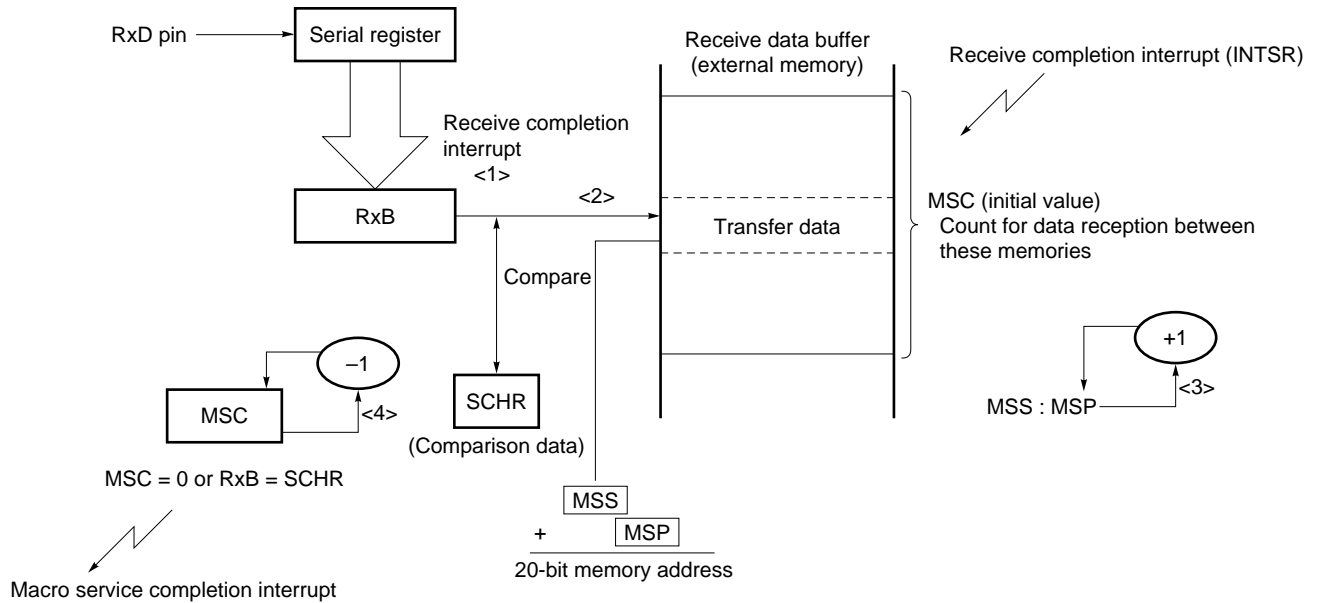
Figure 4-10. Character Search Mode Operation Flow



Remark See section 4.5.4 for SFRP, MSS, MSP, MSC, and SCHR.
See section 4.8 for MS/\overline{INT} .

An example of serial interface reception (with end code) is shown below.

Figure 4-11. Example of Serial Interface Reception



- <1> The serial register contents are transferred to the RxB register, and then a receive completion interrupt occurs.
- <2> The contents of RxB are transferred to the address indicated by "MSS:MSP".
- <3> MSP is incremented.
- <4> MSC is decremented. When MSC is 0 or when RxB = SCHR, a macro service completion interrupt occurs.

Macro service functions are controlled by the macro service control register provided for each interrupt source subject to macro servicing and by the macro service channel specified in the macro service control register.

(3) Macro service completion interrupt

When MSC is 0 or when the transfer data matches the comparison data (during character search mode only), only the MS/\overline{INT} bit is reset to 0 in the interrupt control register and the IF bit is not reset to 0. As a result, when in the EI state, an interrupt specified by the ENCS bit occurs after macro service completion. When in the DI state, this interrupt is held pending after macro service completion.

4.5.4 Macro service control register

The macro service control register is an eight-bit register that controls the macro service function. The format of this register is shown below, followed by explanations of each bit function.

7	6	5	4	3	2	1	0
MSM2	MSM1	MSM0	DIR	0	CH2	CH1	CH0

CH0 to **CH2** Macro service channel specification bits

A value from 0 to 7 can be specified.

DIR Data transfer direction specification bit

When this bit is set to 0, data is transferred from memory to the special function register. When set to 1, data is transferred from the special function register to memory.

MSM0 to **MSM2** Macro service mode specification bits

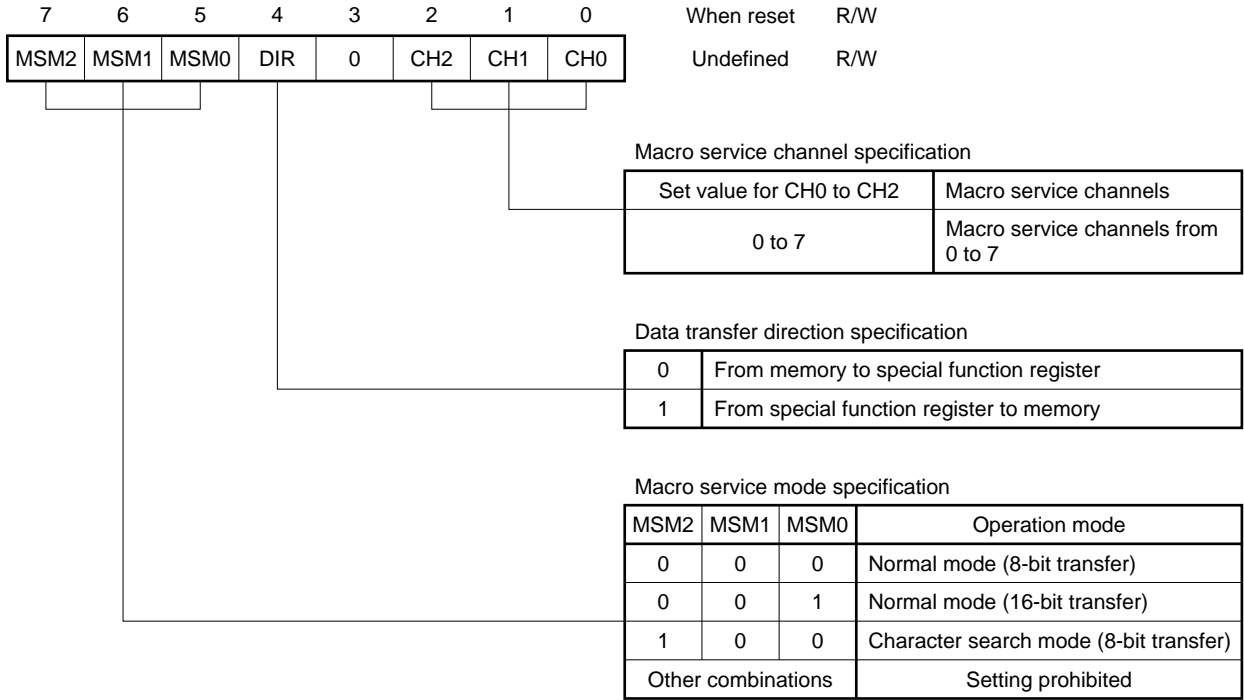
The operation mode (normal or character search) and the number of transfer data bits (8 or 16) for normal mode are specified by setting **MSM0** to **MSM2** bits in a certain combination.

MSM2	MSM1	MSM0	Operation mode
0	0	0	Normal mode (8-bit transfer)
0	0	1	Normal mode (16-bit transfer)
1	0	0	Character search mode (8-bit transfer)
Other combinations			Setting Prohibited

The macro service control register is contained in the special function register area. The register can be written/read by making an 8-bit or 16-bit memory access.

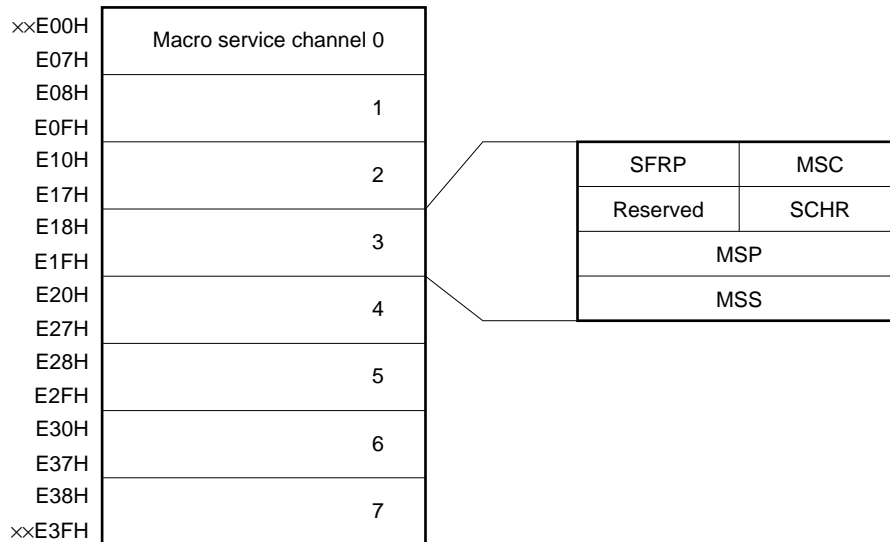
The macro service control register is provided for each interrupt source subject to macro servicing. The interrupt sources subject to macro servicing are timer interrupts (INTTU0 to INTTU2), external interrupts (INTP0 to INTP2), and serial reception and transmission interrupts (INTSR0, INTSR1, INTST0, and INTST1). See the corresponding subject headings for the locations of the macro service control registers for each interrupt source.

Figure 4-12. Macro Service Control Register



The macro service channels are specified to on-chip RAM addresses $\times\times E00H$ to $\times\times E3FH$ ($\times\times$ is the IDB register value), as shown in Figure 4-13. The data destination, data source, number of transfer bytes, and comparison character in the macro service mode are set in the macro service channel. Up to eight macro service channels can be used.

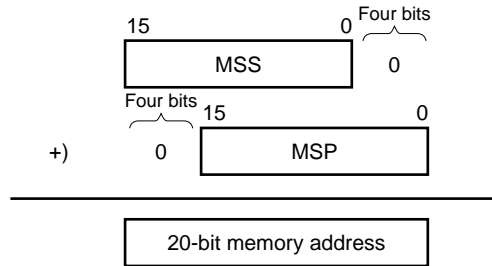
Figure 4-13. Macro Service Channel Configuration



- MSC (+0H) : Macro service transfer count
- SFRP (+1H) : Special function register address offset value. $\times\times F00H + SFRP$ ($\times\times$ is the IDB value) is the special function register address.
- SCHR (+2H) : 8-bit data for comparison in character search mode.
- MSP (+4H) : Offset value of memory address for data transfer when macro service is executed.
- MSS (+6H) : Segment value of memory address for data transfer when macro service is executed. The memory address for data transfer is $MSS \times 16 + MSP$.

The value in parentheses is the offset from the start address of the macro service channel.

The macro service memory address for data transfer is indicated by the segment specified in MSS and the offset value from the segment specified in MSP.



Whenever 8-bit or 16-bit data is transferred, the MSC in the macro service channel is decremented by one and the MSP is incremented by one or two. After this, the interrupt request flag is cleared. If 0 is written to MSC, data is transferred 256 times.

Caution Because the register banks and macro service channels are assigned to the same on-chip RAM, do not use register banks when using the macro service function.

4.6 NMI (Non-Maskable Interrupt)

NMI is the highest-priority interrupt that cannot be disabled (masked). This interrupt is edge-detected. The edge direction can be selected by setting the INTM register (a special function register) bit 0 (ESNMI bit) to 1 or 0. If the ESNMI bit is 0, an NMI interrupt occurs when the falling edge is detected and if this bit is 1, an NMI interrupt occurs when the rising edge is detected. Only a vector response can be made to the interrupt, and the vector type is fixed to 2. The NMI input is also used for the P10 pin, and the level can be tested by reading P10. When NMI is acknowledged, IE is set to 0 and interrupts other than NMI are disabled. (However, a macro service interrupt response is acknowledged.)

NMI requests are acknowledged during NMI servicing.

4.7 INT (Interrupt)

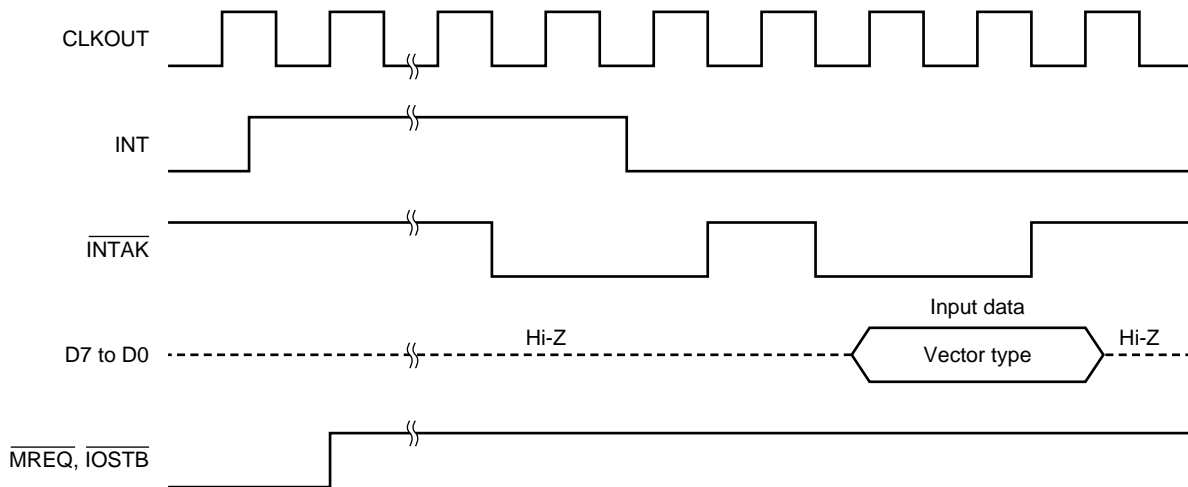
INT is a maskable interrupt. This interrupt is detected by its level (active high). INT is not subject to multiple servicing control by the interrupt controller; and is always acknowledged if interrupts are enabled (IE = 1). However, when a number of interrupts occur at once, the lowest priority level is assigned. Only a vector response can be made to INT, and the vector type is fetched from the data bus during an interrupt acknowledge cycle. The interrupt acknowledge cycle can be confirmed by $\overline{\text{INTAK}}$ output. The INT pin is also used for P14 and $\overline{\text{POLL}}$; it is selected by setting bit 4 of the port 1 mode control register (PMC1), a special function register. Thus, when the INT function is not selected, an INT interrupt does not occur even if interrupts are enabled (IE = 1). $\overline{\text{INTAK}}$ is also used for P13 and $\overline{\text{INTP2}}$, and the function is selected by setting bit 3 of PCM1 (if the $\overline{\text{INTAK}}$ function is not selected, external notification of interrupt acknowledge cycle generation is not possible). Figure 4-14 shows the timing of the interrupt acknowledge cycle.

Hold the INT signal high at least until the first $\overline{\text{INTAK}}$ signal is output.

External interrupt inputs can be expanded to a maximum of 64 by connecting an external $\mu\text{PD71059}$ or an interrupt controller equivalent to the $\mu\text{PD71059}$.

When an INT interrupt is acknowledged, interrupts are disabled (IE = 0).

Figure 4-14. INT Interrupt Acknowledge Timing



4.8 Interrupt Request Control Register

The interrupt request control register is an 8-bit register that controls interrupts other than INT and NMI. The interrupt request control register format is shown below, followed by a description of each bit function.

7	6	5	4	3	2	1	0
IF	IMK	MS/INT	ENCS	0	PR2	PR1	PR0

PR0 to **PR2** : Interrupt group priority level specification bits

One level from 0 to 7 can be specified. The priority level can be specified only in the interrupt request control register for the interrupt having the highest priority level in the group; if the priority level is specified in any other interrupt request control register, it becomes insignificant (7 is always read). The priority levels in other interrupt request control registers conform to the priority level in the interrupt request control register for the interrupt having the highest priority level in the group.

The priority level also specifies the new register bank when the register bank switching function is executed.

ENCS : Bit specifying whether or not the register bank switching function is used

When this bit is set to 1, the register bank switching function is used; when set to 0, the vectored interrupt function is used.

MS/INT : Interrupt response mode selection bit

When this bit is set to 1, the macro service function is used; when set to 0, the vectored interrupt or register bank switching function is used.

IMK : Interrupt mask bit

When this bit is set to 1, the corresponding interrupt is masked; when set to 0, it is not masked.

IF : Bit indicating the corresponding interrupt request

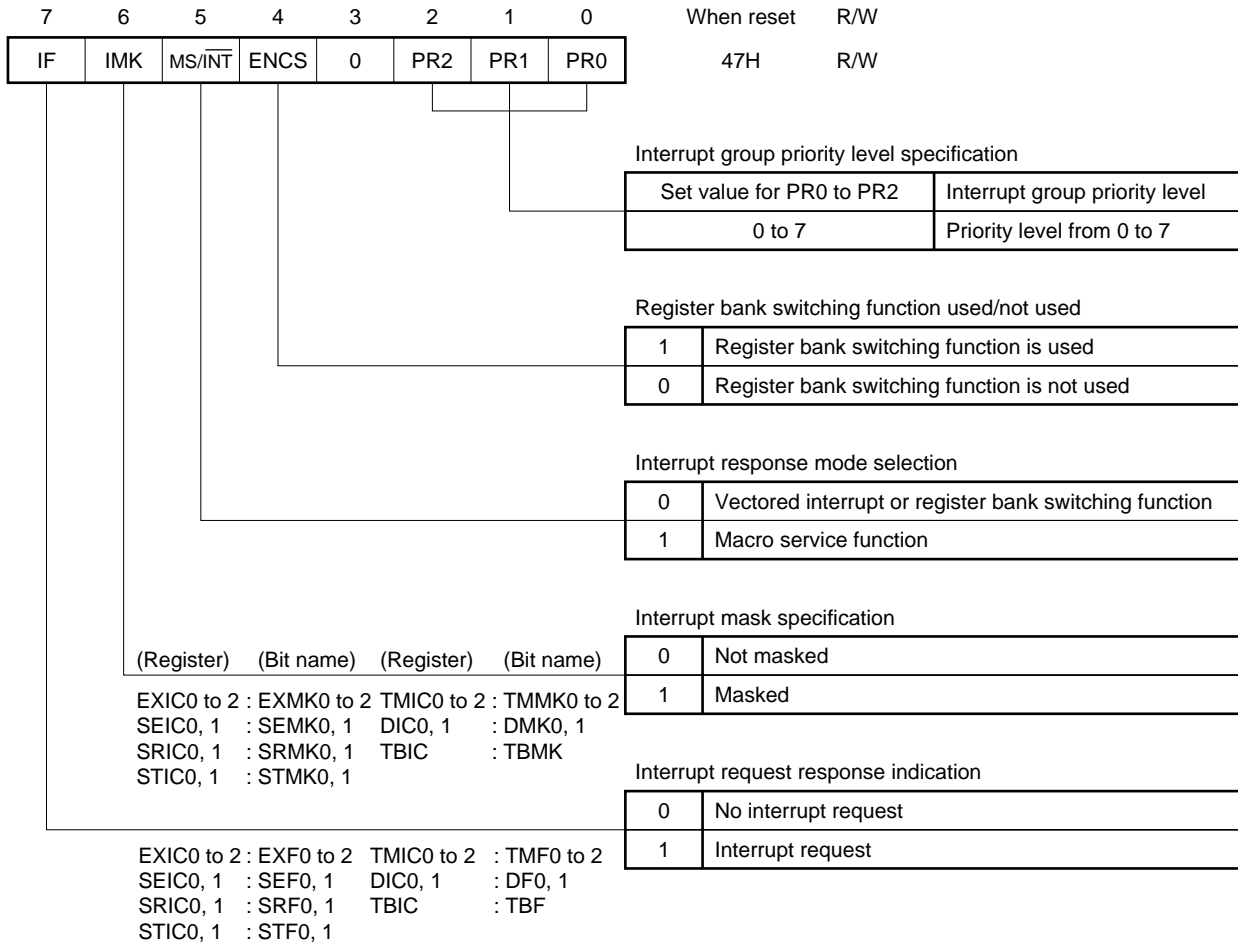
When this bit is set to 1, it indicates that the corresponding interrupt request exists; when set to 0, it indicates that the corresponding interrupt request does not exist. When the corresponding interrupt request occurs, the bit is set to 1. When the interrupt is acknowledged or an instruction such as BTCLR (an additional instruction from the V20/V30) is executed, the bit is reset to 0. This bit is set to 1 whenever an interrupt request occurs, even when interrupts have been masked via the IMK bit.

Caution If the IF bit is set to 0 by a program during interrupt servicing, interrupts will no longer occur.

The interrupt request control register is contained in the special function register area. This register can be written/read by making an 8-bit or 1-bit memory access.

The interrupt request control register is provided for each interrupt source except INT or NMI. See the corresponding subject headings for the locations of the interrupt request control registers for each interrupt source.

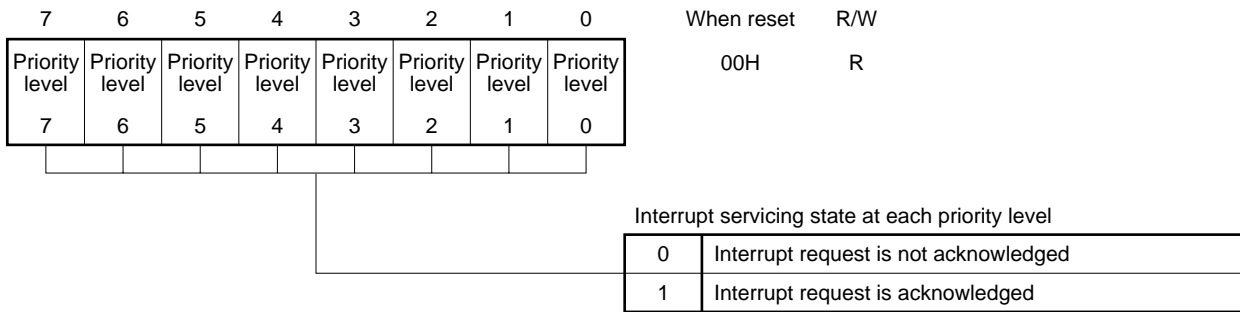
Figure 4-15. Interrupt Request Control Register



4.9 Interrupt Priority Register (ISPR)

The interrupt priority register (ISPR) is an 8-bit register that indicates the multiple interrupt servicing state under the interrupt controller's priority level control. The ISPR register cannot be written to.

Figure 4-16. ISPR



Bits 0 to 7 correspond to priority levels 0 to 7. When interrupt request having one priority level is acknowledged, the bit corresponding to the priority level is set to 1.

The least significant bit (corresponding to the highest priority level) among the bits set to 1 is reset to 0 by executing one FINT instruction.

When one of the bits 0 to 7 is set to 1, any interrupt request having a priority level lower than the priority level corresponding to the bit is not acknowledged and is held pending.

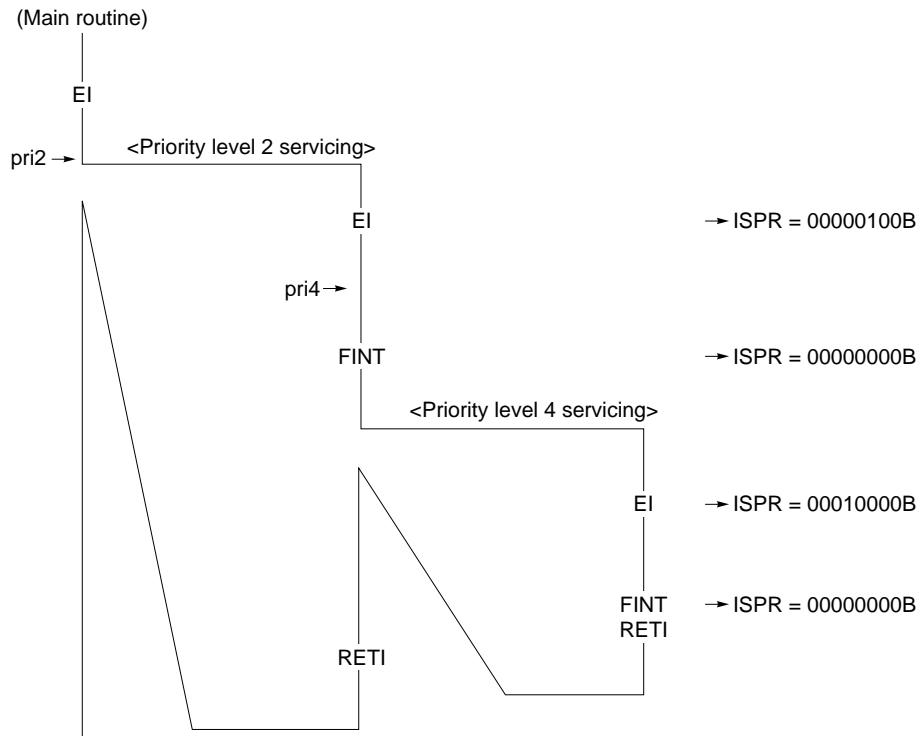
Any interrupt request having a priority level higher than the priority level corresponding to this bit is acknowledged, servicing of the current interrupt is stopped, and the current interrupt is held pending.

The ISPR register is contained in the special function register area and can be read by making an 8-bit memory access.

When $\overline{\text{RESET}}$ is asserted, the ISPR contents are initialized to 00H.

- Cautions**
1. Unless a FINT instruction is executed, the corresponding bit will not be reset to 0 when the servicing ends and, consequently, interrupts with lower priority levels cannot be acknowledged.
 2. If a FINT instruction is used without being immediately followed by a RETI or RETRBI instruction, priority level control cannot be used for the V25 or V35. In such cases, control must come from the application.

Figure 4-17. ISPR States



4.10 External Interrupts

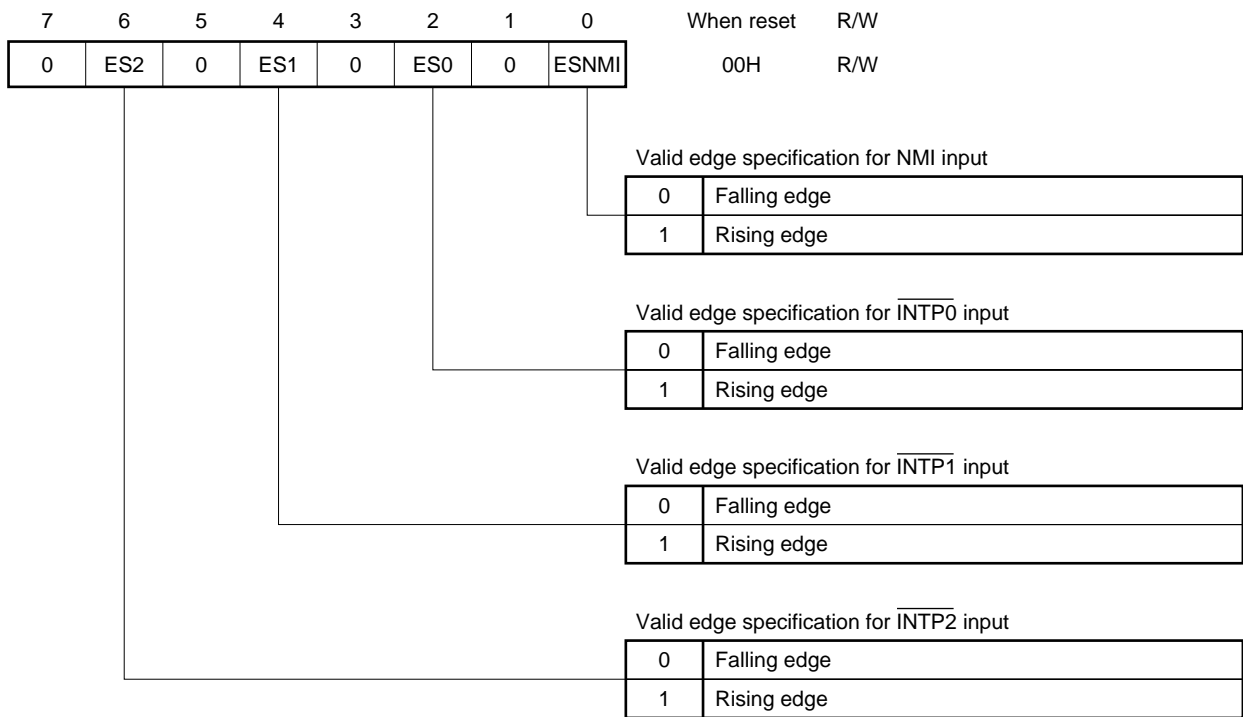
There are five sources of external interrupt requests. INT is level-detected and external interrupts other than INT (NMI and INTP0 to INTP2) are edge-detected. For each of the external interrupts other than INT that are edge-detected, the valid edge can be specified in the external interrupt mode register (INTM), a special function register.

4.10.1 External interrupt mode register (INTM)

The external interrupt mode register (INTM) is an 8-bit register that specifies the valid edge for external interrupt requests. The edge-detected interrupts are NMI and INTP0 to INTP2. The valid edges for these interrupts are specified in the INTM register.

Figure 4-18 shows the INTM register format and bit functions.

Figure 4-18. INTM



The INTM register is contained in the special function register area. It can be written/read by making an 8-bit or 1-bit memory access.

When $\overline{\text{RESET}}$ is asserted, the INTM register contents are initialized to 00H.

4.10.2 External interrupt request control registers (EXIC0 to EXIC2)

The EXICn registers (n = 0 to 2) are 8-bit registers that control interrupt requests (EXF0 to EXF2) occurring from the three external interrupt request pins ($\overline{\text{INTP0}}$ to $\overline{\text{INTP2}}$).

These three interrupt requests make up one group as external interrupt requests, and the group's priority level is programmable. Within the group, the priority levels are fixed as follows.

EXF0 > EXF1 > EXF2

Figure 4-19. EXIC0, EXIC1, and EXIC2

	7	6	5	4	3	2	1	0
EXIC0	EXF0	EXMK0	MS/ $\overline{\text{INT}}$	ENCS	0	PR2	PR1	PR0
EXIC1	EXF1	EXMK1	MS/ $\overline{\text{INT}}$	ENCS	0	1	1	1
EXIC2	EXF2	EXMK2	MS/ $\overline{\text{INT}}$	ENCS	0	1	1	1

Caution Bits 2 to 0 of the EXIC1 and EXIC2 registers are fixed to 1.

The interrupt request priority levels in the EXIC1 and EXIC2 registers conform to the settings for bits PR2 to PR0 in the EXIC0 register.

See section 4.8 **Interrupt Request Control Register** for the description of EXICn register's bits.

The EXICn register can be read/written by executing an 8-bit or 1-bit memory access. In this case, one wait cycle is automatically inserted.

When $\overline{\text{RESET}}$ is asserted, the register contents are reset to 47H.

4.10.3 External interrupt macro service control registers (EMS0 to EMS2)

The EMSn registers (n = 0 to 2) are 8-bit registers that control macro services started when any of three interrupt requests occur from external interrupt request pins ($\overline{\text{INTP0}}$ to $\overline{\text{INTP2}}$).

The EMS0 register controls the macro service started via the EXF0 flag.

The EMS1 register controls the macro service started via the EXF1 flag and the EMS2 register controls the macro service started via the EXF2 flag.

The EMSn register can be read/written by executing an 8-bit or 1-bit memory access. In this case, one wait cycle is inserted.

Figure 4-20. EMS0, EMS1, and EMS2

7	6	5	4	3	2	1	0
MSM2	MSM1	MSM0	DIR	0	CH2	CH1	CH0

See section 4.5.4 **Macro service control register** for description of the EMSn register's bits.

4.11 Software Interrupts

The V25 and V35 each contain nine types of software interrupts (see **Table 4-2**). Six of the software interrupts are compatible with the V20 and V30 (however, emulation mode interrupts are not available). The other three interrupts are unique to the V25 and V35.

The vectors of the interrupts are predefined.

Any interrupt other than the BRK interrupt (single step interrupt) is always acknowledged if the interrupt occurrence condition becomes true (it takes precedence over hardware interrupts). The BRK flag interrupt occurs when BRK = 1 regardless of the hardware or software. When the interrupt is acknowledged, the BRK is automatically reset, and therefore the interrupt priority level is lower than other hardware and software interrupts, and another BRK flag interrupt does not occur during the interrupt servicing.

Table 4-2. Software Interrupts

Interrupt source	Vector	Priority level
DIVU divide error	0	1
DIV divide error		
CHKIND boundary over	5	
BRKV	4	
BRK 3	3	
BRK imm8	32 to 255	
BRK flag (single step)	1	2
I/O instruction ($\overline{\text{IBRK}}$ flag)	19	1
FPO instruction	7	
BRKCS instruction	–	

4.11.1 General software interrupts

The execution sequence for acknowledging software interrupts other than I/O instruction interrupts or FPO instruction interrupts is the same as that of vectored interrupts. That is, address information for the next instruction and the PSW are saved on a stack, IE and BRK are both set to 0, and the vector contents are loaded into the PS and PC.

The software interrupts are described below.

(1) DIVU and DIV divide errors

A software interrupt occurs whenever the quotient overflows while executing a divide instruction.

(2) CHKIND boundary over

When the CHKIND instruction is executed to check whether or not the index value exceeds the predefined array boundaries, a software interrupt occurs if it is determined that the index value exceeds the boundaries.

(3) BRKV

A software interrupt occurs when the overflow flag (V) is set during execution of the BRKV instruction.

(4) BRK 3

A software interrupt occurs when the BRK 3 instruction is executed.

(5) BRK imm8

A software interrupt occurs when the BRK imm8 instruction is executed.

(6) BRK flag (single step)

If BRK is set to 1, a software interrupt occurs whenever one instruction is executed. If a repeat prefix is added, it does not occur until the loop is exited.

4.11.2 I/O instruction interrupts

If an attempt is made to execute an I/O instruction when $\overline{\text{IBRK}} = 0$, an interrupt occurs. When the interrupt is acknowledged, address information saved on a stack becomes the address where the I/O instruction is placed, unlike general software interrupts (see section 4.11.1). If a prefix is added to the I/O instruction, the address information becomes the address where the prefix is placed. Other operations are the same as general software interrupts. When control is returned from the I/O instruction interrupt, the PC value in the stack must be adjusted for a normal return.

If the address information saved on the stack is made the instruction's starting address, the software can be used to determine specifically which instruction was attempted when the interrupt occurred. This function enables easy migration of programs formerly used with the V20 and V30.

The PSW contents immediately prior to the interrupt occurrence are saved on the stack, then $\text{IE} = \text{BRK} = 0$ and $\overline{\text{IBRK}} = 1$ are set automatically. By setting $\overline{\text{IBRK}} = 1$, any I/O instruction is executed as an I/O instruction during interrupt servicing. When control is returned from the interrupt, $\overline{\text{IBRK}}$ is automatically reset to 0.

Example 1. Without prefix



Example 2. With prefix



Reference General software interrupt



4.11.3 FPO instruction interrupt

Because the V25 and V35 differ from the V20 and V30 in their external bus structure, a floating-point math coprocessor for the V20 and V30 cannot be connected. Therefore, an interrupt is generated to emulate the operation of the FPO instruction when an attempt is made to execute the FPO instruction for the coprocessor. The PC value saved on the stack becomes the starting address of the FPO instruction emulated by this interrupt (when a prefix is added, the starting address of the prefix), as is the case for I/O instruction interrupts (see section **4.11.2 I/O instruction interrupts**). Accordingly, the FPO instruction can be decoded for emulation by software. When control is returned from the FPO instruction interrupt, the PC value saved on the stack must be adjusted as with I/O instruction interrupts.

4.12 If Interrupt Requests cannot be Acknowledged

If interrupt requests cannot be acknowledged, check the following items.

- Confirm that the IF bit in the interrupt request control register is set to 1. If it has been reset to 0, check whether or not it was replaced (written over) by software.
- Confirm that the interrupt request control register's IMK bit is set to 1.
- Confirm that the ISPR register's bit having the highest priority level is set to 1.
- Confirm that the PSW's IE flag is in the DI state.

4.13 Timing at which an Interrupt cannot be Acknowledged

No interrupt can be acknowledged between each of the following instructions and the next instruction.

(i) sreg manipulating instructions

MOV sreg, reg16 ; MOV sreg, mem16 ; POP sreg ; POP PSW ; MOVSPB

(ii) Prefix instructions

PS:, SS:, DS0:, DS1:, REPC, REPNC, REP, REPE, REPZ, REPNE, REPNZ, BUSLOCK

(iii) EI, RETI, DI

(iv) FINT

Each interrupt except INT that occurs at a timing where no interrupt can be acknowledged will be acknowledged if it can be acknowledged after termination of the next instruction's execution.

4.14 Interrupt Servicing during Execution of Block Servicing Instruction

An interrupt is acknowledged during execution of a block servicing (transfer, comparison, retrieval, or I/O) instruction.

The interrupt is acknowledged at the termination of one servicing instruction's execution. At that time, the address saved on the stack or the PC save area in the register bank automatically becomes the top of the instruction containing the prefix. The incomplete block servicing can be resumed by re-executing from the prefix when returning from the interrupt.

Theoretically, up to three types of prefixes including repeat prefixes can be added to the block servicing instruction. The V25 and V35 enable the user to determine which type of block servicing instruction with prefix was being executed when the interrupt was acknowledged. The PC value is automatically decremented and saved in an area such as the stack according to the prefix addition state.

Example REP

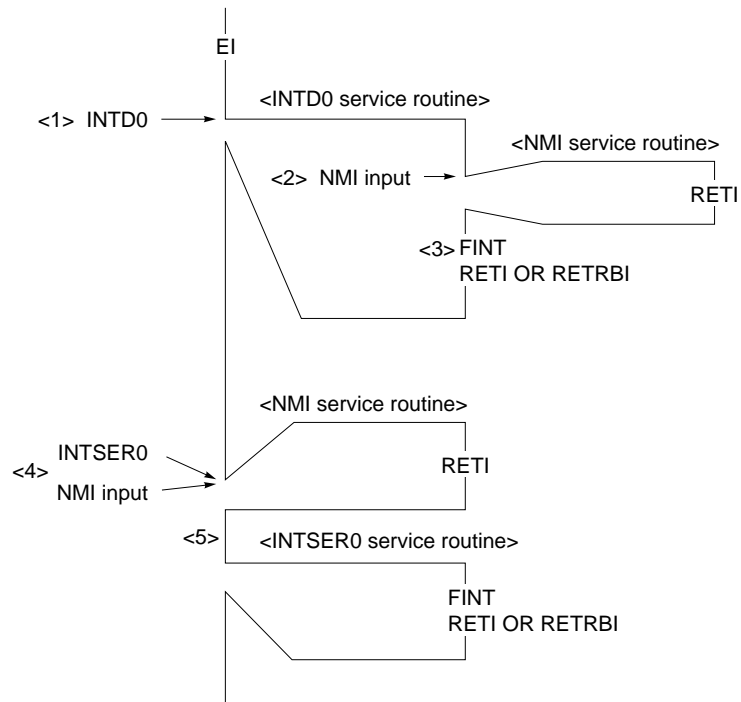
```
MOVBK SS:SRC BLK, DES BLOCK
```

4.15 How Interrupts are Acknowledged

(1) NMI

- o Is not masked by software.
- o Takes precedence over all other interrupts.

<Main routine>

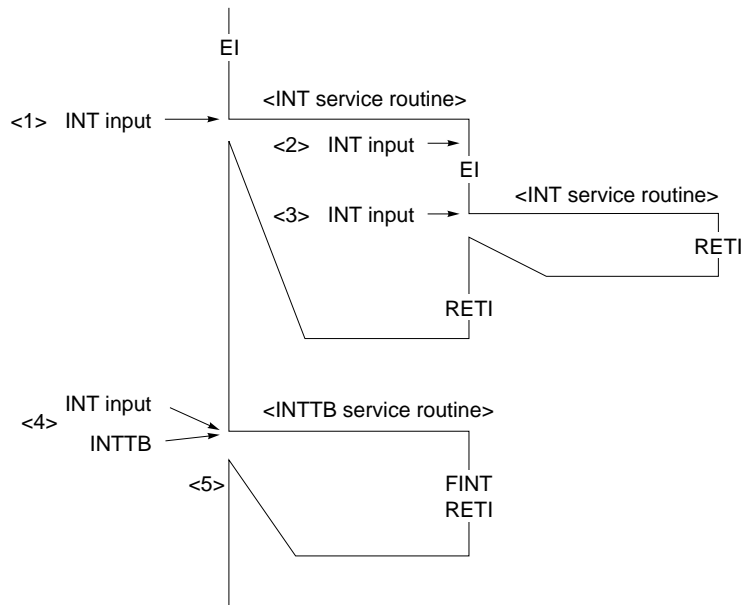


- <1> INTD0 (specified priority level 2) occurs in the interrupt enable state (EI) and INTD0 servicing is started.
- <2> Interrupts are disabled (DI) during INTD0 servicing. When NMI occurs, INTD0 servicing is disabled and NMI servicing is started.
- <3> When NMI servicing is completed, previously disabled INTD0 servicing is resumed.
- <4> When INTSER0 (specified priority level 4) and NMI occur at the same time, the NMI which takes precedence over all other interrupts is acknowledged and NMI servicing is started.
- <5> When the NMI servicing is completed, the pending INTSER0 is serviced.

(2) INT

- o Is always acknowledged if EI is set to 1 (interrupt enable state).
- o Is specified the lowest priority level when multiple interrupts occur.
- o Is not subject to the multiple interrupt servicing controller.

<Main routine>

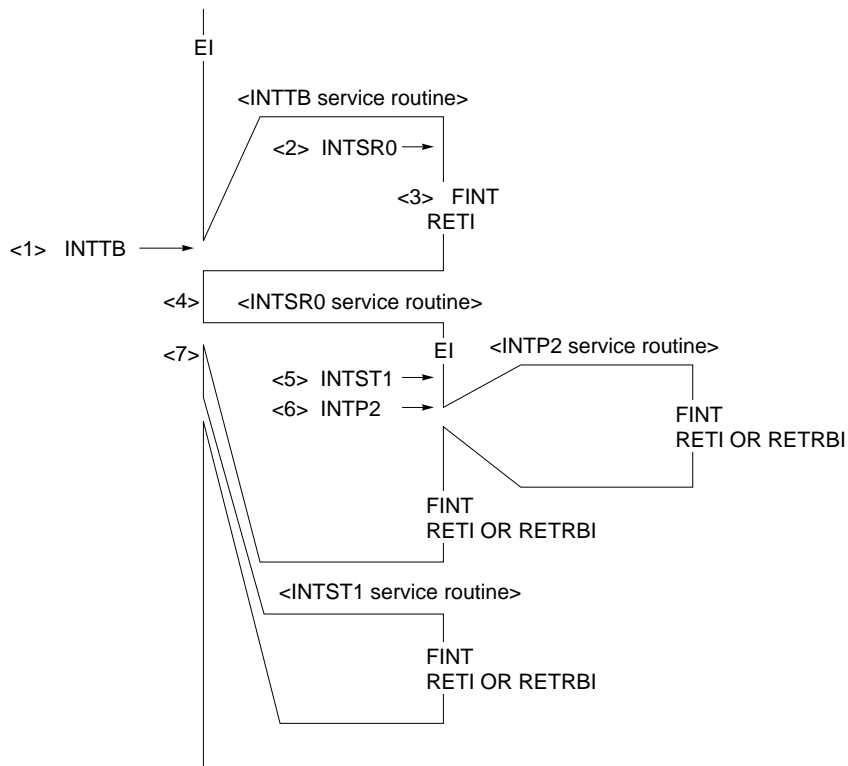


- <1> When INT occurs in the interrupt enable state (EI), INT servicing is started.
- <2> When the INT is acknowledged, interrupts are automatically disabled (DI). If another INT occurs, it is not acknowledged.
- <3> Even when EI is set to 1 (interrupt enable state) during INT servicing, multiple servicing of INT is performed if another INT occurs.
- <4> When INT and INTTB (specified priority level 6) occur at the same time, the INT which has the lowest priority level is not acknowledged, and INTTB servicing is started.
- <5> If INT is inactive after INTTB servicing is completed, the INT interrupt is not acknowledged.

(3) Interrupts subject to multiprocessing control

- o Multiple servicing is performing according to the priority levels.

<Main routine>

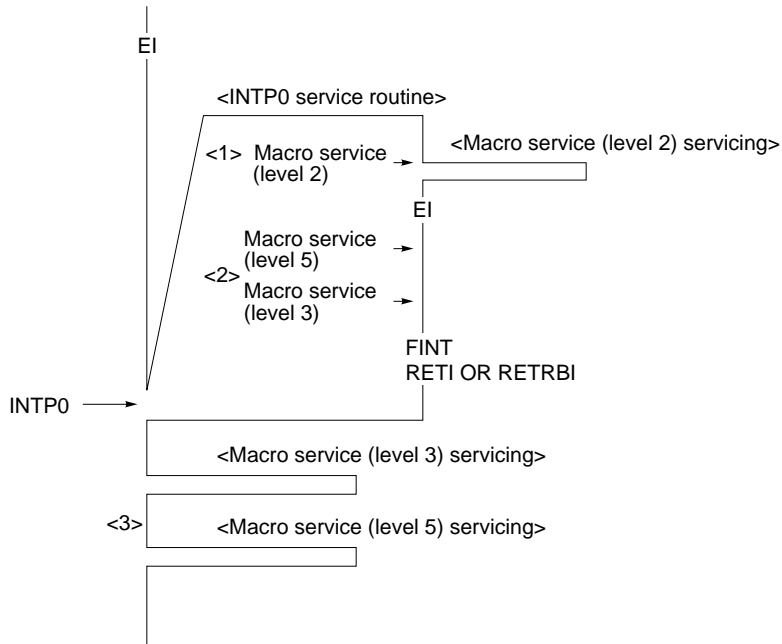


- <1> INTTB (specified priority level 6) occurs in the interrupt enable state (EI) and INTTB servicing is started.
- <2> When the INTTB is acknowledged, interrupts are disabled (DI). Although INTSR0 (specified priority level 4) occurs and is higher than the INTTB priority level, it is not acknowledged.
- <3> The FINT instruction must be executed for interrupts subject to priority level control when returning control from the interrupt.
- <4> When INTTB servicing is completed and interrupts are enabled (EI), pending INTSR0 servicing is started.
- <5> INTST1 (specified priority level 5) occurs in the interrupt enable state (EI). Since INTSR0 interrupt servicing is being performed and INTST1 has a lower priority level than INTSR0, the INTST1 is not acknowledged.
- <6> INTP2 (specified priority level 3) occurs. Because it has a higher priority level than INTSR0, INTP2 is acknowledged and INTP2 servicing is started.
- <7> INTP2 servicing is completed. When pending INTSR0 servicing is also completed and a return is made, INTST1 is acknowledged.

(4) Macro service interrupt

- o Is acknowledged regardless of the interrupt enable (EI) or disable (DI) state.
- o Multiple servicing is performed according to the priority levels as with the interrupts subject to multiple servicing control.

<Main routine>

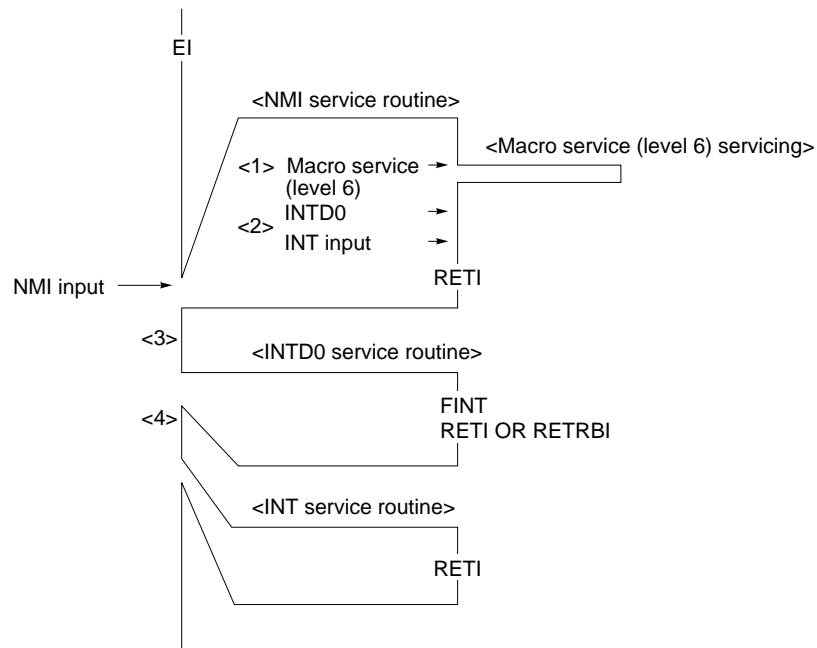


- <1> When a macro service interrupt (specified priority level 2) occurs during servicing of INTPO (specified priority level 3), macro service servicing is performed even though interrupts are disabled (DI).
- <2> Macro service interrupts (specified priority levels 5 and 3) occur during servicing of INTPO placed in the interrupt enable state (EI). The macro service interrupt priority levels are lower than the INTPO priority level, and therefore interrupts are held pending.
- <3> When the INTPO servicing is completed, the pending macro service interrupts are acknowledged according to their priority levels.

(5) Macro service interrupt priority levels for other interrupts

o A macro service interrupt is also acknowledged during NMI servicing.

<Main routine>

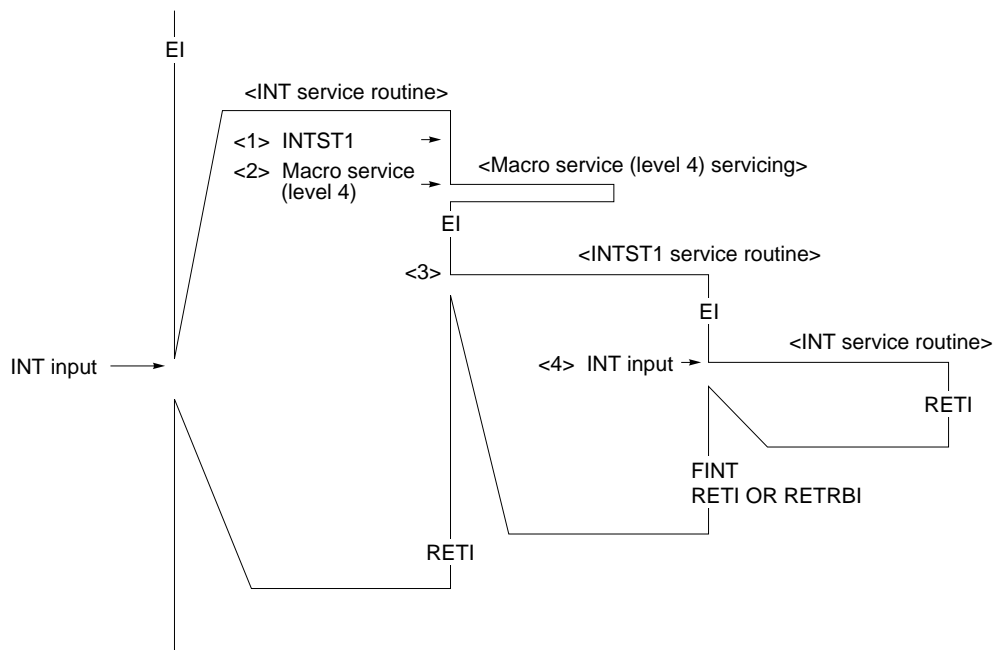


- <1> Even if interrupts are disabled (DI) during NMI servicing, macro service servicing is performed when a macro service interrupt occurs.
- <2> When interrupts are disabled (DI) during NMI servicing, even if an interrupt subject to multiple servicing control (INTD0) or INT occurs, it is not acknowledged.
- <3> When the NMI servicing is completed, the pending INTD0 is acknowledged.
- <4> If INT is active at the completion of INTD0 servicing, INT is acknowledged.

(6) Multiple interrupt servicing

- o INT and other interrupts subject to multiple servicing control, and macro service interrupts

<Main routine>

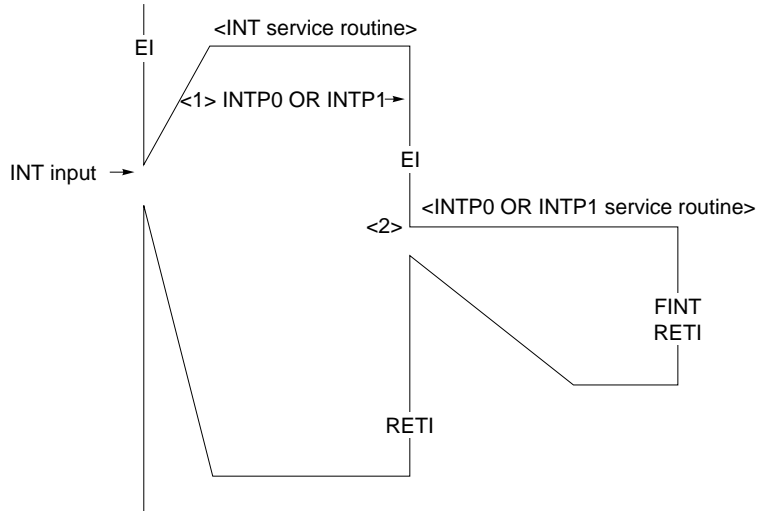


- <1> Because INT is acknowledged and interrupts are disabled (DI), INTST1 (specified priority level 5) occurs, but is not acknowledged.
- <2> When a macro service interrupt (specified priority level 4) occurs, macro service servicing is performed.
- <3> When EI is set to 1 (interrupt enable state), pending INTST1 servicing is started.
- <4> If interrupts are enabled (EI) during INTST1 servicing, INT servicing is started when INT occurs.

(7) Priority levels of external interrupts

- o If an external interrupt for INTP0 or INTP1 input occurs during servicing of an INT interrupt service routine, the interrupt is acknowledged if the interrupt flag is in the EI state.

<Main routine>

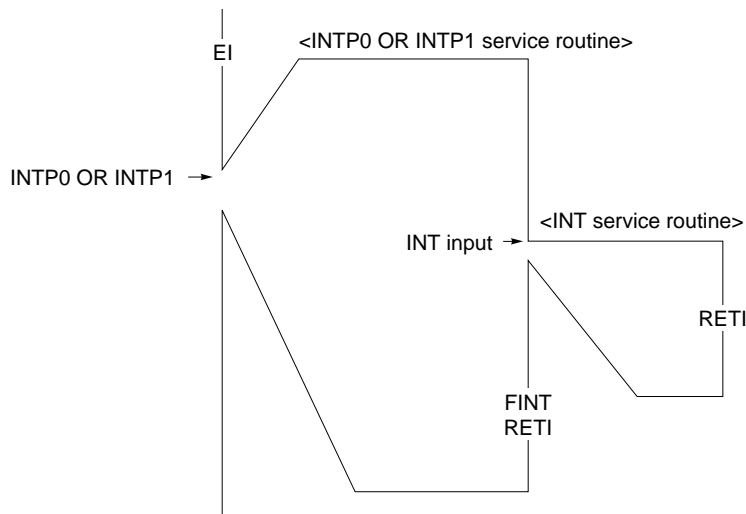


<1> Because the INT is acknowledged and the interrupt disable state (DI) is in effect, if interrupt requests for INTP0 or INTP1 occur they are not acknowledged.

<2> When the interrupt flag is in the enable state (EI), servicing is started for the pending INTP0 or INTP1 interrupt.

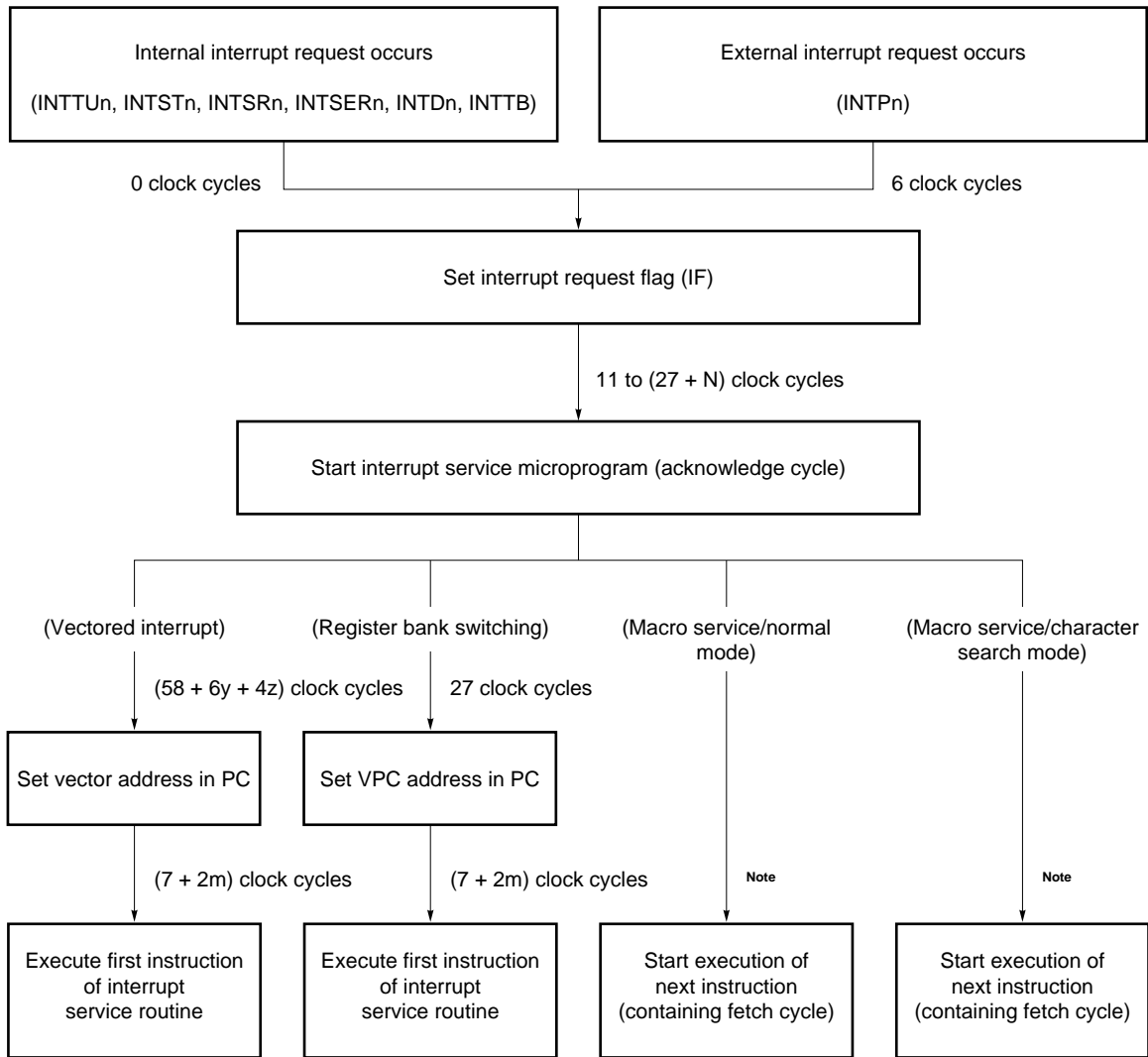
- o If an INT input occurs during execution of an INTP0 or INTP1 interrupt, the INT input is acknowledged at any time when the interrupt flag is in the EI state.

<Main routine>



4.16 Hardware Interrupt Response Time

4.16.1 V25's interrupt response time (number of system clock cycles)



Note Macro service transfer processing time (number of system clock cycles)

- N : Remaining number of execution clock cycles for the instruction being executed by the CPU at this time
- y : Number of wait cycles for memory when PC, PS, and PSW are saved on a stack
- z : Number of wait cycles for memory when vector PC or vector PS is read
- m : Number of wait cycles for memory when the first instruction of interrupt service routine is fetched (when two bytes are fetched, instruction execution is started)

Caution Refresh cycle, hold request, DMA request, other interrupt requests, etc., are not considered.

Normal mode (unit: clock cycles)

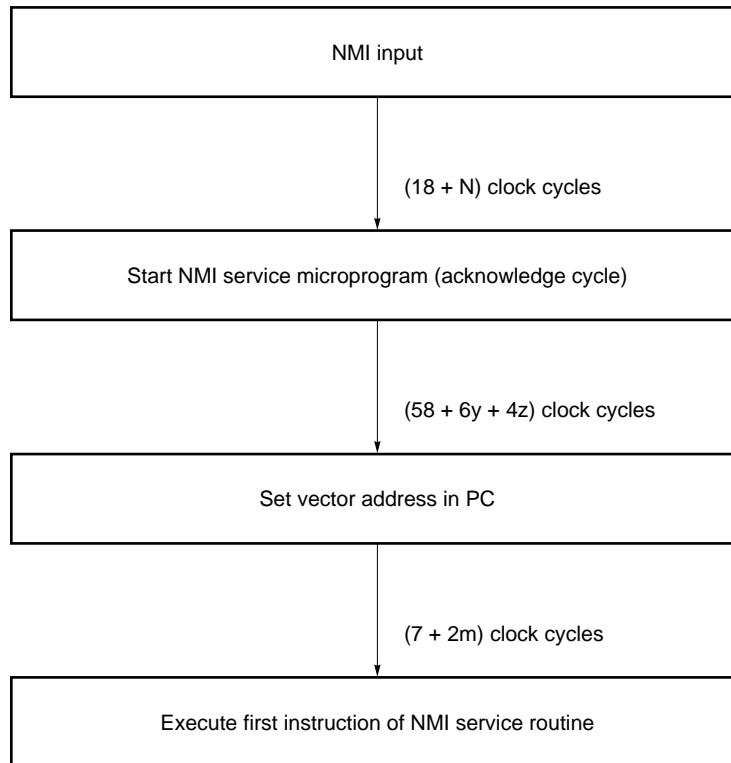
	Byte transfer		Word transfer	
	On-chip RAM access enabled	On-chip RAM access disabled	On-chip RAM access enabled	On-chip RAM access disabled
Memory to SFR	$24 + t$	$19 + t$	$26 + 2t$	$21 + 2t$
SFR to memory	$22 + t$	$20 + t$	$22 + 2t$	$22 + 2t$

Character search mode (unit: clock cycles)

	Byte transfer	
	On-chip RAM access enabled	On-chip RAM access disabled
Memory to SFR	$27 + t$	$27 + t$
SFR to memory	$37 + t$	$34 + t$

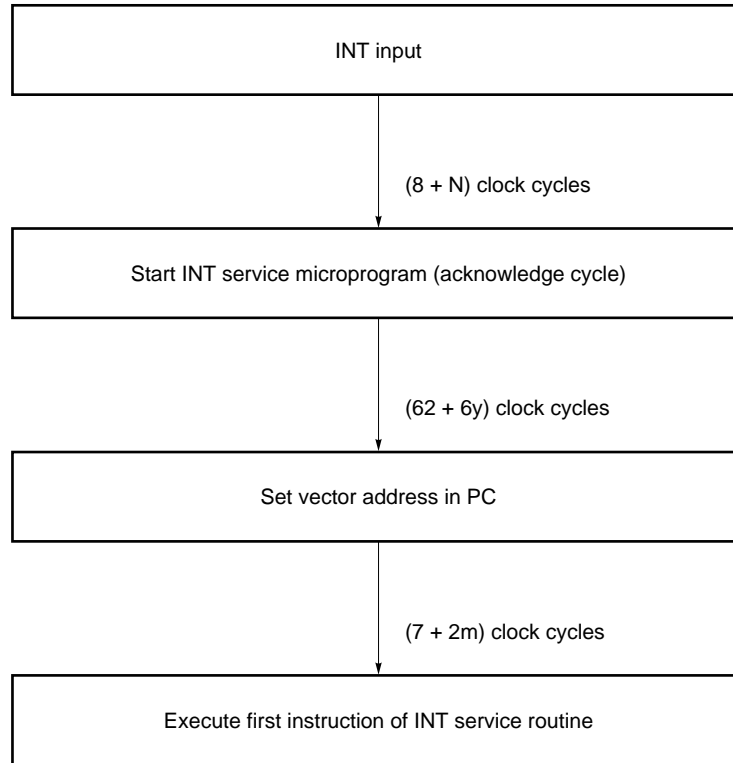
t: Number of wait cycles for memory used for data transfer

4.16.2 V25's NMI response time (number of system clock cycles)



- N : Remaining number of execution clock cycles for the instruction being executed by the CPU at this time
 y : Number of wait cycles for memory when PC, PS, and PSW are saved on a stack
 z : Number of wait cycles for memory when vector PC or vector PS is read
 m : Number of wait cycles for memory when the first instruction of interrupt service routine is fetched (when two bytes are fetched, instruction execution is started)

4.16.3 V25's INT response time (number of system clock cycles)



- N : Remaining number of execution clock cycles for the instruction being executed by the CPU at this time
- y : Number of wait cycles for memory when PC, PS, and PSW are saved on a stack
- z : Number of wait cycles for memory when vector PC or vector PS is read
- m : Number of wait cycles for memory when the first instruction of interrupt service routine is fetched (when two bytes are fetched, instruction execution is started)

Normal mode (unit: clock cycles)

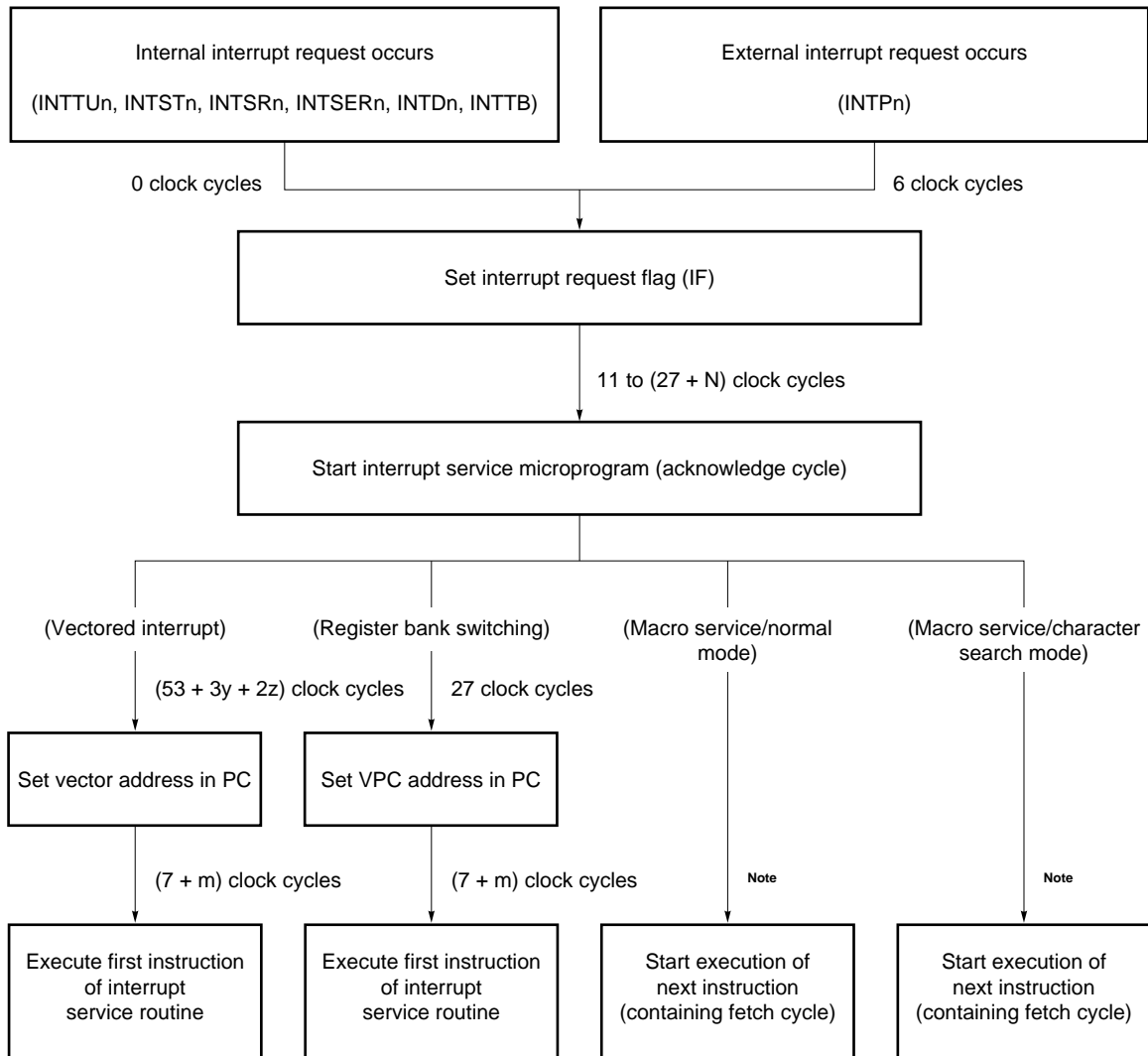
	Byte transfer		Word transfer	
	On-chip RAM access enabled	On-chip RAM access disabled	On-chip RAM access enabled	On-chip RAM access disabled
Memory to SFR	25 + t	20 + t	25 + t	20 + t
SFR to memory	22 + t	21 + t	22 + t	21 + t

Character search mode (unit: clock cycles)

	Byte transfer	
	On-chip RAM access enabled	On-chip RAM access disabled
Memory to SFR	28 + t	28 + t
SFR to memory	38 + t	35 + t

t: Number of wait cycles for memory used for data transfer

4.16.4 V35's interrupt response time (number of system clock cycles)

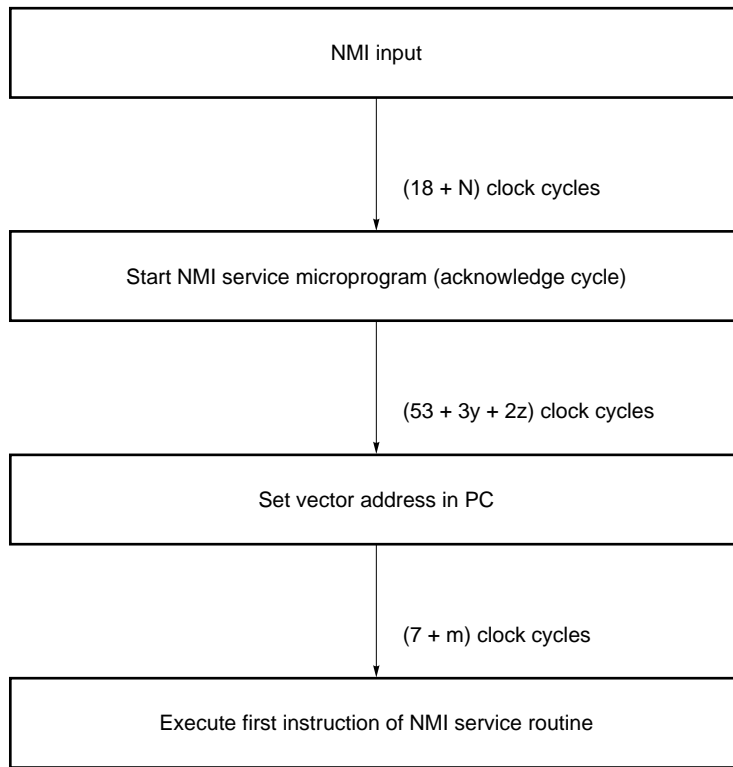


Note Macro service transfer processing time (number of system clock cycles)

- N : Remaining number of execution clock cycles for the instruction being executed by the CPU at this time
- y : Number of wait cycles for memory when PC, PS, and PSW are saved on a stack
- z : Number of wait cycles for memory when vector PC or vector PS is read
- m : Number of wait cycles for memory when the first instruction of interrupt service routine is fetched (when two bytes are fetched, instruction execution is started)

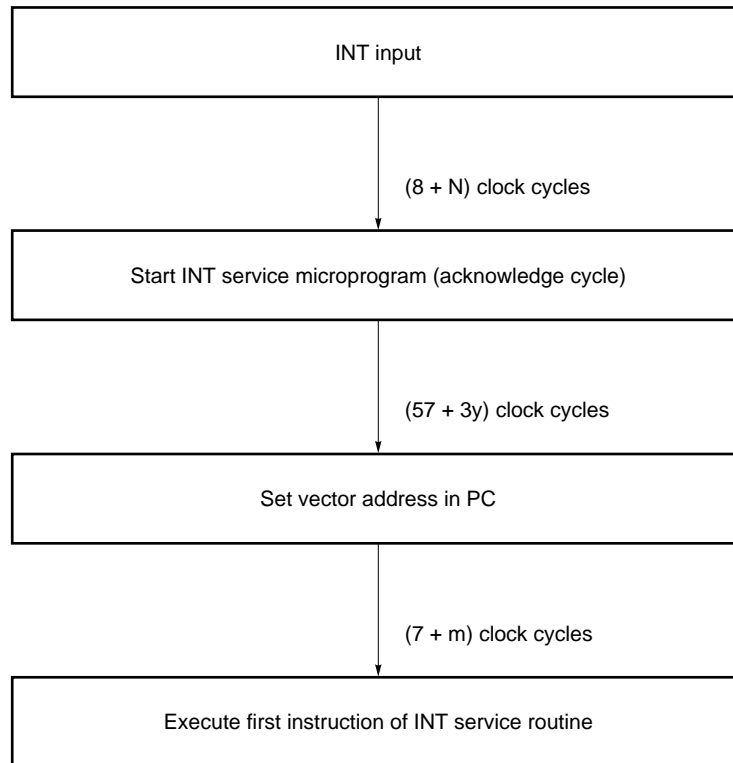
Caution Refresh cycle, hold request, DMA request, other interrupt requests, etc., are not considered.

4.16.5 V35's NMI response time (number of system clock cycles)



- N : Remaining number of execution clock cycles for the instruction being executed by the CPU at this time
 y : Number of wait cycles for memory when PC, PS, and PSW are saved on a stack
 z : Number of wait cycles for memory when vector PC or vector PS is read
 m : Number of wait cycles for memory when the first instruction of interrupt service routine is fetched (when two bytes are fetched, instruction execution is started)

4.16.6 V35's INT response time (number of system clock cycles)



- N : Remaining number of execution clock cycles for the instruction being executed by the CPU at this time
 y : Number of wait cycles for memory when PC, PS, and PSW are saved on a stack
 z : Number of wait cycles for memory when vector PC or vector PS is read
 m : Number of wait cycles for memory when the first instruction of interrupt service routine is fetched (when two bytes are fetched, instruction execution is started)

[MEMO]

CHAPTER 5 BUS CONTROL

The V25 and V35 have the bus control pins listed in parts (a) and (b) of Table 5-1.

When using the functions of the shared pins, the desired function must be selected by setting the port mode control register (PMCn).

Table 5-1. Bus Control Pin Functions

(a) V25

Pin name	I/O	Function	Remark
A0 to A19	O	Address bus	
D0 to D7	I/O	Data bus	
R/W	O	Read/write identification	
$\overline{\text{MREQ}}$	O	Indicates memory cycle	
$\overline{\text{MSTB}}$	O	Memory read/memory write strobe signal	
$\overline{\text{IOSTB}}$	O	I/O cycle strobe signal	
$\overline{\text{REFRQ}}$	O	Indicates memory refresh cycle	
HLD $\overline{\text{RQ}}$	I	Bus hold request signal	Also used for P27
$\overline{\text{HLD}}\overline{\text{AK}}$	O	Bus hold acknowledge signal	Also used for P26
DMARQ0	I	DMA request signal	Also used for P20
DMARQ1	I	DMA request signal	Also used for P23
$\overline{\text{DMA}}\overline{\text{AK}}0$	O	Indicates DMA acknowledge cycle	Also used for P21
$\overline{\text{DMA}}\overline{\text{AK}}1$	O	Indicates DMA acknowledge cycle	Also used for P24
READY	I	Wait insertion in bus cycle from external source	Also used for P17
$\overline{\text{INT}}\overline{\text{AK}}$	O	Indicates interrupt acknowledge cycle	Also used for P13 and $\overline{\text{INT}}\overline{\text{P}}2$

The V25's $\overline{\text{MSTB}}$ signal becomes active 1/2 clock cycles behind the $\overline{\text{MREQ}}$ signal and becomes inactive simultaneously with the $\overline{\text{MREQ}}$ signal. When generating a memory access signal, using the $\overline{\text{MREQ}}$ signal instead of the $\overline{\text{MSTB}}$ signal enables the memory access signal to become active 1/2 clock cycles faster.

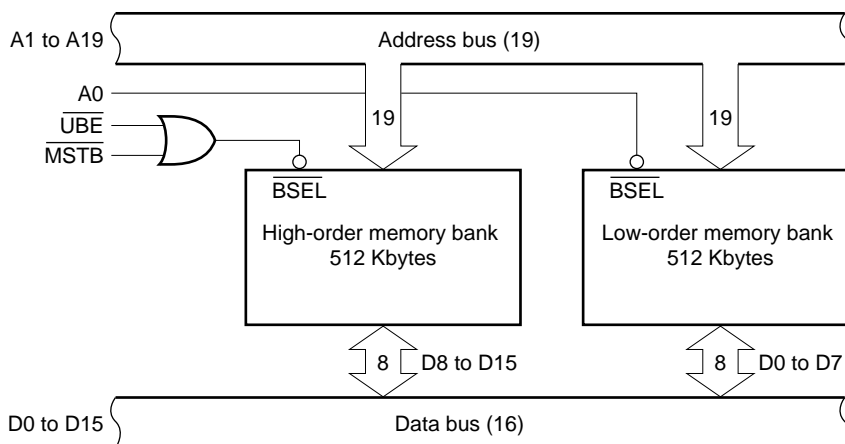
(b) V35

Pin name	I/O	Function	Remark
A0	O	Used for address LSB output and low-order memory bank selection	
A9/A1 to A16/A8, A17/A18,A19	O	19-bit address output by multiplexing	
A18/ \overline{UBE}	O	Address bit 18 output and high-order memory bank selection signal output by multiplexing	
D0 to D15	I/O	Data bus	
$\overline{R/W}$	O	Read/write identification	
\overline{MREQ}	O	Indicates that a bus cycle has started. High-order address strobe signal	
\overline{MSTB}	O	Memory read/memory write strobe signal. Low-order address strobe signal	
\overline{IOSTB}	O	I/O cycle strobe signal. Low-order address strobe signal	
\overline{REFRQ}	O	Indicates memory refresh cycle	
HLDRQ	I	Bus hold request signal	Also used for P27
\overline{HLDAK}	O	Bus hold acknowledge signal	Also used for P26
DMARQ0	I	DMA request signal	Also used for P20
DMARQ1	I	DMA request signal	Also used for P23
$\overline{DMAAK0}$	O	Indicates DMA acknowledge cycle	Also used for P21
$\overline{DMAAK1}$	O	Indicates DMA acknowledge cycle	Also used for P24
READY	I	Wait insertion in bus cycle from external source	Also used for P17
\overline{INTAK}	O	Indicates interrupt acknowledge cycle	Also used for P13 and $\overline{INTP2}$

Because the V35 manages the memory addresses for each byte and has a 16-bit external data bus, memory is separated into high-order and low-order banks for connection. Figure 5-1 shows an outline of the memory bank structure.

The high-order 19 bits (all except A0) of physical addresses are input to the memory address pins of each of the high-order and low-order banks. The A0 signal is used to select the low-order memory banks, and the A18/ \overline{UBE} signal is used to select the high-order memory bank.

Figure 5-1. Memory Bank Structure



The V35's memory cycle consists of three states—T1, T2, and T3 (see section 5.5 Bus Timings).

In the T1 state, the first address (high-order address) of a 20-bit address is output onto the external address bus. In the T2 state, the second address (low-order address) of a 20-bit address is output onto the external address bus. In the T3 state, data is read and written.

Wait state TW is inserted between the T2 and T3 states of a read cycle. It is inserted between the T1 and T2 states of a write cycle.

From the T1 state to the T3 state, the least significant bit of a physical address is output from the A0 pin. From the A18/ \overline{UBE} pin, the 18th bit of a physical address is output in the T1 state, and the \overline{UBE} signal is output in the state following the T1 state.

Table 5-2 describes the relationship between the A0 and \overline{UBE} signals.

Table 5-2. V35 Data Access

Access	\overline{UBE}	A0	Number of bus cycles
Word at even address	0	0	1
Word at odd address	0	1	2
	1	0	
Byte at even address	1	0	1
Byte at odd address	0	1	1

The I/O read/write cycle timing is the same as the memory read/write cycle timing except that \overline{IOSTB} rather than \overline{MSTB} is activated.

In a memory refresh cycle, the \overline{MREQ} and \overline{MSTB} signals are inactive.

To access the external memory or I/O, the V35 outputs a 20-bit physical address from a total of 12 pins (11 address pins and the A18/ \overline{UBE} pin) in time-division multiplexing, as listed in Table 5-3.

★

Table 5-3. Address Time-Division Multiplexing Output (V35)

Pin name	Memory cycle		I/O cycle		Refresh cycle
	First	Second	First	Second	
A0	A0	A0	A0	A0	"0"
A9/A1	A9	A1	A9	A1	A0
A10/A2	A10	A2	A10	A2	A1
A11/A3	A11	A3	A11	A3	A2
A12/A4	A12	A4	A12	A4	A3
A13/A5	A13	A5	A13	A5	A4
A14/A6	A14	A6	A14	A6	A5
A15/A7	A15	A7	A15	A7	A6
A16/A8	A16	A8	"0"	A8	A7
A17/A18	A17	A18	"0"	"0"	A8
A19	A19	A19	"0"	"0"	"0"
A18/ \overline{UBE}	A18	\overline{UBE}	"0"	\overline{UBE}	"0"
\overline{REFRQ}	"1"	"1"	"1"	"1"	"0"

5.1 Programmable Wait Function

For the V25 and 35, wait state insertion in a bus cycle (except a memory refresh cycle) can be specified by software. It is specified for each of the eight 128-Kbyte blocks of the 1-Mbyte memory space and the I/O space by setting the wait control register (WTC) as shown in Figure 5-2. However, the same value is set for memory space block 6 (C0000H to DFFFFH) and block 7 (E0000H to FFFFFH).

Wait state specification can be selected among four types listed in Table 5-4 as desired for each block. When READY pin control is used, port 1 mode control register (PMC1) bit 7 must be set to 1 because the READY pin is also used for P17. When PMC1 bit 7 is set to 0, the READY state is always set; that is, two wait states are specified. When READY pin control is selected, two wait states of TAW are inserted regardless of the READY pin state. The READY pin is sensed by its level. Wait states are inserted when the pin is low.

Access to the internal data area is not affected by the programmable wait function. These settings are applied to all external area access except for memory refresh.

When RESET is input, the WTC register contents are initialized to FFFFH.

Figure 5-2. WTC

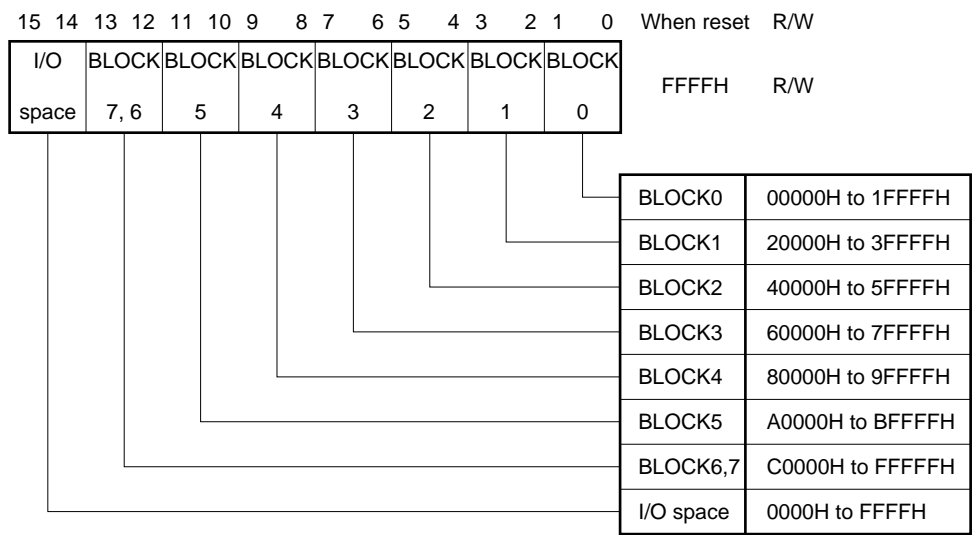


Table 5-4. Wait State Settings

BLOCKn, I/O space	Wait state
00	0 states
01	1 state
10	2 states
11	2 states + inserted state(s) depending on READY pin

When wait control with the READY pin is selected in the wait control register (WTC), the CPU in the V25 or V35 automatically inserts two wait states (TAWs), as described below. Figure 5-3 shows the wait for the READY pin in the V25 and Figure 5-4 shows the wait for the READY pin in the V35.

V25: between T1 and T2 states

V35: between T2 and T3 states

The V25 and V35 sample the READY pin state in the first TAW. At that time, wait state (TW) insertion is enabled or disabled depending on the READY pin's state.

(1) When READY pin is high

This disables wait state (TW) insertion after the automatically inserted TAW. Only set the READY pin high when disabling wait state insertion; otherwise, set it low.

(2) When READY pin is low

This adds wait states (TW) after the automatically inserted TAW. TW is inserted as many times as the READY pin is sampled when low. If the READY pin is high during the first TAW, a READY wait state (TW) will not be inserted afterward even if the next TAW is low.

The V25 and V35 differ in their output timing of control signals $\overline{\text{MREQ}}$, $\overline{\text{MSTB}}$, and $\overline{\text{IOSTB}}$. The READY signal timing in these products is defined as follows.

V25: During the memory read/write cycle, it is defined by the $\overline{\text{MREQ}}$ signal.

During the I/O read/write cycle, it is defined by the $\overline{\text{IOSTB}}$ signal.

V35: During the memory read/write cycle^{Note}, it is defined by the $\overline{\text{MREQ}}$ signal.

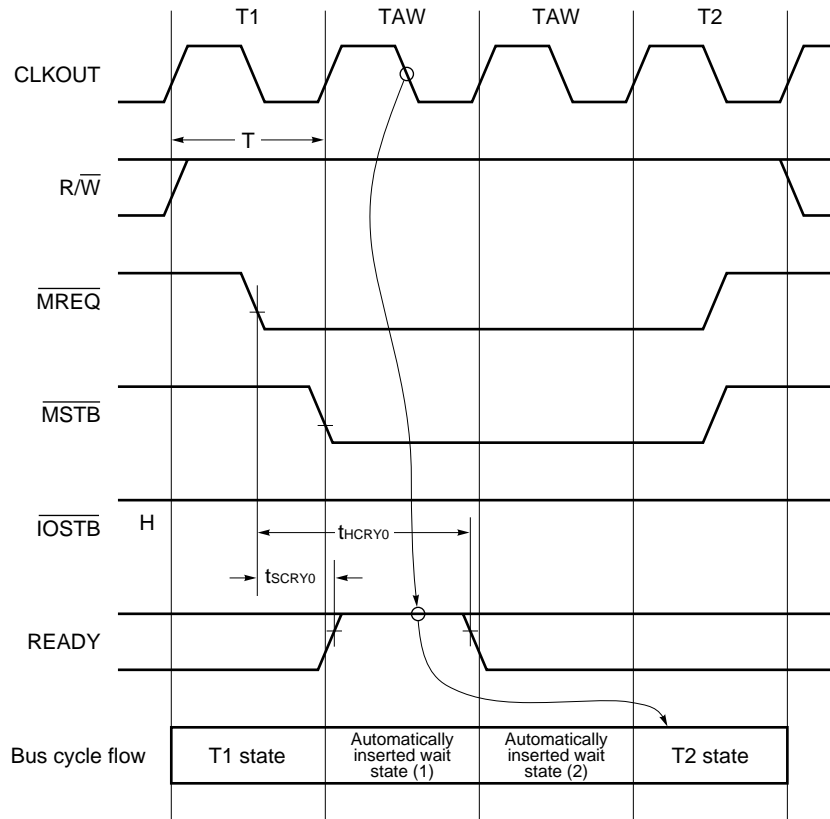
During the I/O read/write cycle, it is defined by the $\overline{\text{MREQ}}$ and $\overline{\text{IOSTB}}$ signals.

Note In the V35, the memory read cycle and memory write cycle use different $\overline{\text{MSTB}}$ output timing.

Caution No refresh cycle is inserted in the wait state. Accordingly, when the DRAM is refreshed using the refresh function, if the wait state is prolonged, refresh is not executed at that time and the DRAM contents may not be retained.

Figure 5-3. Wait via READY Pin (V25) (1/8)

(a) No addition of wait state to memory read cycle



When operation frequency is 8 MHz

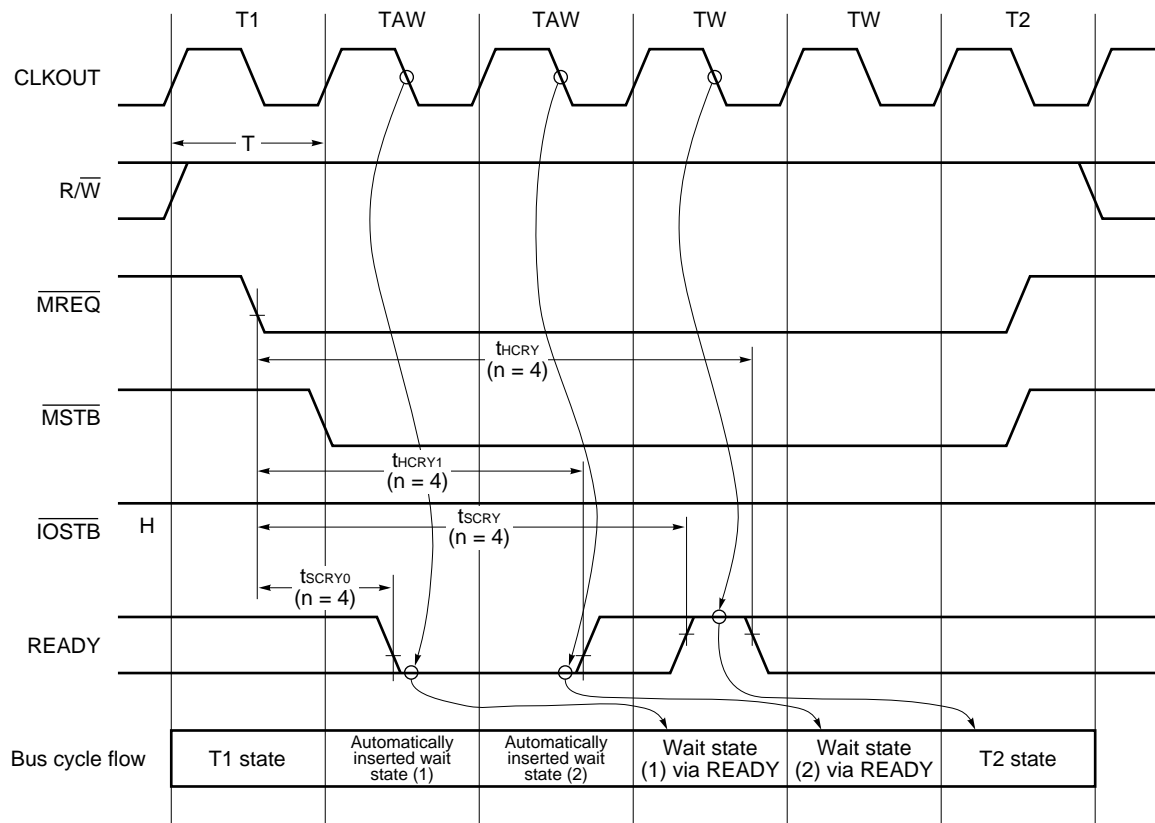
$$T = 125 \text{ ns}$$

$$t_{SCRY0} = T - 100 \text{ (Max.)} = 25 \text{ ns}$$

$$t_{HCRY0} = T \text{ (Min.)} = 125 \text{ ns}$$

Figure 5-3. Wait via READY Pin (V25) (2/8)

(b) Addition of two wait states to memory read cycle



When operation frequency is 8 MHz

$$T = 125 \text{ ns}$$

Assign 4 to n because the total number of wait states is four.

$$t_{SCRY0} = T - 100 \text{ (Max.)} = 25 \text{ ns}$$

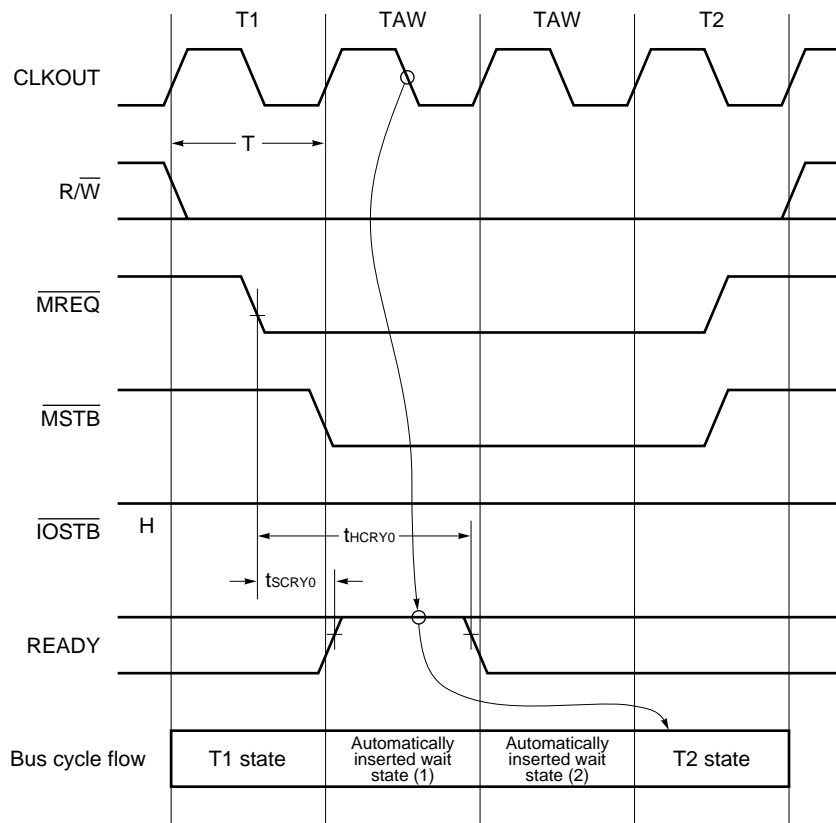
$$t_{HCRY1} = (n - 2) T \text{ (Min.)} = 250 \text{ ns}$$

$$t_{SCRY} = (n - 1) T - 100 \text{ (Max.)} = 275 \text{ ns}$$

$$t_{HCRY} = (n - 1) T \text{ (Min.)} = 375 \text{ ns}$$

Figure 5-3. Wait via READY Pin (V25) (3/8)

(c) No addition of wait state to memory write cycle



When operation frequency is 8 MHz

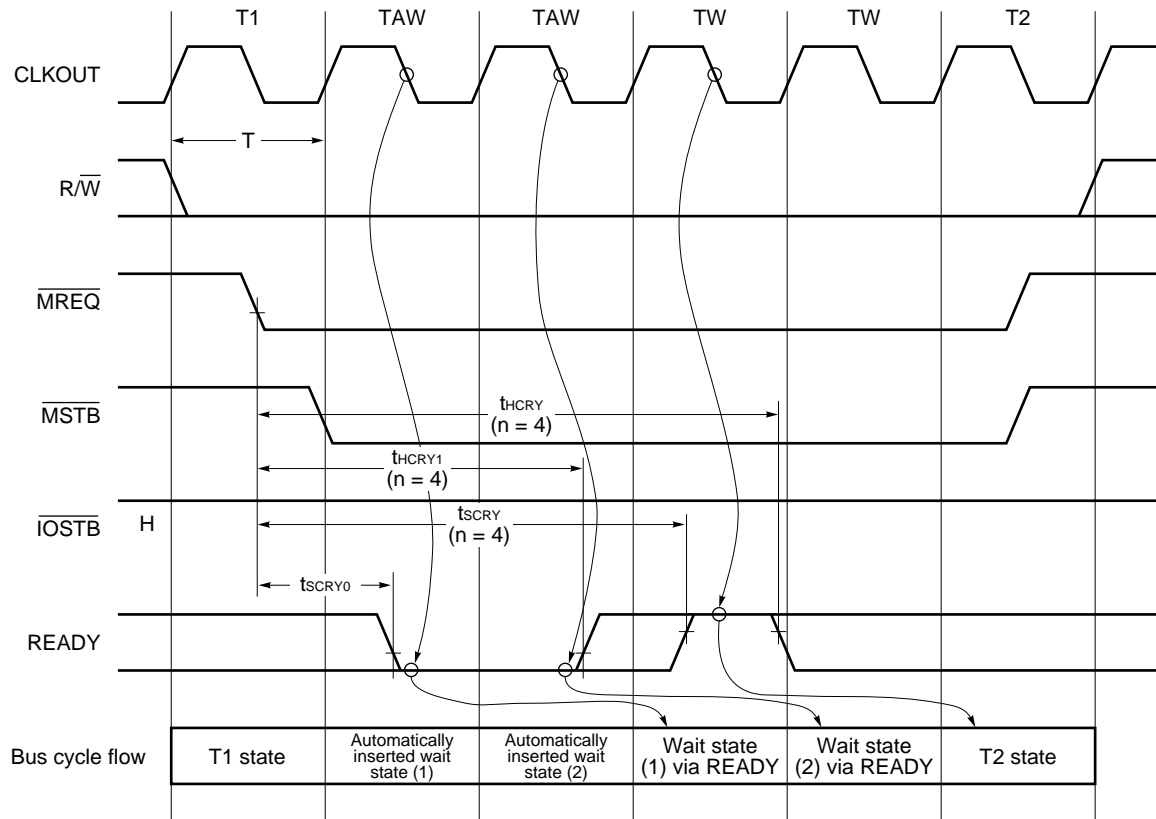
$$T = 125 \text{ ns}$$

$$t_{SCRY0} = T - 100 \text{ (Max.)} = 25 \text{ ns}$$

$$t_{HCRY0} = T \text{ (Min.)} = 125 \text{ ns}$$

Figure 5-3. Wait via READY Pin (V25) (4/8)

(d) Addition of two wait states to memory write cycle



When operation frequency is 8 MHz

$T = 125 \text{ ns}$

Assign 4 to n because the total number of wait states is four.

$t_{SCRY0} = T - 100 \text{ (Max.)} = 25 \text{ ns}$

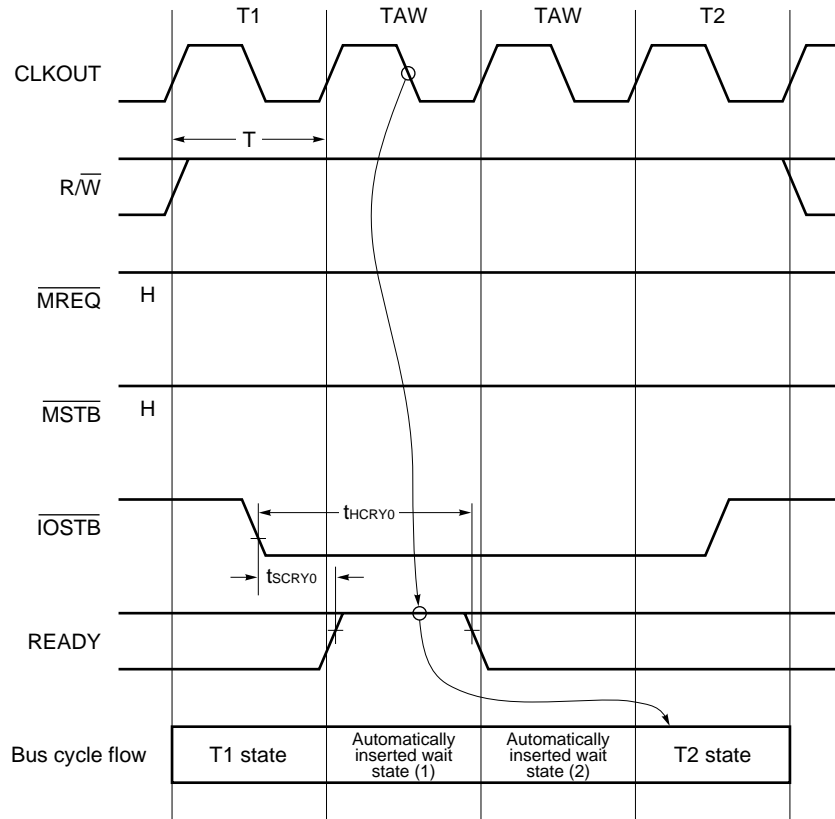
$t_{HCRY1} = (n - 2) T \text{ (Min.)} = 250 \text{ ns}$

$t_{SCRY} = (n - 1) T - 100 \text{ (Max.)} = 275 \text{ ns}$

$t_{HCRY} = (n - 1) T \text{ (Min.)} = 375 \text{ ns}$

Figure 5-3. Wait via READY Pin (V25) (5/8)

(e) No addition of wait state to I/O read cycle



When operation frequency is 8 MHz

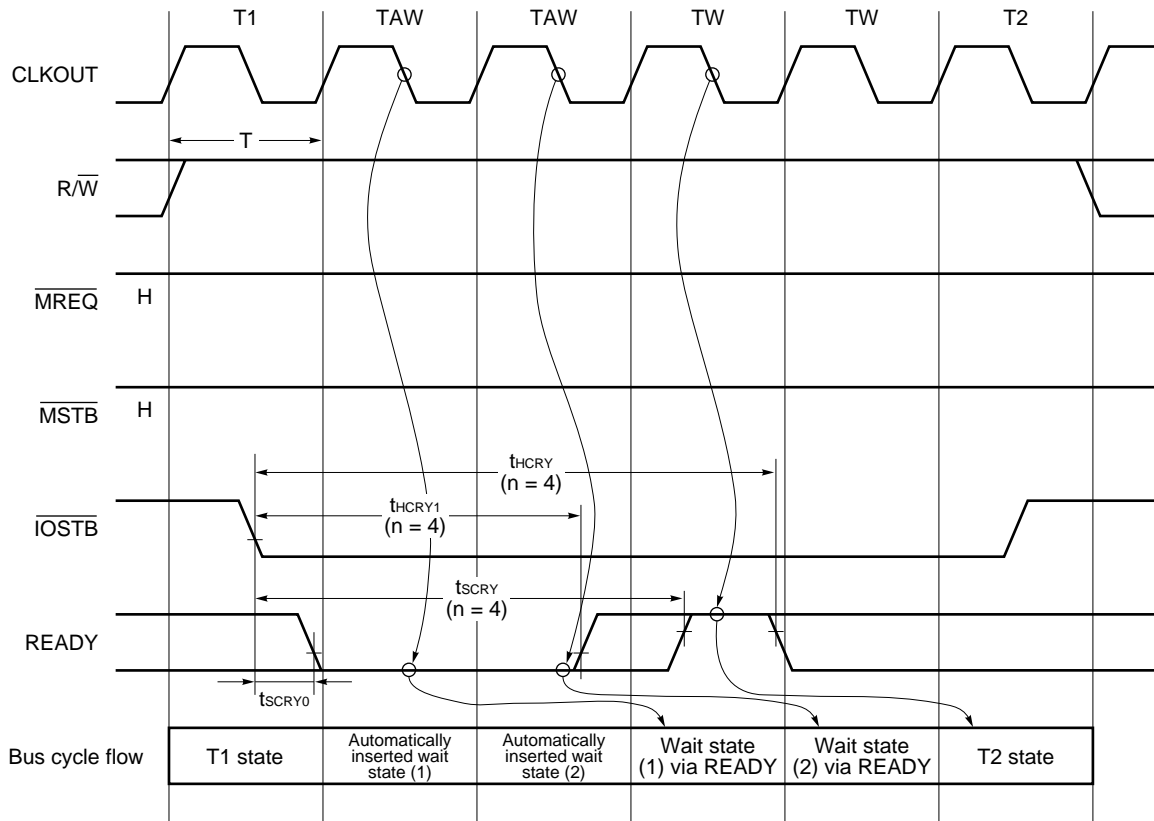
$$T = 125 \text{ ns}$$

$$t_{\text{SCRY0}} = T - 100 \text{ (Max.)} = 25 \text{ ns}$$

$$t_{\text{HCRY0}} = T \text{ (Min.)} = 125 \text{ ns}$$

Figure 5-3. Wait via READY Pin (V25) (6/8)

(f) Addition of two wait states to I/O read cycle



When operation frequency is 8 MHz

$$T = 125 \text{ ns}$$

Assign 4 to n because the total number of wait states is four.

$$t_{SCRY0} = T - 100 \text{ (Max.)} = 25 \text{ ns}$$

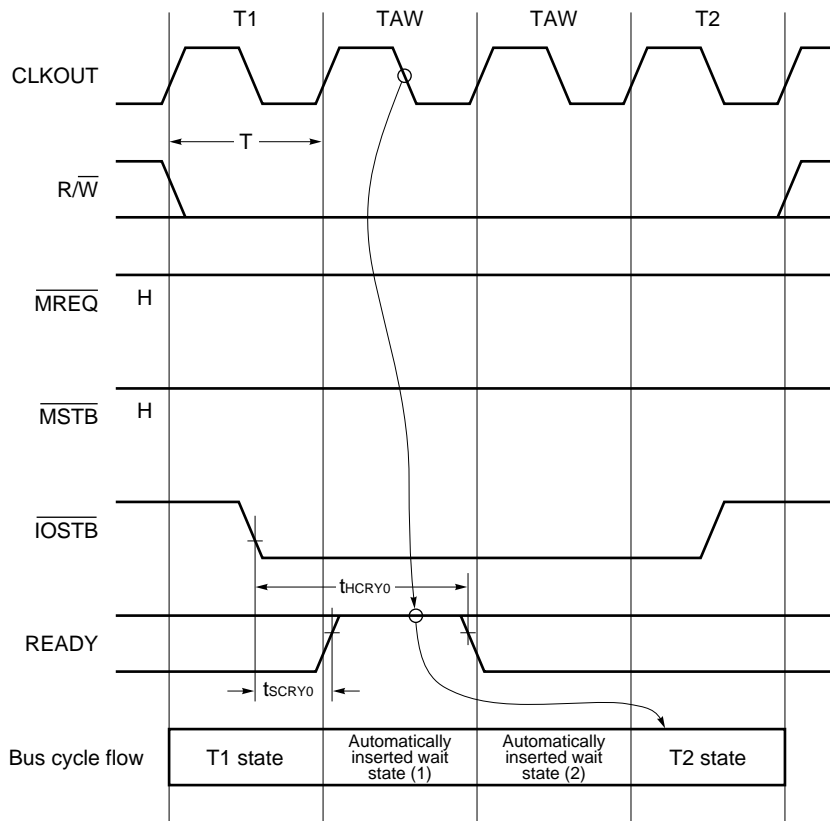
$$t_{HCRY1} = (n - 2) T \text{ (Min.)} = 250 \text{ ns}$$

$$t_{SCRY} = (n - 1) T - 100 \text{ (Max.)} = 275 \text{ ns}$$

$$t_{HCRY} = (n - 1) T \text{ (Min.)} = 375 \text{ ns}$$

Figure 5-3. Wait via READY Pin (V25) (7/8)

(g) No addition of wait state to I/O write cycle



When operation frequency is 8 MHz

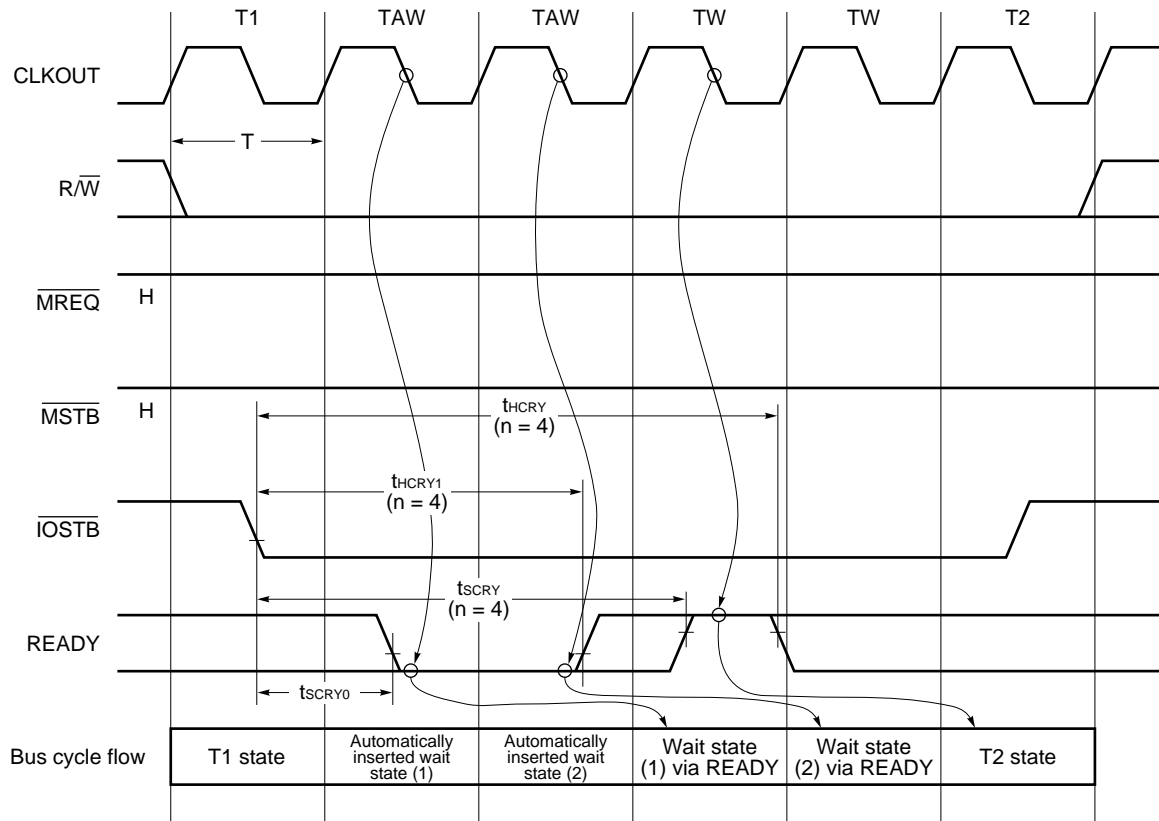
$T = 125 \text{ ns}$

$t_{SCRY0} = T - 100 \text{ (Max.)} = 25 \text{ ns}$

$t_{HCRY0} = T \text{ (Min.)} = 125 \text{ ns}$

Figure 5-3. Wait via READY Pin (V25) (8/8)

(h) Addition of two wait states to I/O write cycle



When operation frequency is 8 MHz

$T = 125 \text{ ns}$

Assign 4 to n because the total number of wait states is four.

$t_{SCRY0} = T - 100 \text{ (Max.)} = 25 \text{ ns}$

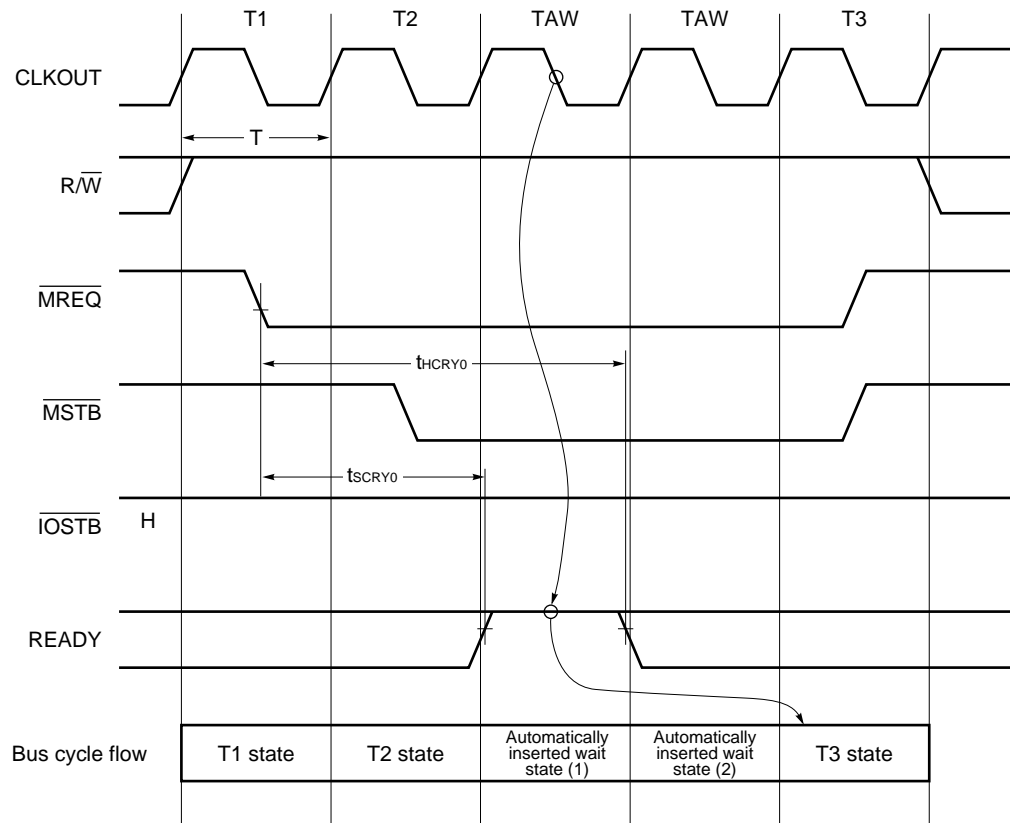
$t_{HCRY1} = (n - 2) T \text{ (Min.)} = 250 \text{ ns}$

$t_{SCRY} = (n - 1) T - 100 \text{ (Max.)} = 275 \text{ ns}$

$t_{HCRY} = (n - 1) T \text{ (Min.)} = 375 \text{ ns}$

Figure 5-4. Wait via READY Pin (V35) (1/8)

(a) No addition of wait state to memory read cycle



When operation frequency is 8 MHz

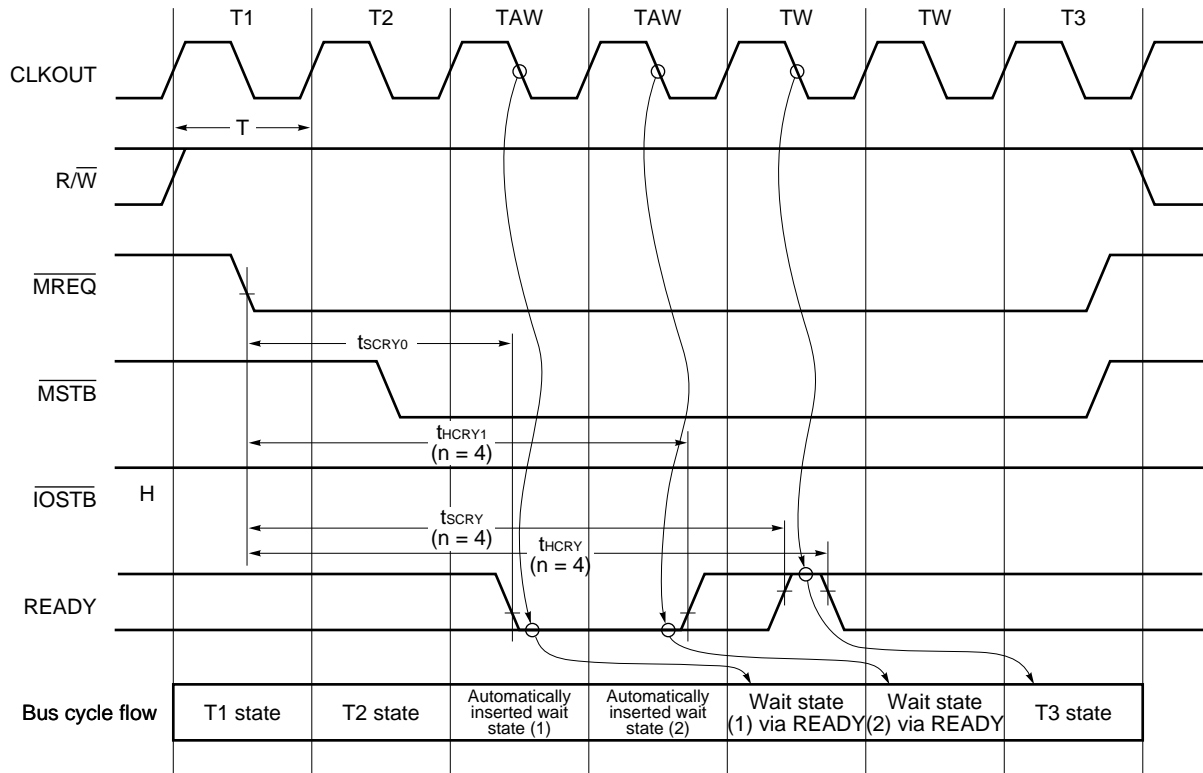
$$T = 125 \text{ ns}$$

$$t_{SCRY0} = 2T - 100 \text{ (Max.)} = 150 \text{ ns}$$

$$t_{HCRY0} = 2T \text{ (Min.)} = 250 \text{ ns}$$

Figure 5-4. Wait via READY Pin (V35) (2/8)

(b) Addition of two wait states to memory read cycle



When operation frequency is 8 MHz

$T = 125 \text{ ns}$

Assign 4 to n because the total number of wait states is four.

$t_{SCRY0} = 2T - 100 \text{ (Max.)} = 150 \text{ ns}$

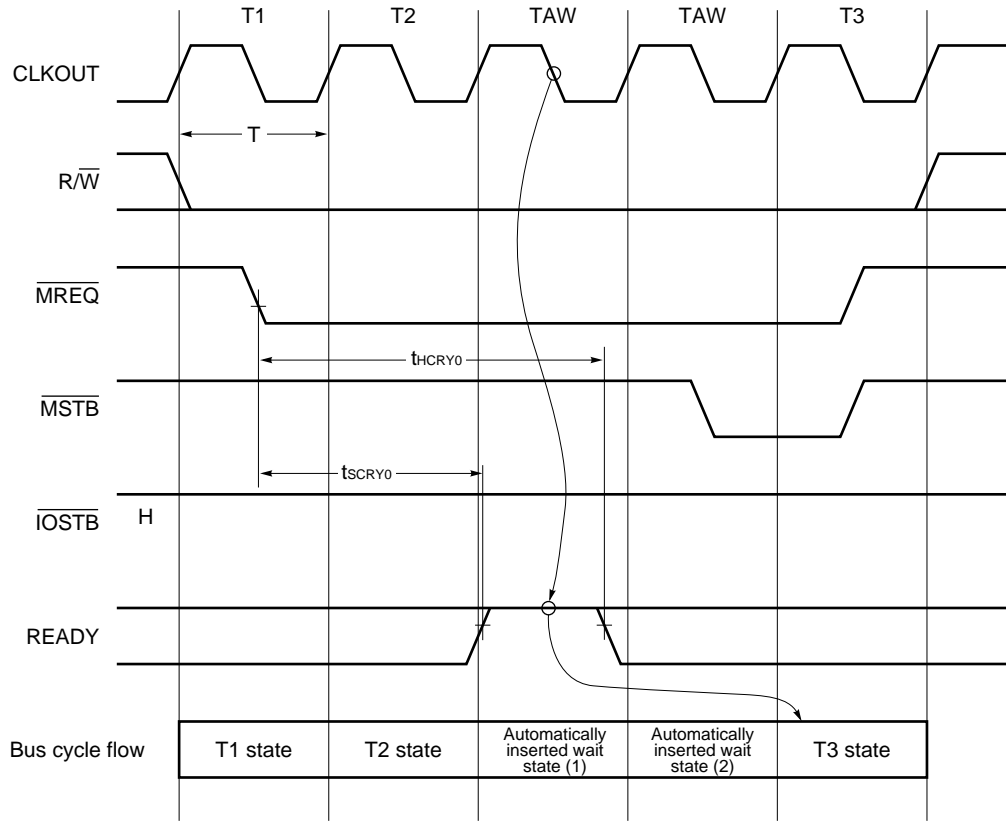
$t_{HCY1} = (n - 1) T \text{ (Min.)} = 375 \text{ ns}$

$t_{SCRY} = nT - 100 \text{ (Max.)} = 400 \text{ ns}$

$t_{HCY} = nT \text{ (Min.)} = 500 \text{ ns}$

Figure 5-4. Wait via READY Pin (V35) (3/8)

(c) No addition of wait state to memory write cycle



When operation frequency is 8 MHz

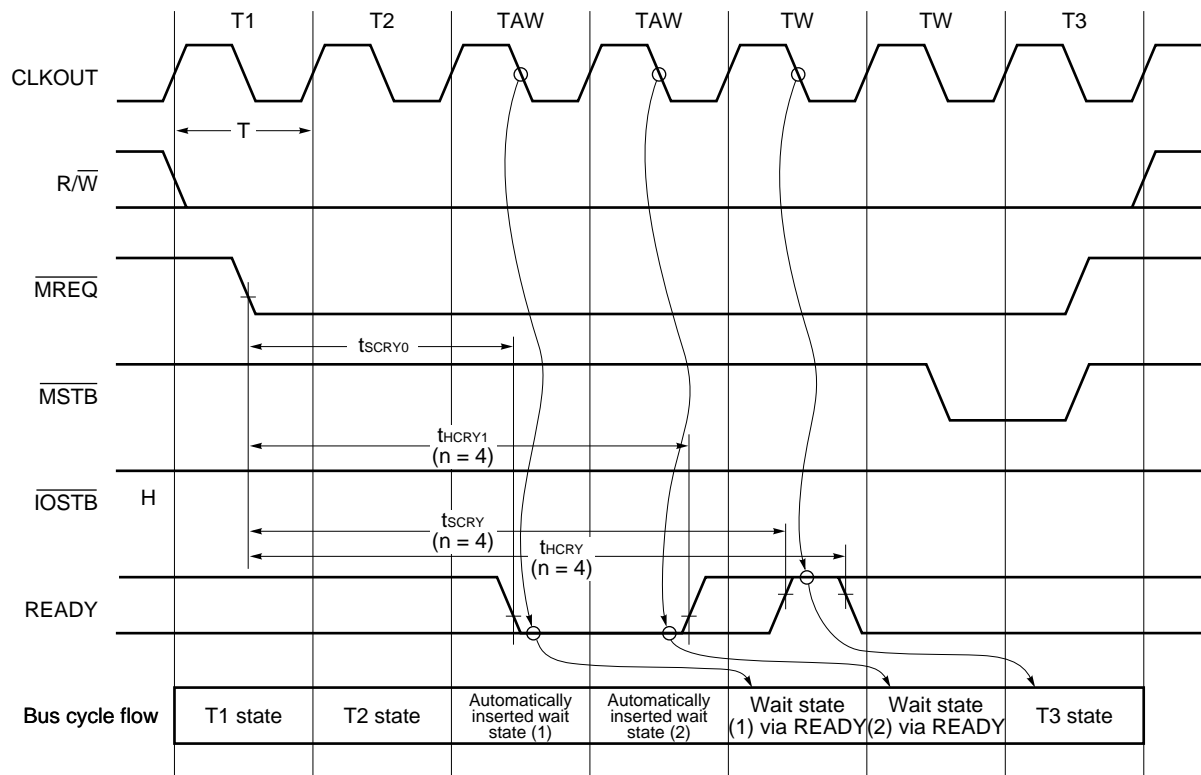
$$T = 125 \text{ ns}$$

$$t_{SCRY0} = 2T - 100 \text{ (Max.)} = 150 \text{ ns}$$

$$t_{HCRY0} = 2T \text{ (Min.)} = 250 \text{ ns}$$

Figure 5-4. Wait via READY Pin (V35) (4/8)

(d) Addition of two wait states to memory write cycle



When operation frequency is 8 MHz

$T = 125 \text{ ns}$

Assign 4 to n because the total number of wait states is four.

$t_{SCRY0} = 2T - 100 \text{ (Max.)} = 150 \text{ ns}$

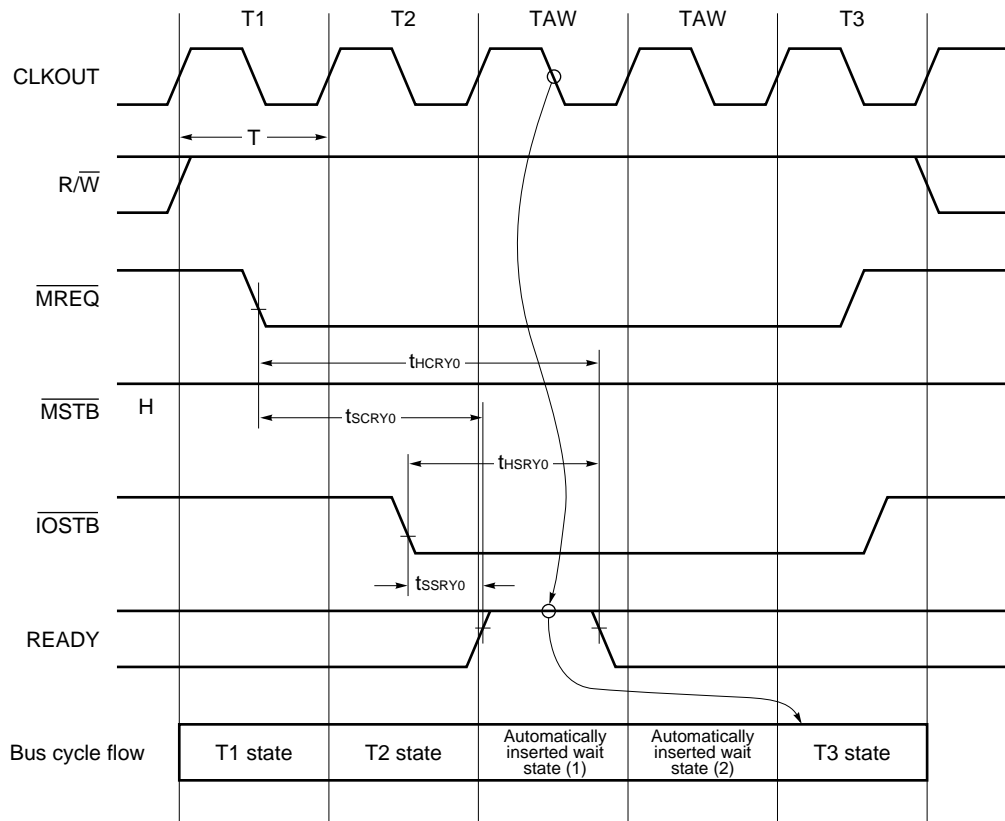
$t_{HCRY1} = (n - 1) T \text{ (Min.)} = 375 \text{ ns}$

$t_{SCRY} = nT - 100 \text{ (Max.)} = 400 \text{ ns}$

$t_{HCRY} = nT \text{ (Min.)} = 500 \text{ ns}$

Figure 5-4. Wait via READY Pin (V35) (5/8)

(e) No addition of wait state to I/O read cycle



When operation frequency is 8 MHz

$$T = 125 \text{ ns}$$

$$t_{SCRY0} = 2T - 100 \text{ (Max.)} = 150 \text{ ns}$$

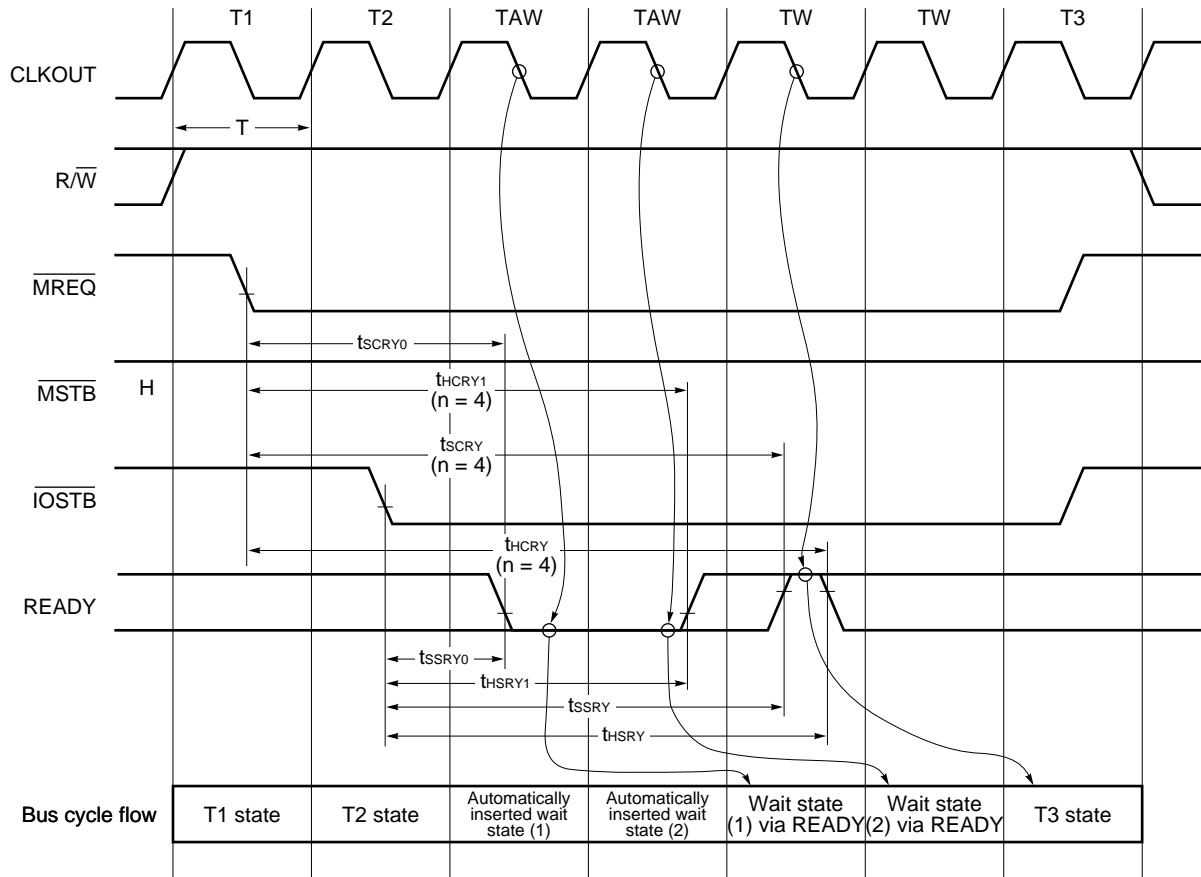
$$t_{HCRY0} = 2T \text{ (Min.)} = 250 \text{ ns}$$

$$t_{SSRY0} = T - 100 \text{ (Max.)} = 25 \text{ ns}$$

$$t_{HSRY0} = T \text{ (Min.)} = 125 \text{ ns}$$

Figure 5-4. Wait via READY Pin (V35) (6/8)

(f) Addition of two wait states to I/O read cycle



When operation frequency is 8 MHz

$$T = 125 \text{ ns}$$

Assign 4 to n because the total number of wait states is four.

$$t_{SCRY0} = 2T - 100 \text{ (Max.)} = 150 \text{ ns}$$

$$t_{HCRY1} = (n - 1) T \text{ (Min.)} = 375 \text{ ns}$$

$$t_{SCRY} = nT - 100 \text{ (Max.)} = 400 \text{ ns}$$

$$t_{HCRY} = nT \text{ (Min.)} = 500 \text{ ns}$$

$$t_{SSRY0} = T - 100 \text{ (Max.)} = 25 \text{ ns}$$

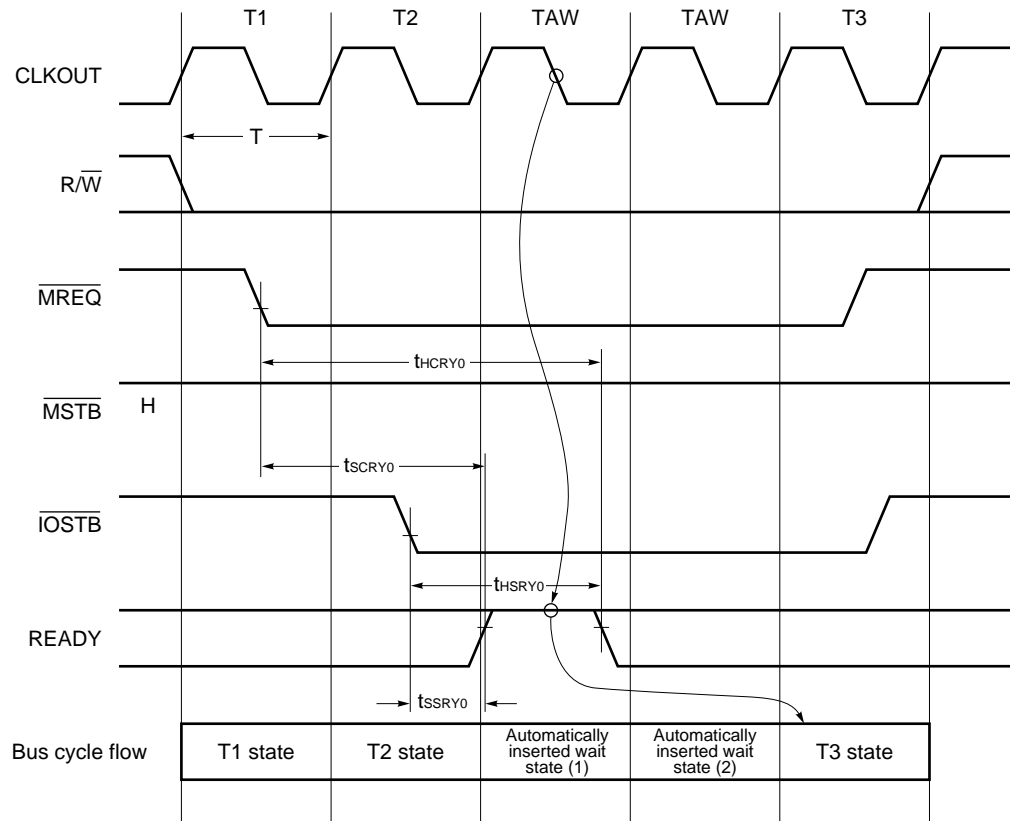
$$t_{HSRY1} = (n - 2) T \text{ (Min.)} = 250 \text{ ns}$$

$$t_{SSRY} = (n - 1) T - 100 \text{ (Max.)} = 275 \text{ ns}$$

$$t_{HSRY} = (n - 1) T \text{ (Min.)} = 375 \text{ ns}$$

Figure 5-4. Wait via READY Pin (V35) (7/8)

(g) No addition of wait state to I/O write cycle



When operation frequency is 8 MHz

$$T = 125 \text{ ns}$$

$$t_{SCRY0} = 2T - 100 \text{ (Max.)} = 150 \text{ ns}$$

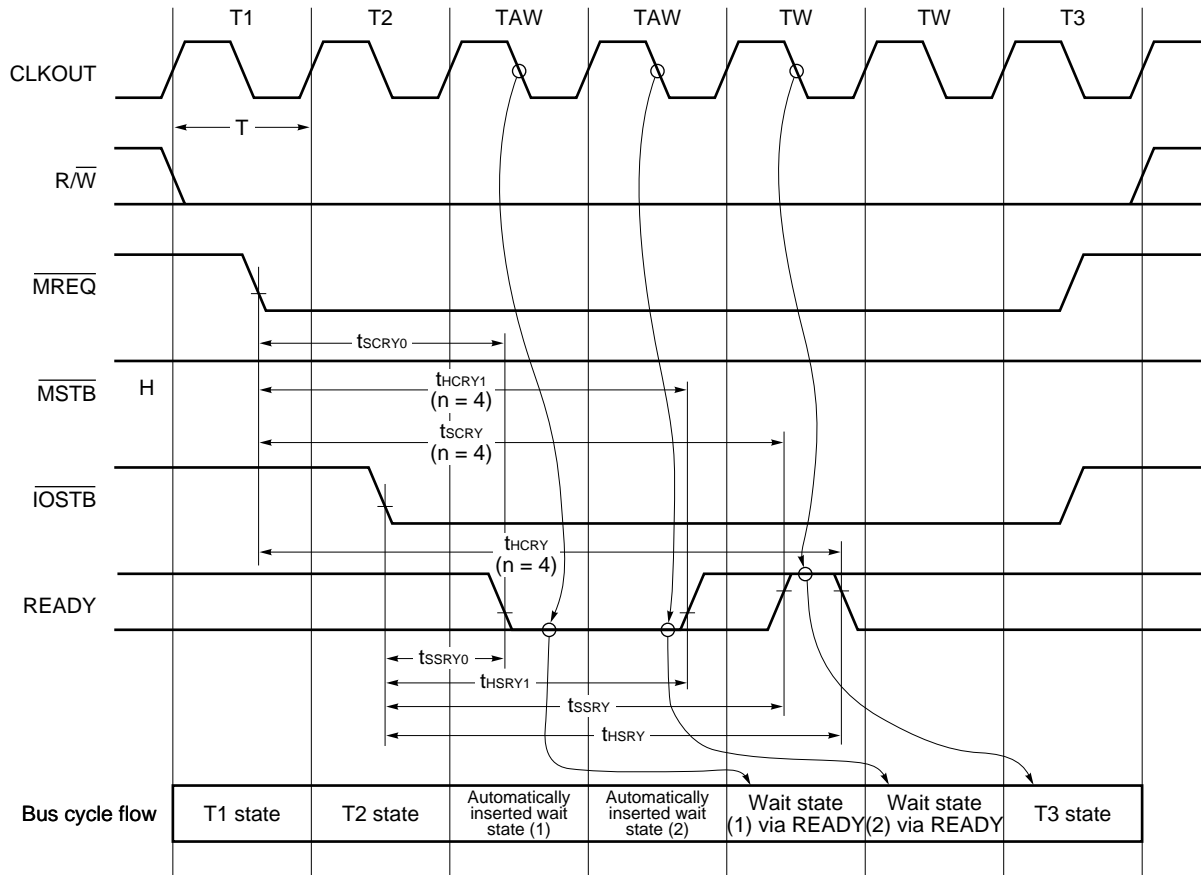
$$t_{HCRY0} = 2T \text{ (Min.)} = 250 \text{ ns}$$

$$t_{SSRY0} = T - 100 \text{ (Max.)} = 25 \text{ ns}$$

$$t_{HSRY0} = T \text{ (Min.)} = 125 \text{ ns}$$

Figure 5-4. Wait via READY Pin (V35) (8/8)

(h) Addition of two wait states to I/O write cycle



When operation frequency is 8 MHz

$T = 125 \text{ ns}$

Assign 4 to n because the total number of wait states is four.

$t_{SCRY0} = 2T - 100 \text{ (Max.)} = 150 \text{ ns}$

$t_{HCRY1} = (n - 1) T \text{ (Min.)} = 375 \text{ ns}$

$t_{SCRY} = nT - 100 \text{ (Max.)} = 400 \text{ ns}$

$t_{HCRY} = nT \text{ (Min.)} = 500 \text{ ns}$

$t_{SSRY0} = T - 100 \text{ (Max.)} = 25 \text{ ns}$

$t_{HSRY1} = (n - 2) T \text{ (Min.)} = 250 \text{ ns}$

$t_{SSRY} = (n - 1) T - 100 \text{ (Max.)} = 275 \text{ ns}$

$t_{HSRY} = (n - 1) T \text{ (Min.)} = 375 \text{ ns}$

5.2 Bus Hold Function

The V25 and V35 each have a bus hold function. High pulse input from an external device to the HLDRQ pin indicates that an external device has used the bus. When detecting that the HLDRQ pin is high, the V25 or V35 sets high-impedance output from A0 to A19, D0 to D7 (D0 to D15), $\overline{\text{REFRQ}}$, $\overline{\text{MREQ}}$, $\overline{\text{MSTB}}$, $\overline{\text{IOSTB}}$, and $\overline{\text{R/W}}$ and sets the $\overline{\text{HLDAK}}$ pin low to inform the external device that the buses are released. Then the V25 or V35 enters the hold mode. During the hold mode, operations such as instruction execution and prefetch interrupt acknowledgment are stopped and only the on-chip peripheral hardware which does not use the buses is operated. During the hold mode, the V25 or V35 checks the HLDRQ pin and sets the $\overline{\text{HLDAK}}$ signal high when HLDRQ is detected as low to inform the external device that the buses are not released. After one clock, the V25 or V35 restarts execution of instructions.

A bus hold request can also be acknowledged during the HALT mode (see section 12.2). When the hold mode is released (if the HLDRQ signal is low), a return is made to the HALT mode.

During execution of a block transfer instruction that has a repeat prefix added, a bus hold request can be acknowledged after each bus cycle.

Bus hold requests are not acknowledged when one instruction following BUSLOCK ^{Note} prefix is being executed or when an interrupt acknowledgment operation is being performed.

In the hold mode, the V25 or V35 can insert a memory refresh cycle by setting refresh mode (RFM) register HLDRF (bit 6). At every refresh timing, the V25 or V35 forcibly sets the $\overline{\text{HLDAK}}$ signal high and checks that HLDRQ goes low, then executes a refresh cycle. After this, if the HLDRQ signal goes high, the V25 or V35 again enters the hold mode. If the HLDRQ signal remains low, the hold mode is released and the V25 or V35 restarts execution of instructions.

The HLDRQ pin is also used for P27, and the $\overline{\text{HLDAK}}$ pin is also used for P26. To use the bus hold function, set port 2 mode control register (PMC2) bits 6 and 7 to 1.

Note BUSLOCK
 REP
 MOVBK

No bus hold requests are acknowledged during block servicing instruction execution in such a program.

5.2.1 Response time from HLDRQ to $\overline{\text{HLDAK}}$ (unit: clock cycles)

The response time from HLDRQ to $\overline{\text{HLDAK}}$ is shown below. However, the following cases are exceptions to this.

- When an interrupt acknowledge cycle is generated by an external interrupt controller
- When a BUSLOCK instruction is executed
- When in STOP mode

	MIN.	MAX.
V25	3	7 + 2W
V35	3	6 + W

W: Number of wait states

5.3 Refresh Functions

The V25 and V35 each have functions for refreshing DRAM and pseudo SRAM. These functions include:

- Periodical refresh cycle insertion function in a series of bus cycles
- Refresh address and refresh pulse output function to refresh DRAM and pseudo SRAM
- Pseudo SRAM power-down self-refresh mode support function
- Refresh cycle generation function during hold mode or HALT mode
- Wait state insertion function in a refresh cycle

5.3.1 Refresh mode register (RFM)

The refresh mode register (RFM) is an 8-bit register that controls the refresh function. The register can be written or read by making an 8-bit or 1-bit memory access.

When $\overline{\text{RESET}}$ is asserted, the RFM register contents are initialized to FCH.

The RFM register format is shown below. The bit functions are described as follows.

7	6	5	4	3	2	1	0
RFLV	HLDRF	HLTRF	RFEN	RFW1	RFW0	RFT1	RFT0

RFT0 and **RFT1**: Refresh cycle specification bits

Refresh cycle can be selected out of output taps 3 to 6 of the time base counter (see **CHAPTER 10**). A refresh cycle is generated at intervals listed in Table 5-5.

Table 5-5. Refresh Cycles

When $f_{\text{CLK}} = 8 \text{ MHz}$

RFT 1	RFT 0	Refresh cycles
0	0	$2^4/f_{\text{CLK}}$ (2.0 μs)
0	1	$2^5/f_{\text{CLK}}$ (4.0 μs)
1	0	$2^6/f_{\text{CLK}}$ (8.0 μs)
1	1	$2^7/f_{\text{CLK}}$ (16.0 μs)

RFW0 and **RFW1**: Bits specifying the number of wait states to be inserted in a refresh cycle

The number of wait states inserted in a refresh cycle is specified by setting RFW0 and RFW1 as shown in Table 5-6 rather than the programmable wait function described in section 5.1 above.

Table 5-6. Number of Wait States Inserted in Refresh Cycle

RFW	RFW	Wait states
1	0	
0	0	0 states
0	1	1 state
1	0	2 states
1	1	2 states

RFEN: Bit to enable automatic refresh cycle insertion

When this bit is set to 1, automatic refresh cycle insertion is enabled. When set to 0, automatic refresh cycle insertion is disabled. $\overline{\text{REFRQ}}$ pin output is controlled by the RFLV bit contents (for details, see the **RFLV bit description** below).

HLTRF: Bit to enable automatic refresh cycle insertion during the HALT mode

When this bit is set to 1, automatic refresh cycle insertion during HALT mode is enabled. When set to 0, it is disabled. However, when the RFEN bit is set to 0, the automatic refresh cycle insertion is disabled regardless of the HLTRF bit contents.

HLDRF: Bit to enable automatic refresh cycle insertion during the hold mode

When this bit set is to 1, automatic refresh cycle insertion during hold mode is enabled. When set to 0, it is disabled.

When it is enabled, $\overline{\text{HLDK}}$ output is forcibly set high at every refresh timing and a refresh cycle is automatically inserted.

RFLV: $\overline{\text{REFRQ}}$ signal output level specification bit

Figure 5-5 shows the control circuit that depends upon the RFLV bit contents. Output is determined according to the logic listed in Table 5-7. When the RFLV bit is read, it becomes the master RFLV output. When this bit is written, it is written into slave RFLV. Master RFLV is written into when the refresh timing is generated.

The pseudo SRAM power-down self-refresh mode can be supported by using the RFLV bit.

Figure 5-5. Control Circuit Depending on the RFLV Bit Contents

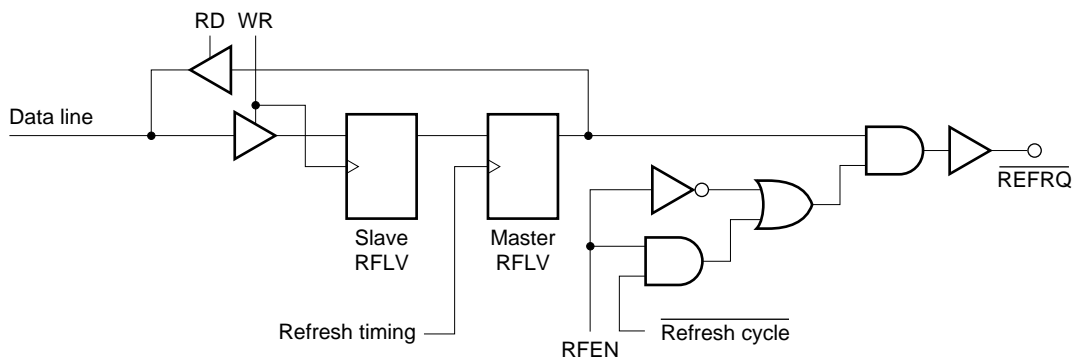


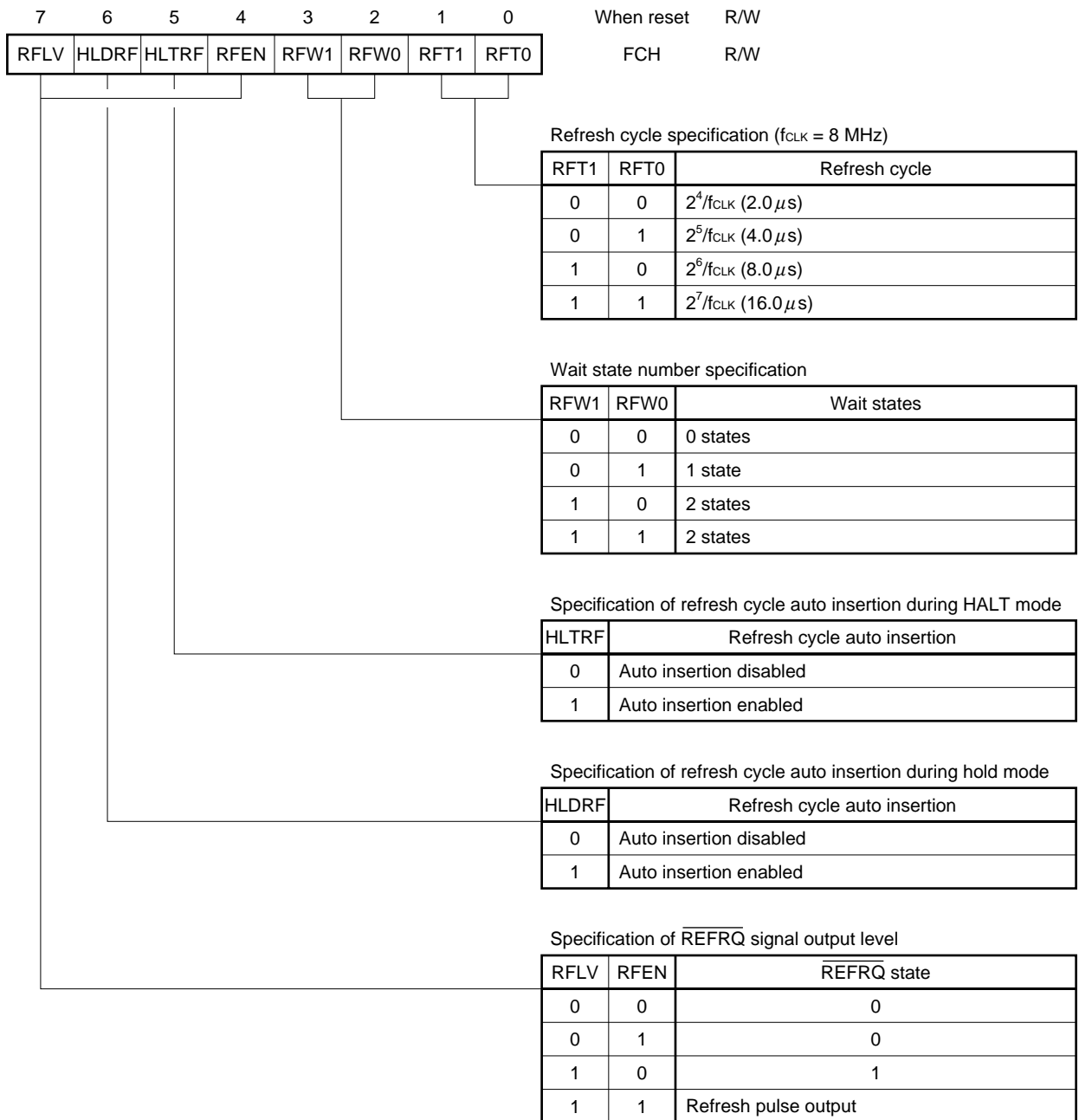
Table 5-7. $\overline{\text{REFRQ}}$ Signal Output Levels

RFLV	RFEN	$\overline{\text{REFRQ}}$ state
0	0	0
0	1	0
1	0	1
1	1	Refresh pulse output

A refresh cycle is inserted at the refresh timing when the RFEN bit is set to 1. At that time, $\overline{\text{MREQ}}$, $\overline{\text{MSTB}}$, and $\overline{\text{IOSTB}}$ go high, a refresh address is output to A0 to A8, low level is output to A9 to A19 (for the V25; see **Table 5-3** for the V35), and a refresh pulse is output from the $\overline{\text{REFRQ}}$ pin.

Even if the RFLV bit is written, it does not become read data until the next refresh timing.

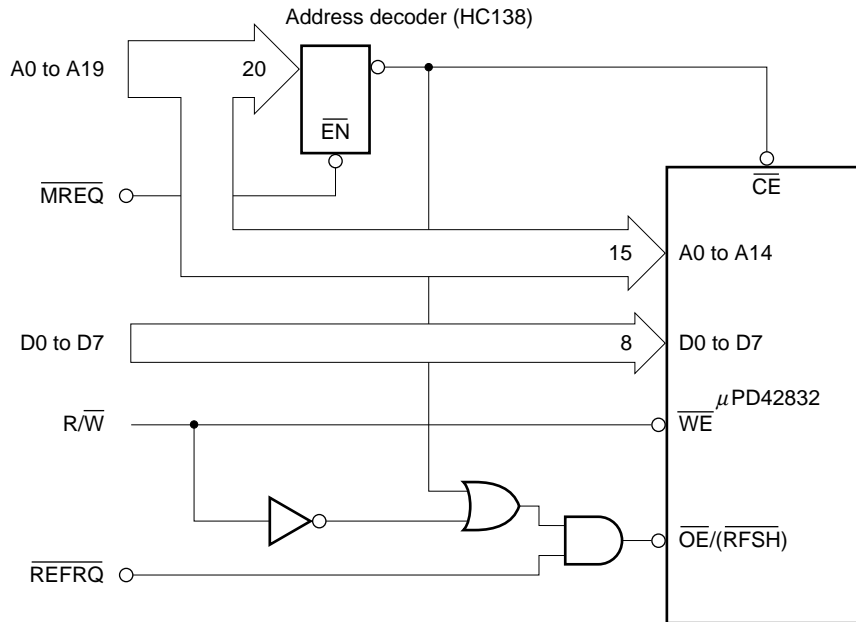
Figure 5-6. RFM



5.3.2 Connection to pseudo SRAM

Figure 5-7 shows a circuit example connecting pseudo SRAM equivalent to the μ PD42832.

Figure 5-7. μ PD42832 Connection Circuit Example



In this connection example, two modes can be used: pulse refresh mode and power-down self-refresh mode. In the pulse refresh mode, pulses are given to the $\overline{OE}/(\overline{RFSH})$ pin from the REFRQ pin when the \overline{CE} pin is high. In the power-down self-refresh mode, the REFRQ pin is set low by software which resets bit 7 (RFLV) to 0 in the refresh mode register (RFM).

The power-down self-refresh mode is used when the CPU is in the standby (STOP) mode and cannot perform pulse refresh operations. Thus, the RFLV bit is reset to 0 just before the CPU enters the standby mode. When the CPU is restored from the standby mode, the RFLV bit is set to 1 so that pulse refresh operations can be performed.

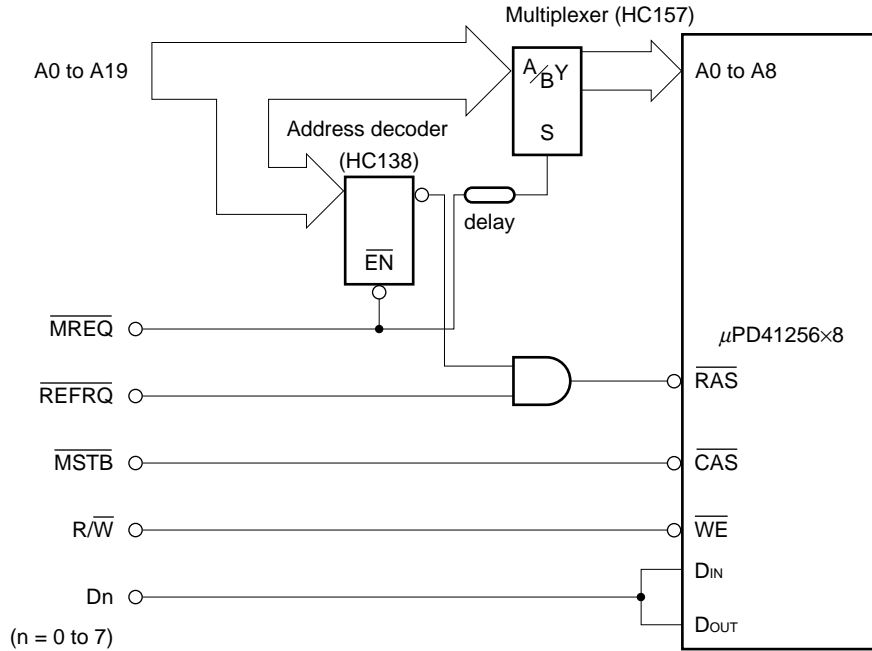
Caution When reset (\overline{RESET} pin = 0), the REFRQ pin goes to high-impedance.

5.3.3 Connection to DRAM

Figure 5-8 shows a circuit examples connecting the μ PD41256 (256 Kbytes \times 1-bit structure).

★

Figure 5-8. μ PD41256 Connection Circuit Example (when 5 MHz)



In this connection example, refresh operations are performed by \overline{RAS} only refresh using the 9-bit refresh address output to the address bus in synchronization with a pulse output from the \overline{REFREQ} pin.

5.4 Bus Mastership

The bus mastership priority levels for the V25 and V35 follow the order shown below.

(1) Refresh cycle (see section 5.3)

Whenever refresh cycle insertion is enabled, a refresh cycle is generated. However, during the hold mode, the $\overline{\text{HLDAK}}$ signal is forcibly set high and a wait is made for the HLDRQ signal to go low before a refresh cycle is executed.

No refresh cycle is started while wait cycles are inserted by the READY pin.

(2) Hold mode (see section 5.2)

The transition to the hold mode is made except during execution of one instruction following a BUSLOCK prefix or interrupt acknowledge cycle.

(3) DMA cycle (see Chapter 6)

(4) Other bus cycles

However, when an $\overline{\text{INTAK}}$ cycle is being executed, refresh cycles, hold modes, and DMA cycles are temporarily held pending. DMA cycles are also held pending during operations of interrupt acknowledgment for internal interrupts. See section 4.16 **Hardware Interrupt Response Time** for description of time required for interrupt acknowledgment.

In the STOP mode, the buses do not operate. (See **Table 12-2** for the bus state).

Data transfer (block transfer, DMA transfer, macro service, etc.) can be used at the same time with the same program, but simultaneous execution is not possible because the data transfer uses a single bus.

5.5 Bus Timings

Figures 5-9 to 5-30 show the main bus timings (except for DMA).

When no bus access is made, the control pins are deactivated and both data bus output and address bus output are undefined.

5.5.1 Bus timing of V25

Figure 5-9. Memory Read Cycle

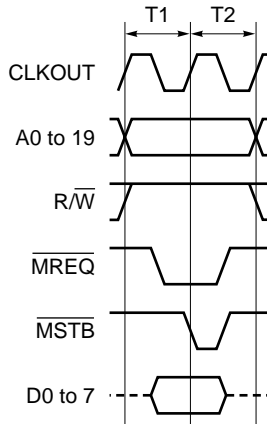


Figure 5-10. Memory Write Cycle

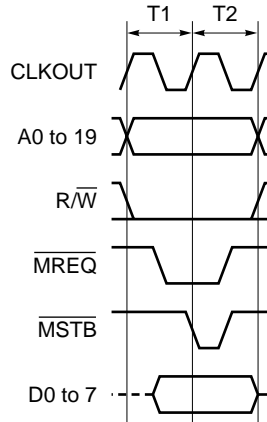


Figure 5-11. I/O Read Cycle

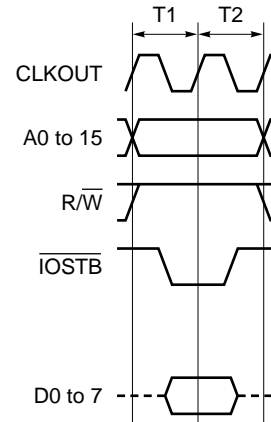


Figure 5-12. I/O Write Cycle

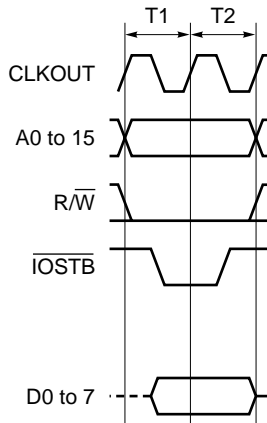


Figure 5-13. Memory Read Cycle (when one wait state is inserted)

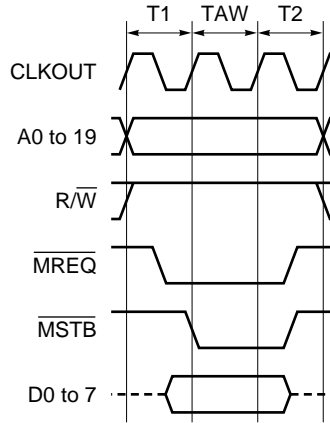
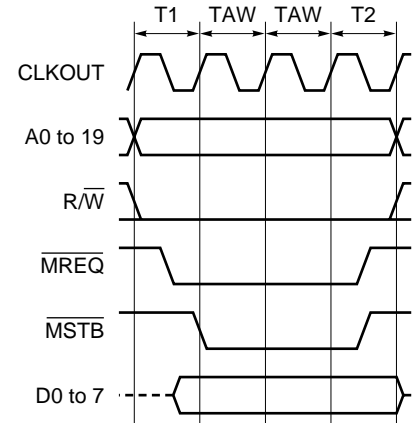
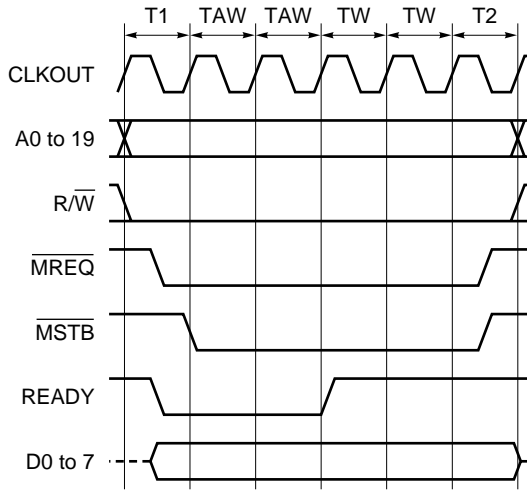


Figure 5-14. Memory Write Cycle (when two wait states are inserted)



Remark The broken line indicates high-impedance.

Figure 5-15. Memory Write Cycle (during READY pin operation)



Remark The broken line indicates high-impedance.

Figure 5-16. Refresh Cycle (when one wait state is inserted)

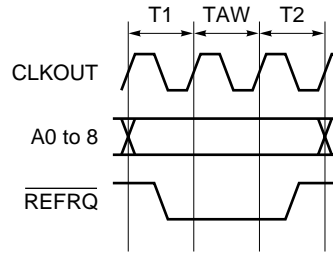
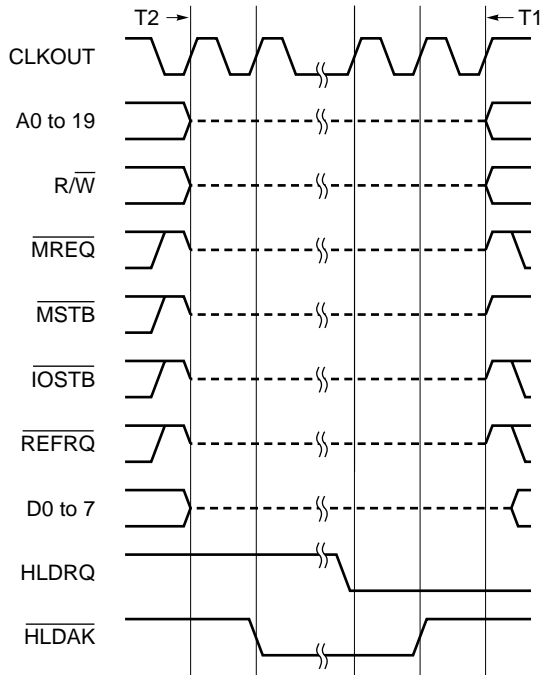
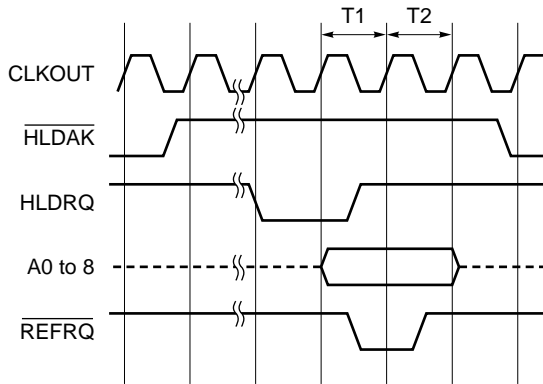


Figure 5-17. Bus Hold Acknowledgment Release Timing



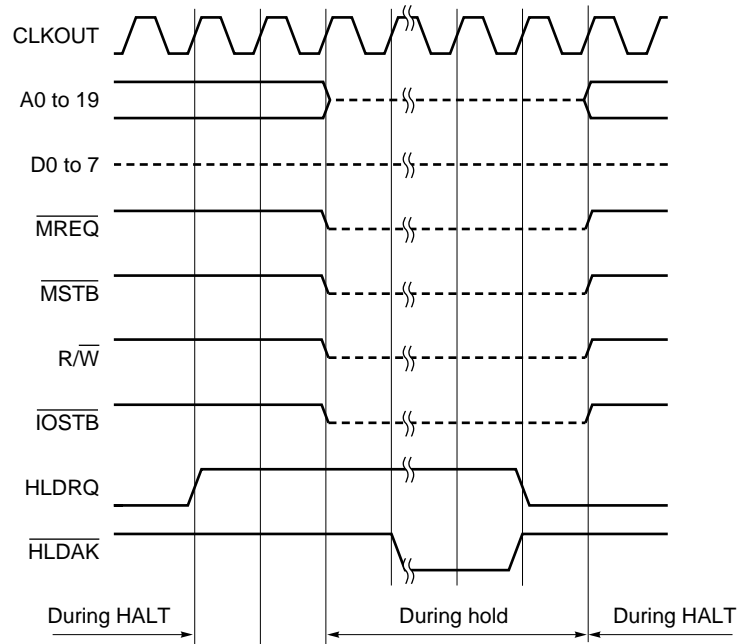
Remark The broken line indicates high-impedance.

Figure 5-18. Refresh Cycle in Hold Mode (0 wait state)



Remark The broken line indicates high-impedance.

Figure 5-19. Bus Hold Acknowledgment Release Timing during HALT Mode



Remark The broken line indicates high-impedance.

5.5.2 Bus timing of V35

Figure 5-20. Memory Read Cycle

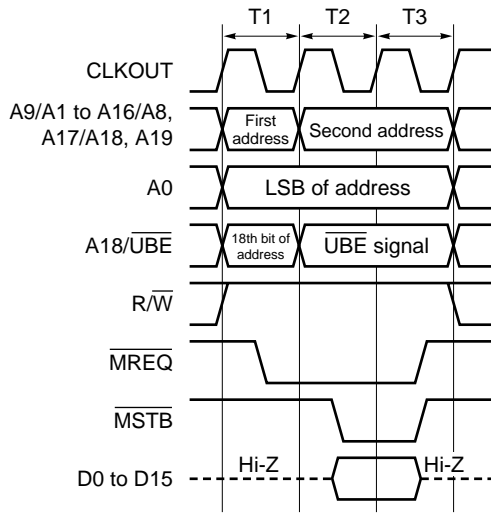


Figure 5-21. Memory Write Cycle

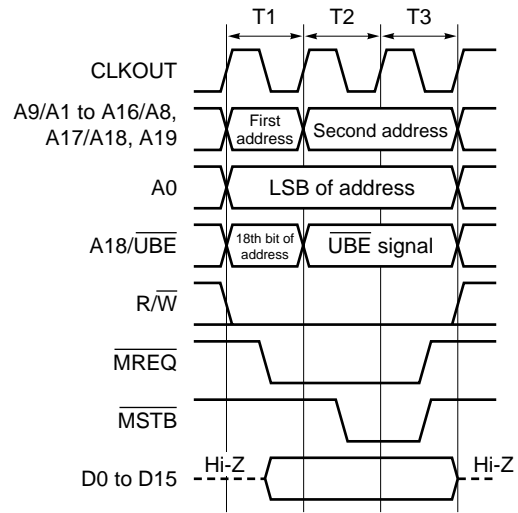


Figure 5-22. I/O Read Cycle

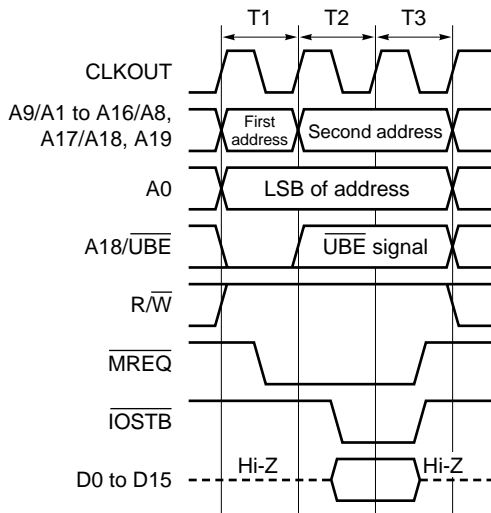


Figure 5-23. I/O Write Cycle

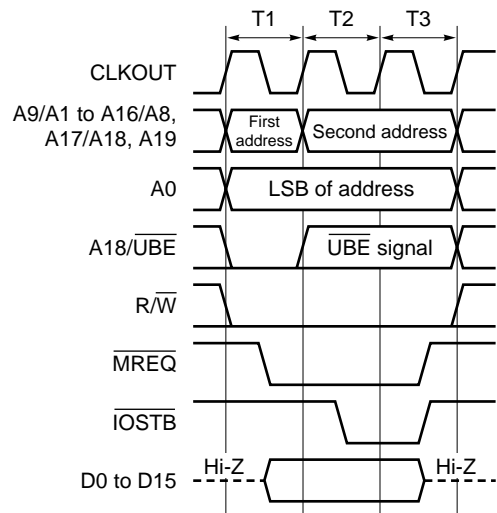


Figure 5-24. Memory Read Cycle (when one wait state is inserted)

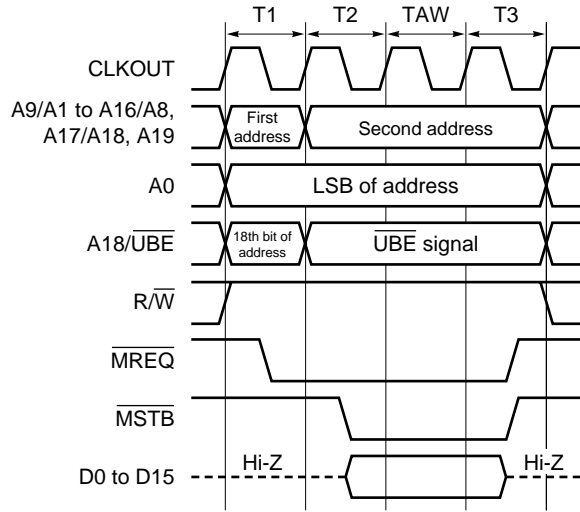


Figure 5-25. Memory Write Cycle (when two wait states are inserted)

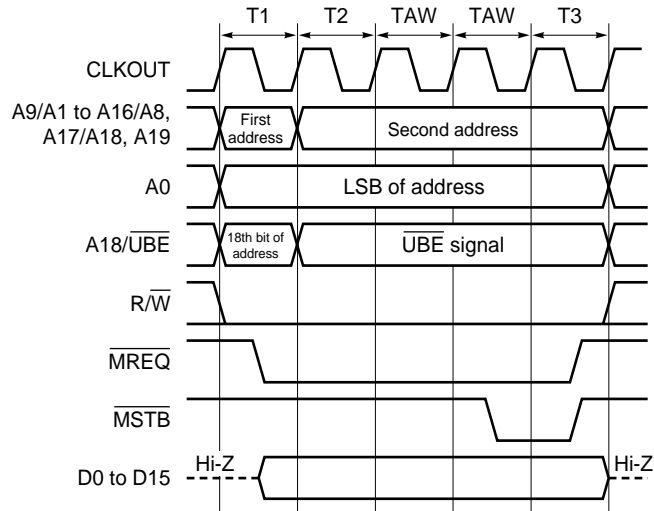


Figure 5-26. Memory Write Cycle (during READY pin operation)

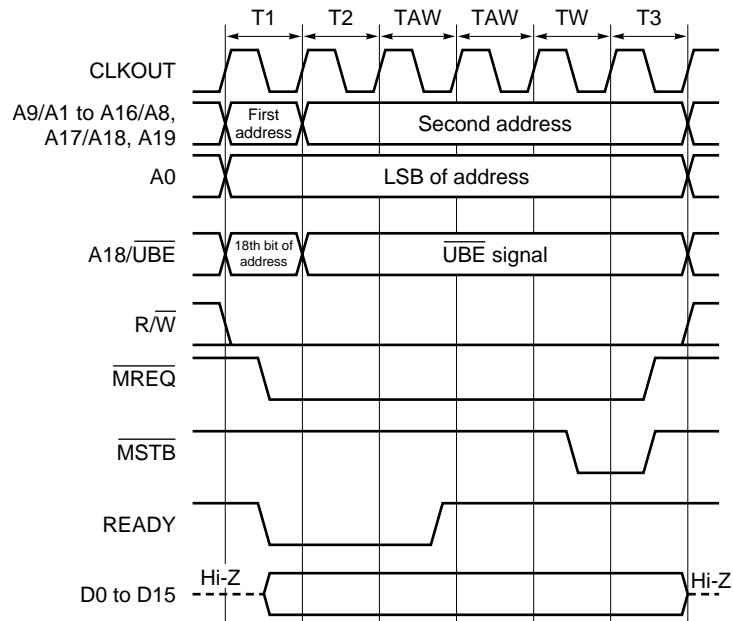


Figure 5-27. Refresh Cycle (when one wait state is inserted)

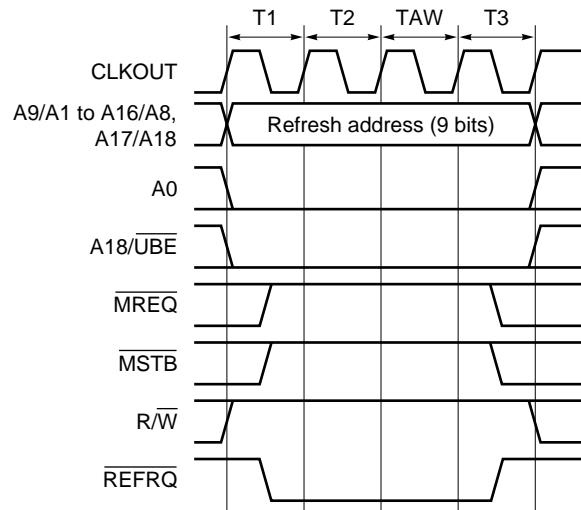


Figure 5-28. Bus Hold Acknowledgment Release Timing

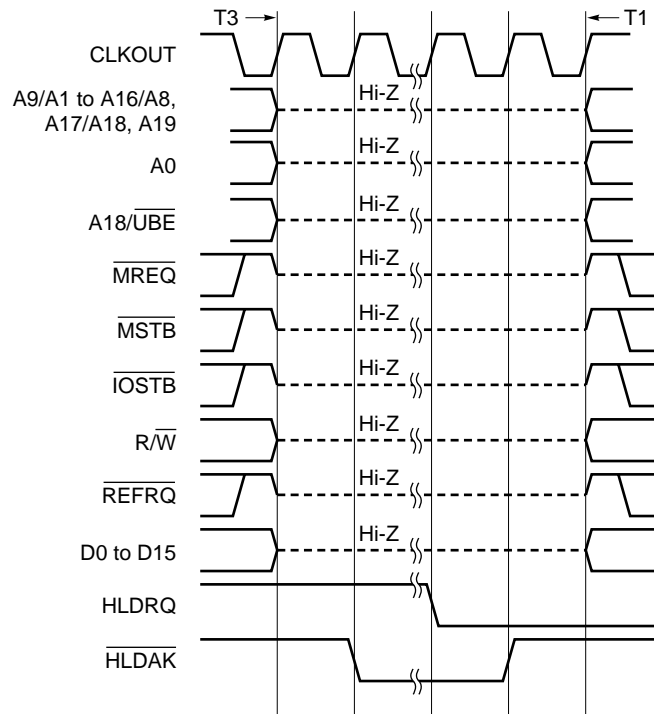


Figure 5-29. Refresh Cycle in Hold Mode

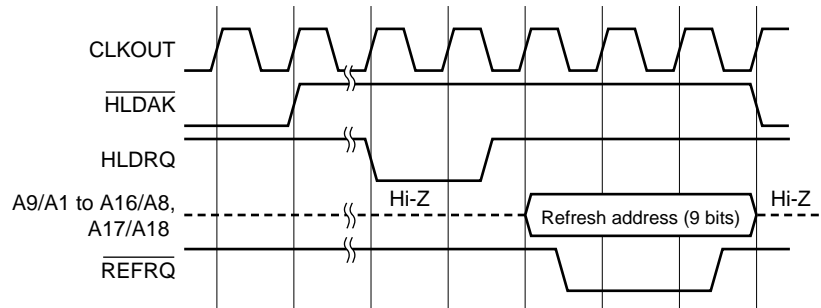
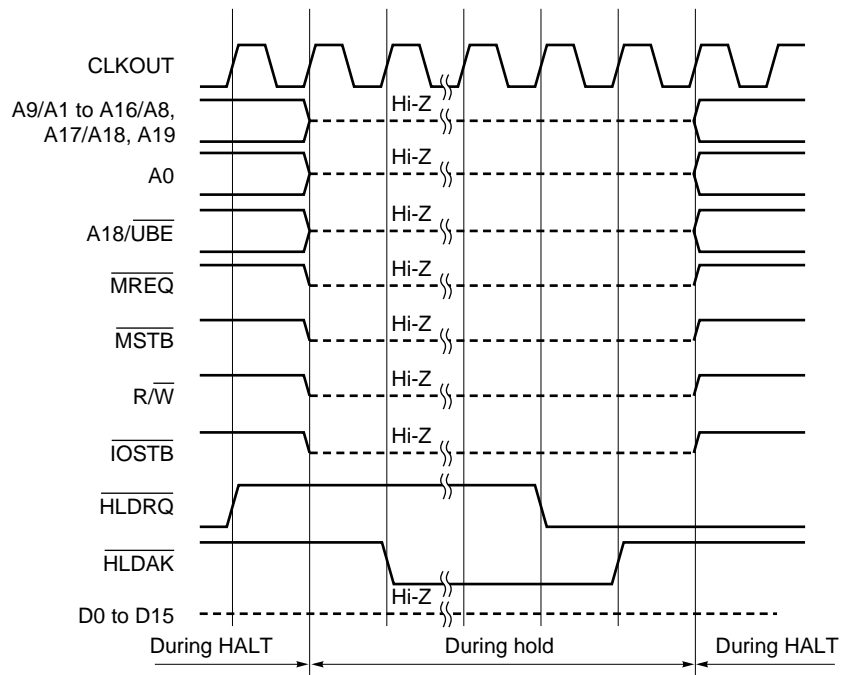


Figure 5-30. Bus Hold Acknowledgment Release Timing during HALT Mode



CHAPTER 6 DMA CONTROLLER

The V25 and V35 each contain a two-channel DMA controller to enable direct addressing of the 1-Mbyte memory space.

6.1 Pin Functions

The following pins are provided for the DMA controller. Because all of these pins are also used as ports, the corresponding bits of the port 2 mode control register (PMC2) must be set to 1 to use these pins for the DMA controller.

(1) DMARQ0 and DMARQ1 (P20 and P23)

These are active-high DMA request input pins.

(2) $\overline{\text{DMAAK0}}$ and $\overline{\text{DMAAK1}}$ (P21 and P24)

These are active-low DMA response output pins.

However, the signals are not output during DMA transfer between memory and memory (burst mode or single-step mode).

(3) $\overline{\text{TC0}}$ and $\overline{\text{TC1}}$ (P22 and P25)

These are active-low DMA completion output pins.

Output occurs when TC0 or TC1 in the DMA service channel is set to 0.

6.2 DMA Operation

The V25 and V35 each have four DMA transfer modes. Table 6-1 lists the transfer mode functions, etc.

- ★ In DMA transfer, the internal data area cannot be accessed. When any address corresponding to the internal data area is accessed, the external memory location at the same address as the internal data area is accessed.

Table 6-1. Transfer Mode Functions

Mode name	Transfer between	Function	DMA start	Stop method	Interrupt	During HALT	DMA request during DMA operation
Single step	Memory and memory	When one DMA request occurs, executions of one instruction and one DMA transfer are repeated alternately as many times as specified.	<ul style="list-style-type: none"> On the DMARQ rising edge When the DMA control register's TDMA bit is set 	By software (clear EDMA bit)	All are acknowledged	DMA transfer is executed consecutively as many times as specified.	Channel 1 DMA is pending or stopped and channel 0 DMA is executed.
Burst	Memory and memory	When one DMA request occurs, DMA transfer is executed consecutively as many times as specified.	<ul style="list-style-type: none"> On the DMARQ rising edge When DMA control register TDMA bit is set 	None	<ul style="list-style-type: none"> Not acknowledged during DMA transfer 	DMA transfer is executed consecutively as many times specified.	All other DMA is held pending until DMA transfer terminates.
One transfer	Memory and I/O	Each time a DMA request occurs, one DMA transfer is executed.	<ul style="list-style-type: none"> On the DMARQ rising edge 	By software (clear EDMA bit)	All are acknowledged	As usual	After one DMA transfer, requested DMA is executed.
Demand release	Memory and I/O	While the DMARQ pin remains high, DMA transfer is executed.	<ul style="list-style-type: none"> When DMARQ is set high 	<ul style="list-style-type: none"> By setting DMARQ low during DMA By software in other cases (clear EDMA bit) 	<ul style="list-style-type: none"> All except NMI are not acknowledged during DMA transfer All are acknowledged except during DMA transfer 	As usual	Channel 1 DMA is pending or stopped and channel 0 DMA is executed.

In DMA transfer between memory and memory, the $\overline{\text{DMAAK}}$ signal is not asserted. In DMA transfer between memory and I/O, the $\overline{\text{DMAAK}}$ signal is asserted for each DMA cycle. Use the $\overline{\text{DMAAK}}$ signal instead of the $\overline{\text{IOSTB}}$ signal to access I/O during DMA transfer.

The programmable wait function (see section 5.1) is effective even during DMA transfer. During DMA transfer between memory and memory, the specified number of wait states are inserted for each destination and source. During DMA transfer between memory and I/O, one transfer is completed with each bus cycle, thus the specified number of wait states are inserted for memory or I/O, whichever is slower.

The bus hold function and refresh function are also effective during DMA transfer, and DMA transfer is temporarily stopped when either of these functions is executed.

If DMA transfer is requested, it is also executed when a block servicing (transfer, comparison, or retrieval input/output) instruction with a repeat prefix is being executed. At that time, execution of the block servicing instruction is temporarily stopped.

Likewise, DMA transfer is also executed if a $\overline{\text{BUSLOCK}}$ prefix is added.

During DMA transfer, no interrupts are acknowledged and all are held pending.

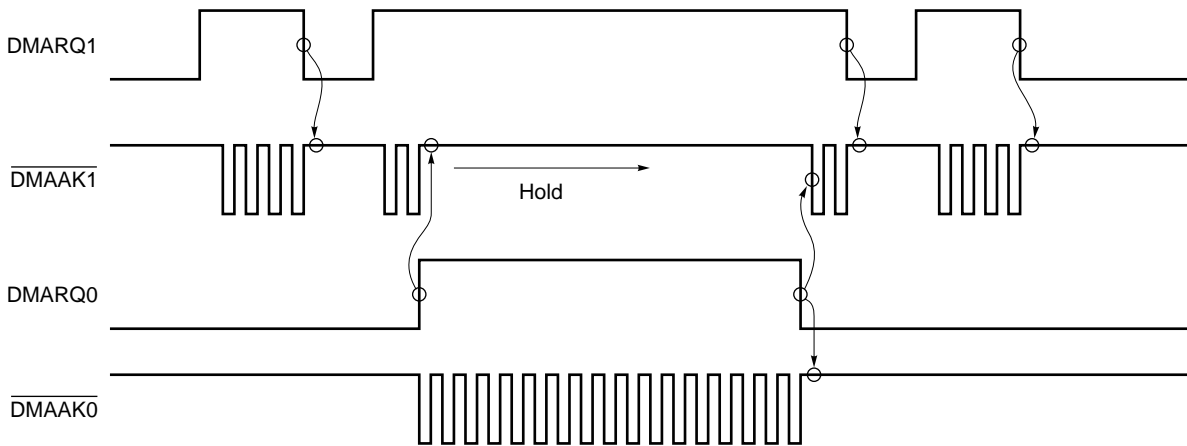
During the HALT mode, if DMA transfer is requested, it is executed. A return is made to the HALT mode when DMA transfer terminates. If a DMA transfer completion interrupt occurs after a return is made to the HALT mode, the HALT mode is released.

If DMA requests occur at the same time, channel 0 takes precedence over channel 1.

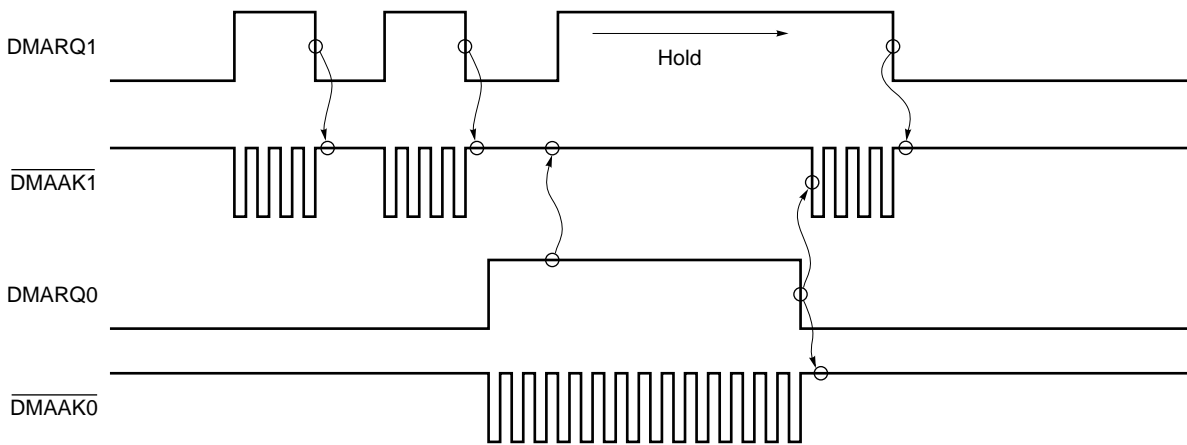
When DMA transfer terminates, the following operations occur.

- Output to the TC pin goes low.
- The EDMA bit is cleared (reset to 0).
- A DMA completion interrupt request occurs.

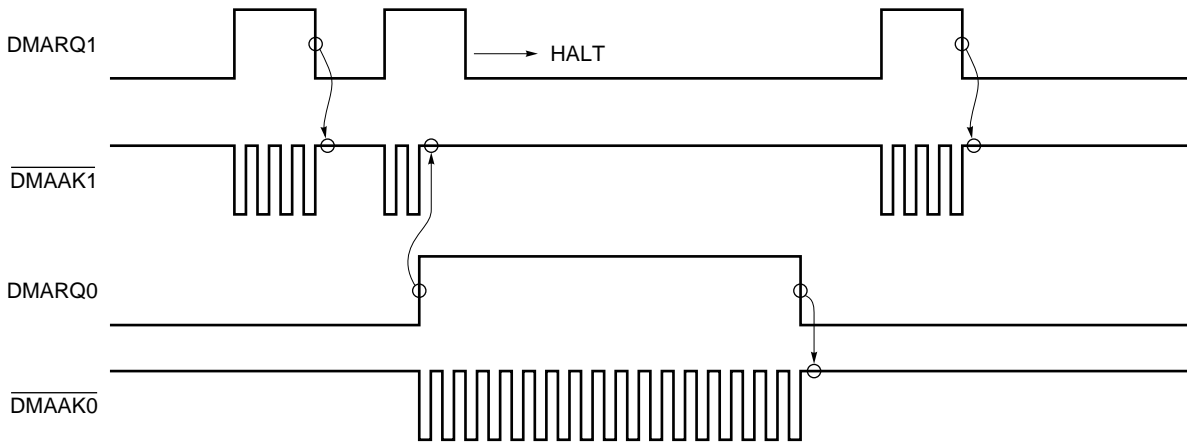
Example 1. When in demand release mode: held pending by DMA request



Example 2. When in demand release mode: held pending by DMA request



Example. When in demand release mode: stopped by DMA request



6.3 DMA Control Registers

DMA mode registers and DMA control registers are provided for DMA transfer mode specification, etc. Also, DMA service channels that specify the transfer destination, transfer source, and number of transfers are assigned to the on-chip RAM area. DMA interrupt request control registers are also provided. These registers are provided for each channel.

6.3.1 DMA mode registers (DMAM0 and DMAM1)

The DMA mode registers are 8-bit registers that specify the DMA transfer mode, etc. The registers can be written/read by making an 8-bit or 1-bit memory access. DMAM0 and DMAM1 correspond to channels 0 and 1, respectively.

When RESET is asserted, the DMAMn register contents are initialized to 00H.

7	6	5	4	3	2	1	0
MD2	MD1	MD0	W	EDMA	TDMA	0	0

MD2, **MD1**, and **MD0** : Transfer mode specification bits

MD2	MD1	MD0	Transfer mode
0	0	0	Single-step mode
0	0	1	Demand release mode (I/O to memory)
0	1	0	Demand release mode (memory to I/O)
0	1	1	Setting prohibited
1	0	0	Burst mode
1	0	1	One transfer mode (I/O to memory)
1	1	0	One transfer mode (memory to I/O)
1	1	1	Setting prohibited

W : Bit specifying whether transfer processing is performed in byte or word units

When this bit is set to 0, byte transfer is specified; when set to 1, word transfer is specified.

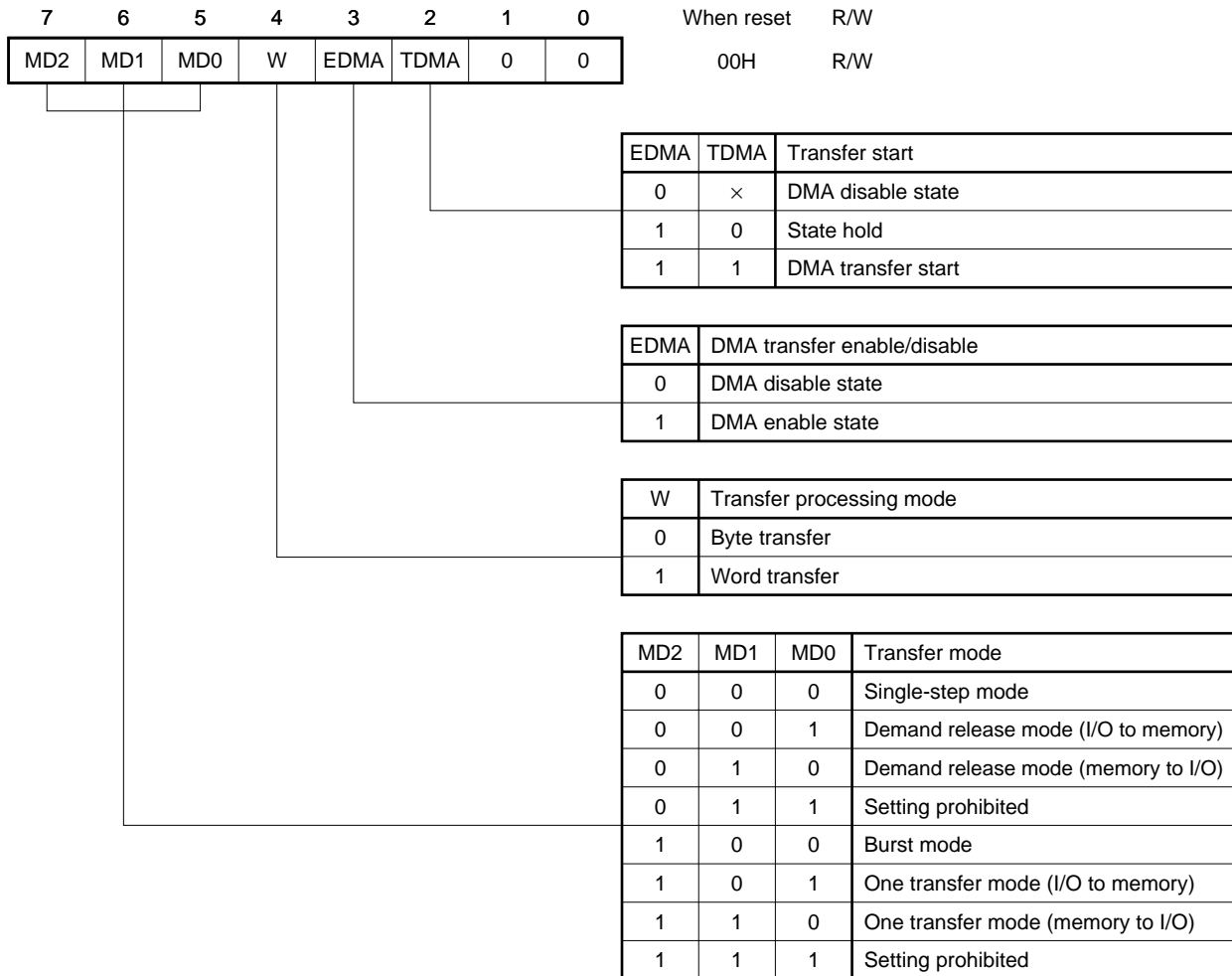
EDMA : DMA transfer enable/disable specification bit

Set this bit to 1 to enable DMA transfer and to 0 to disable DMA transfer. When set to 0 (disable), DMA requests are ignored and are not held pending. When the DMA service channel terminal counter (TC) is set to 0H, this bit is automatically cleared (reset to 0).

TDMA : Transfer start bit

This bit is valid only in the single-step or burst mode. DMA transfer is started by writing 1 for this bit, as long as the EDMA bit is set to 1. The bit read level is always 0. This bit has no meaning in the demand release mode or one transfer mode.

Figure 6-1. DMAM0 and DMAM1



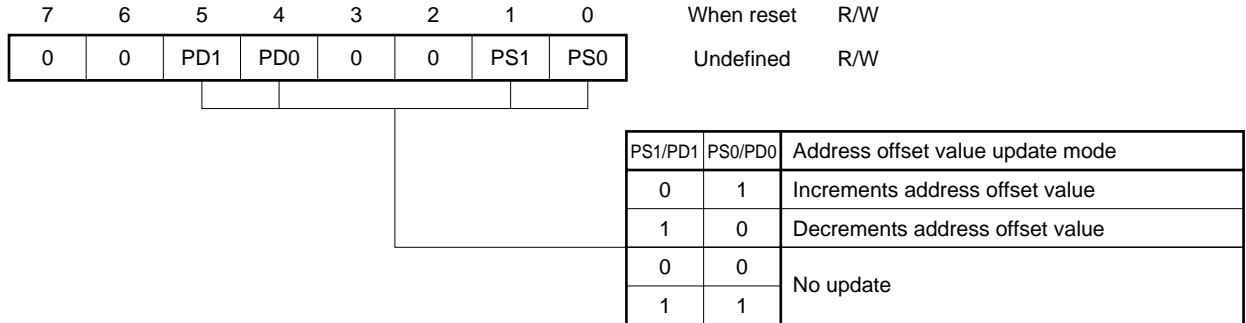
6.3.2 DMA control registers (DMAC0 and DMAC1)

The DMA control registers are 8-bit registers that specify the source address and destination address update modes. The registers can be written/read by making an 8-bit or 1-bit memory access.

The DMACn register contents are maintained even when $\overline{\text{RESET}}$ is asserted.

As shown in Figure 6-2, the source address offset value update mode is specified by setting DMACn register bits 1 and 0 (PS1 and PS0) and the destination address offset value update mode is specified by setting bits 5 and 4 (PD1 and PD0).

Figure 6-2. DMAC0 and DMAC1



6.3.3 DMA interrupt request control registers (DIC0 and DIC1)

The DMA interrupt request control registers are 8-bit registers that control interrupts which occur upon completion of a DMA transfer. The interrupt occurs when the terminal counter (TC) is set to 0H.

The registers can be written/read by making an 8-bit or 1-bit memory access, in which case one wait state is inserted.

When $\overline{\text{RESET}}$ is asserted, the DICn register contents are initialized to 47H.

These interrupts do not support the macro service function. Channel 0 (INTD0) and channel 1 (INTD1) DMA transfer completion interrupts make up one group and channel 0 is specified a higher interrupt priority level than channel 1. INTD0 is controlled by setting the DIC0 register and the vector becomes 20. INTD1 is controlled by setting the DIC1 register and the vector becomes 21. (See section 3.5.5 Vector table area.)

Interrupt priority levels: DF0 > DF1

Figure 6-3. DIC0 and DIC1

7	6	5	4	3	2	1	0
DF0	DMK0	0	ENCS	0	PR2	PR1	PR0
DF1	DMK1	0	ENCS	0	1	1	1

Caution DIC1 register bits 2 to 0 are fixed to 1.

The DIC1 register interrupt request priority level conforms to the settings of DIC0 register bits PR2 to PR0.

The DF0/DF1 bit is a DMA transfer completion interrupt request flag. The DMK0/DMK1 bit is a DMA transfer completion interrupt mask bit.

See section 4.8 Interrupt Request Control Register for details of other bit fields.

6.4 DMA Service Channels

DMA service channels specify the transfer source, transfer destination to be used for DMA transfers, along with the number of transfers. They are assigned in on-chip RAM as shown below. These areas are assigned to the same areas as macro service channels 0 and 1 and register bank 0.

- For channel 0: $\times\times E00H$ to $\times\times E07H$
- For channel 1: $\times\times E08H$ to $\times\times E0FH$

Remark $\times\times$ indicates the IDB register contents.

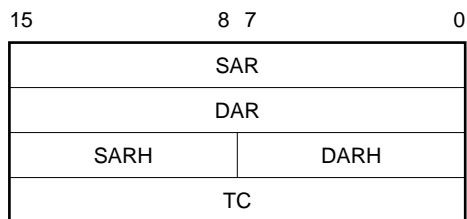
The addresses for the DMA source and destination are the same as for normal memory access and are specified by the offset between segments. However, only the high-order eight bits can be used to specify the segments, and the low-order eight bits are fixed to 0. When the address is updated by a DMA transfer, only this offset is updated.

Memory addresses are output during DMA transfers. Use \overline{DMAAK} to access I/O.

Figure 6-4 shows the structure of the DMA service channel and Figure 6-5 shows the DMA address generation methods.

Remark Address specification for I/O access has no meaning in the demand release mode or one transfer mode.

Figure 6-4. Structure of DMA Service Channel



- SAR (+0H) : Specifies address offset (low-order 16 bits) for DMA transfer source.
- DAR (+2H) : Specifies address offset (low-order 16 bits) for DMA transfer destination.
- DARH (+4H) : Specifies high-order eight bits of segment value for DMA transfer destination address. The segment value's low-order eight bits are fixed to 0.
- SARH (+5H) : Specifies high-order eight bits of segment value for DMA transfer source address. The segment value's low-order eight bits are fixed to 0.
- TC (+6H) : Specifies the number of DMA transfers.

Remark The values shown in parentheses are the offset from the DMA service channel's start address.

Figure 6-5. DMA Address Generation Methods



The DMA service channel is automatically updated during DMA operation. The terminal counter (TC) is decremented (-1) once per DMA transfer (whether for byte data or word data).

The address offset value is updated according to the mode specified by the DMA control register (DMACn). If using byte data, the value is offset by ± 1 or remains unchanged. If using word data, the value is offset by ± 2 or remains unchanged.

6.5 DMA Transfer Timings

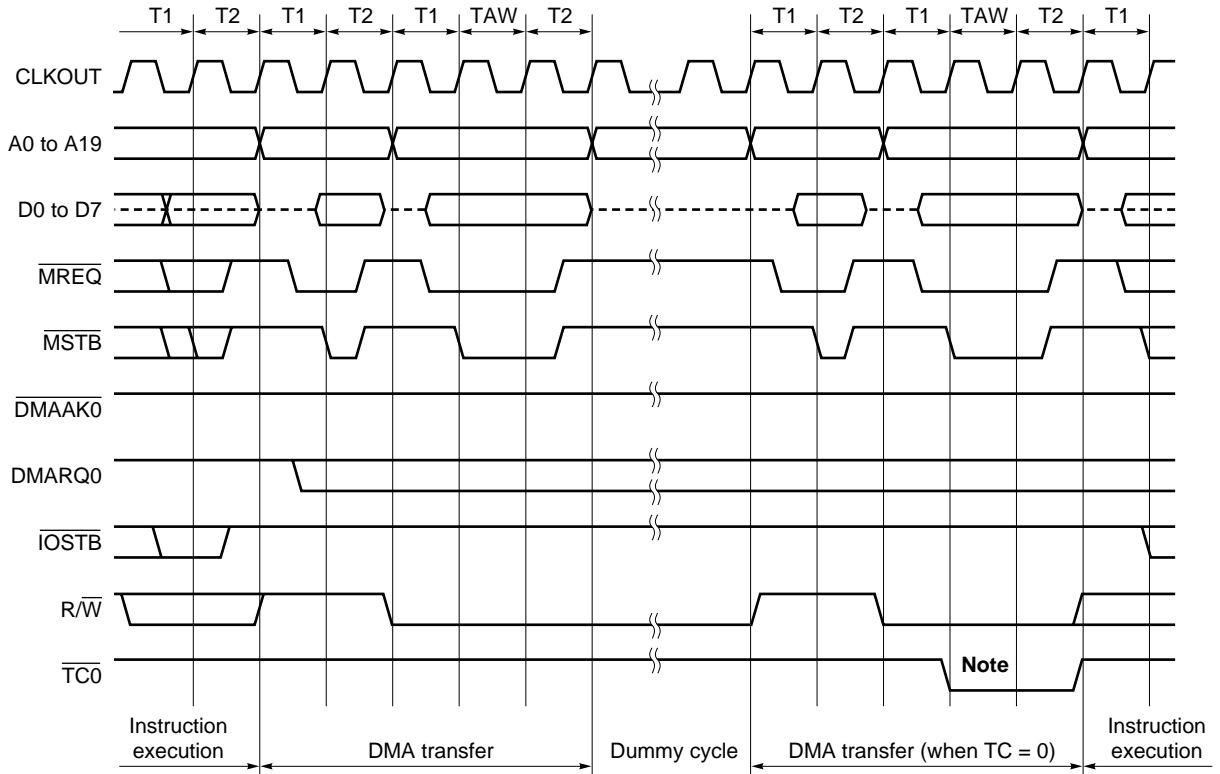
Figures 6-6 to 6-13 show the main DMA transfer timings.

6.5.1 DMA transfer timing of V25

Figure 6-6 shows the burst mode timing with no wait state insertion for the source memory bank and 1-wait state insertion for the destination when DMA is started by the DMARQ signal with TC = 2.

When word transfer is specified for the V25, two DMA cycles are started for each DMA request.

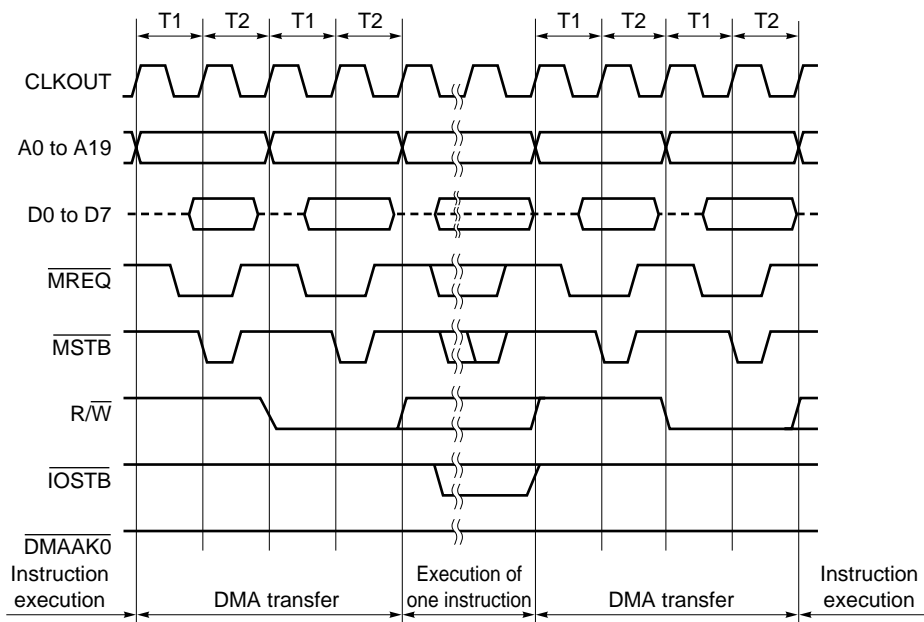
Figure 6-6. When in Burst Mode



Note The active period of TC0 is 1 + W clock cycles (W: number of wait states).

Remark The broken lines indicate high impedance.

Figure 6-7. When in Single-Step Mode (V25)



Remark The broken lines indicate high impedance.

Figure 6-8. One Transfer Mode on V25 (memory to I/O with no wait state insertion)

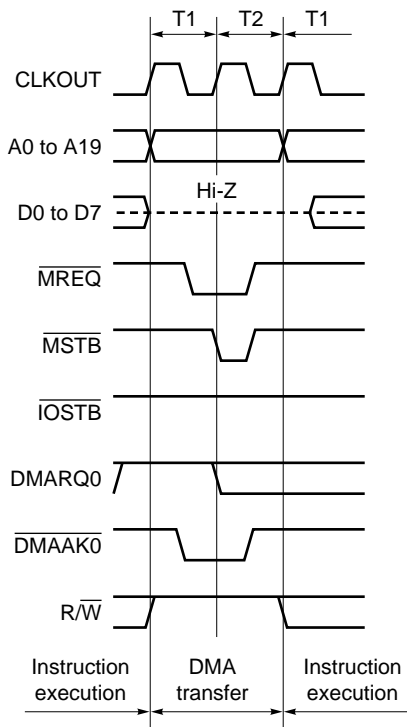
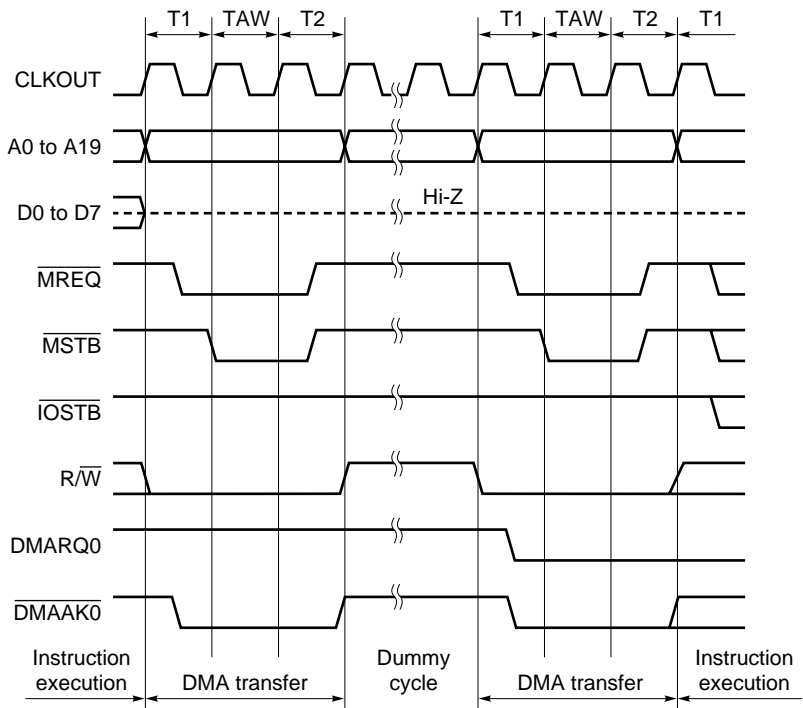


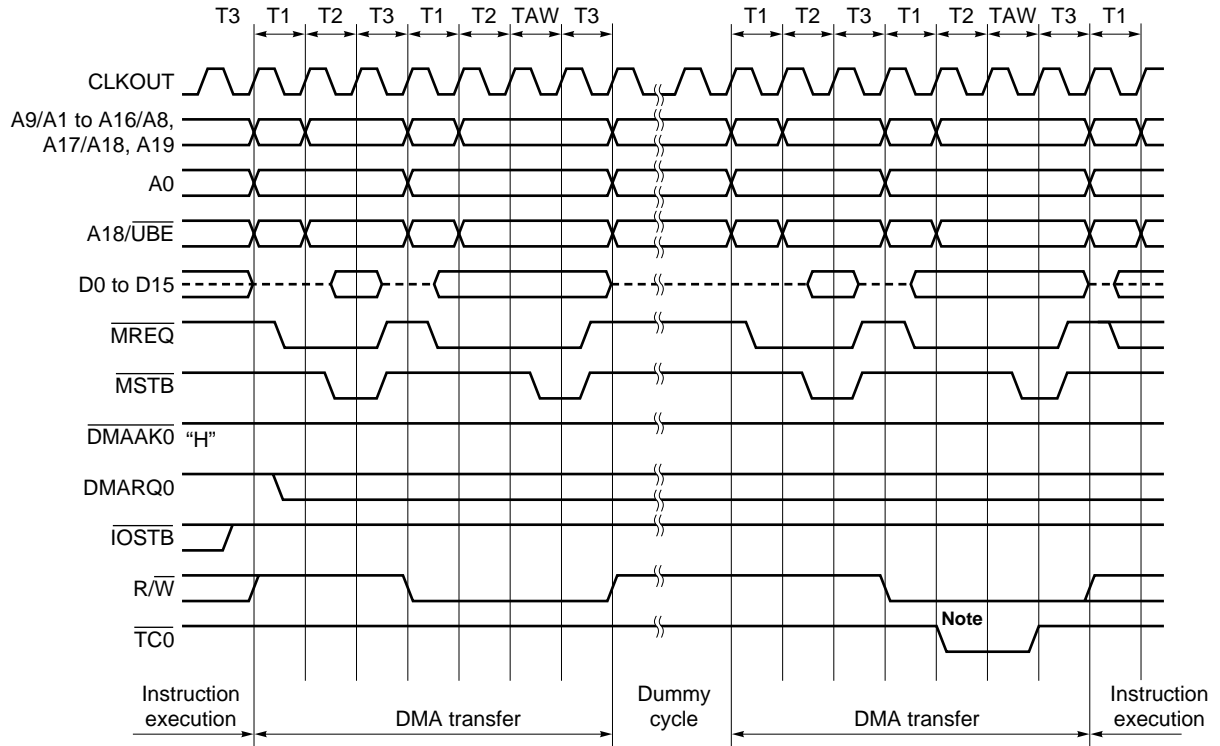
Figure 6-9. Demand Release Mode (I/O to memory, I/O: one wait state insertion, memory: no wait state insertion)



6.5.2 DMA transfer timing of V35

Figure 6-10 shows the burst mode (byte transfer) timing with no wait state insertion for the source memory bank and one wait state insertion for the destination when DMA is started by the DMARQ signal with TC = 2.

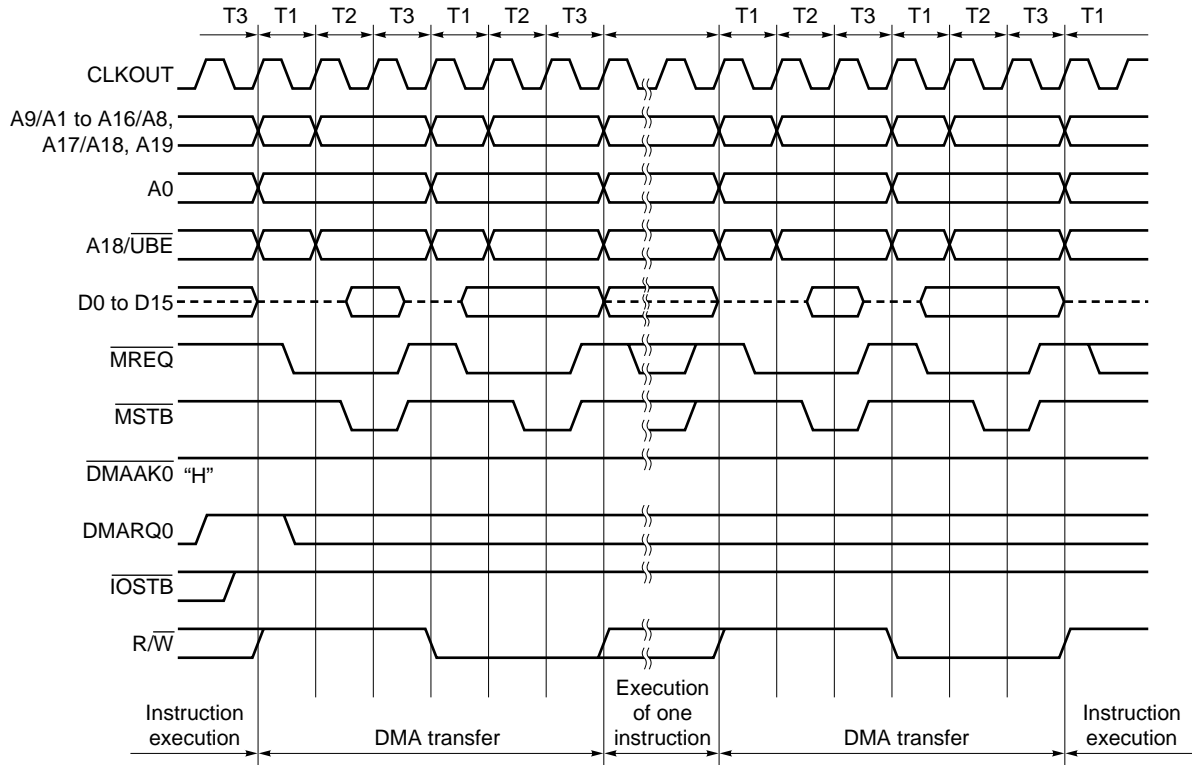
Figure 6-10. When in Burst Mode (byte transfer)



Note The active period of $\overline{\text{TC0}}$ is $1 + W$ clock cycles (W: number of wait states).

Remark The broken lines indicate high impedance.

Figure 6-11. When in Single-Step Mode (V35)



Remark The broken lines indicate high impedance.

Figure 6-12. One Transfer Mode on V35 (memory to I/O with no wait state insertion)

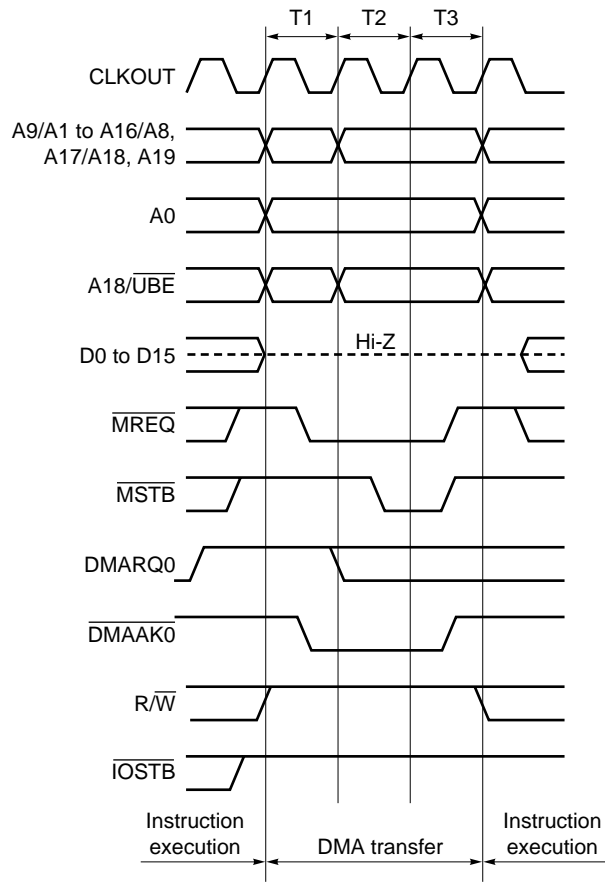
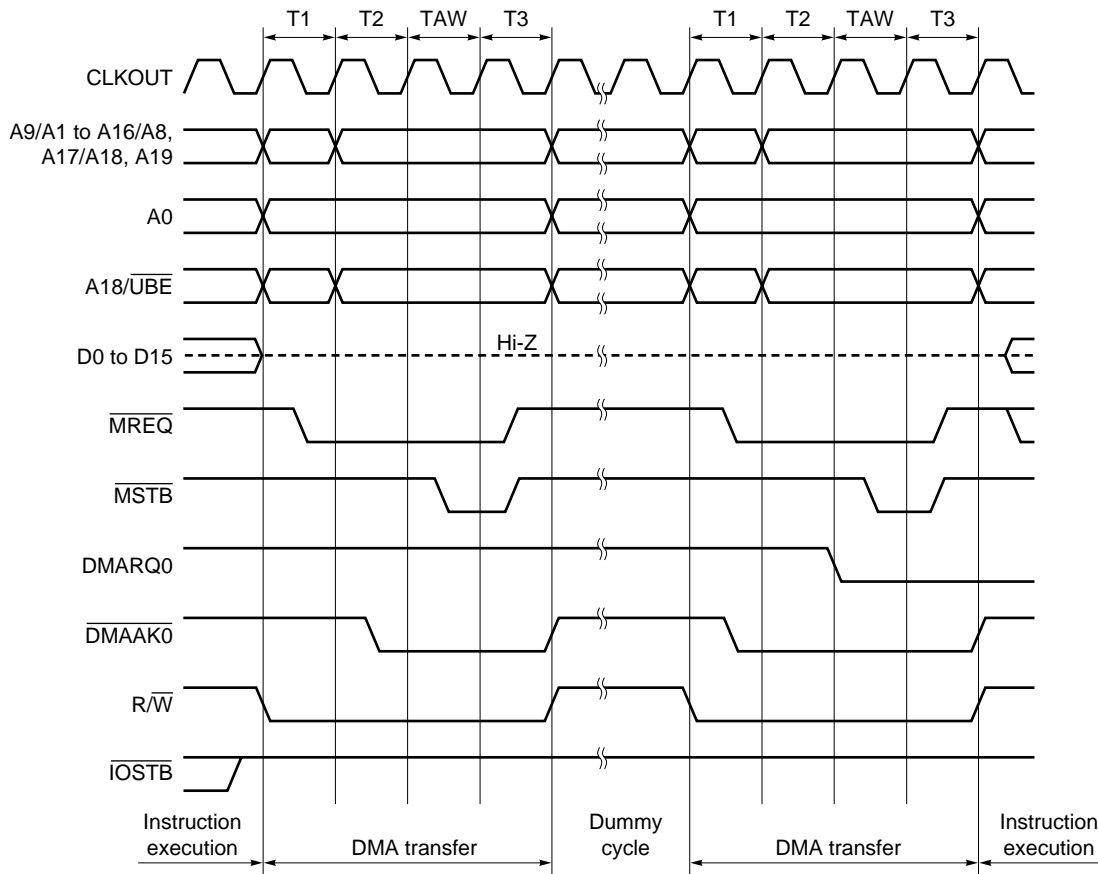


Figure 6-13. Demand Release Mode (I/O to memory, I/O: one wait state insertion, memory: no wait state insertion)



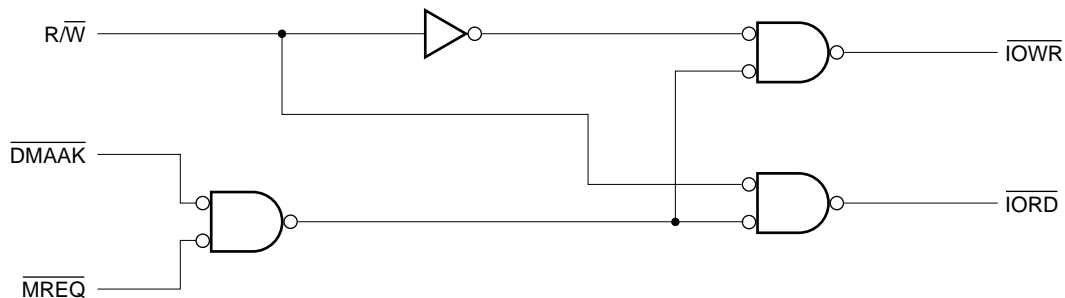
In DMA transfer between I/O and memory, the \overline{IOSTB} signal is not output ($\overline{IOSTB} = \text{high}$) and the \overline{DMAAK} signal is output ($\overline{DMAAK} = \text{low}$).

Therefore, the \overline{DMAAK} signal is used for I/O access decision. In DMA transfer between memory and memory, the \overline{DMAAK} signal is not output.

The delay time of \overline{MREQ} and \overline{DMAAK} during memory to I/O cannot be specified. Design the circuit by considering a case where \overline{DMAAK} is delayed for \overline{MREQ} .

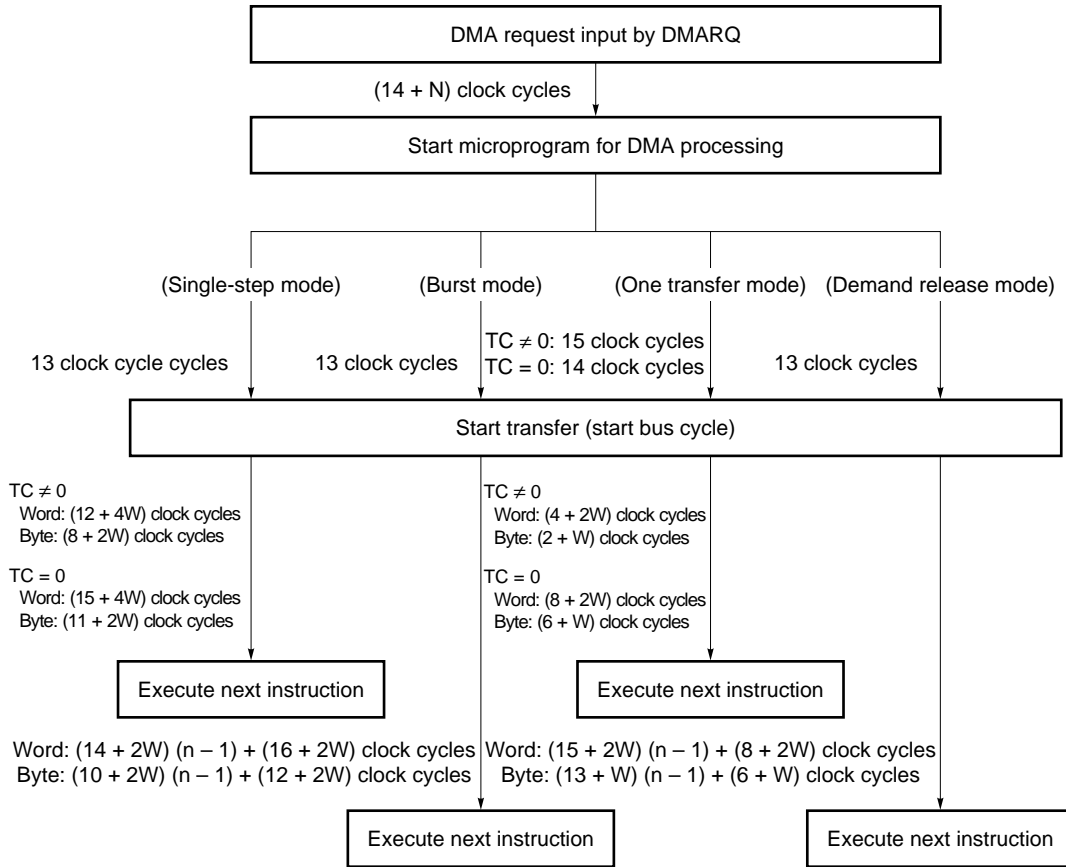
Figure 6-14 shows a circuit example in which the \overline{IORD} and \overline{IOWR} signals are generated from the R/\overline{W} , \overline{IOSTB} , \overline{DMAAK} and \overline{MREQ} signals.

Figure 6-14. Example of \overline{IORD} and \overline{IOWR} Signal Generation Circuit



6.6 DMA Execution Time

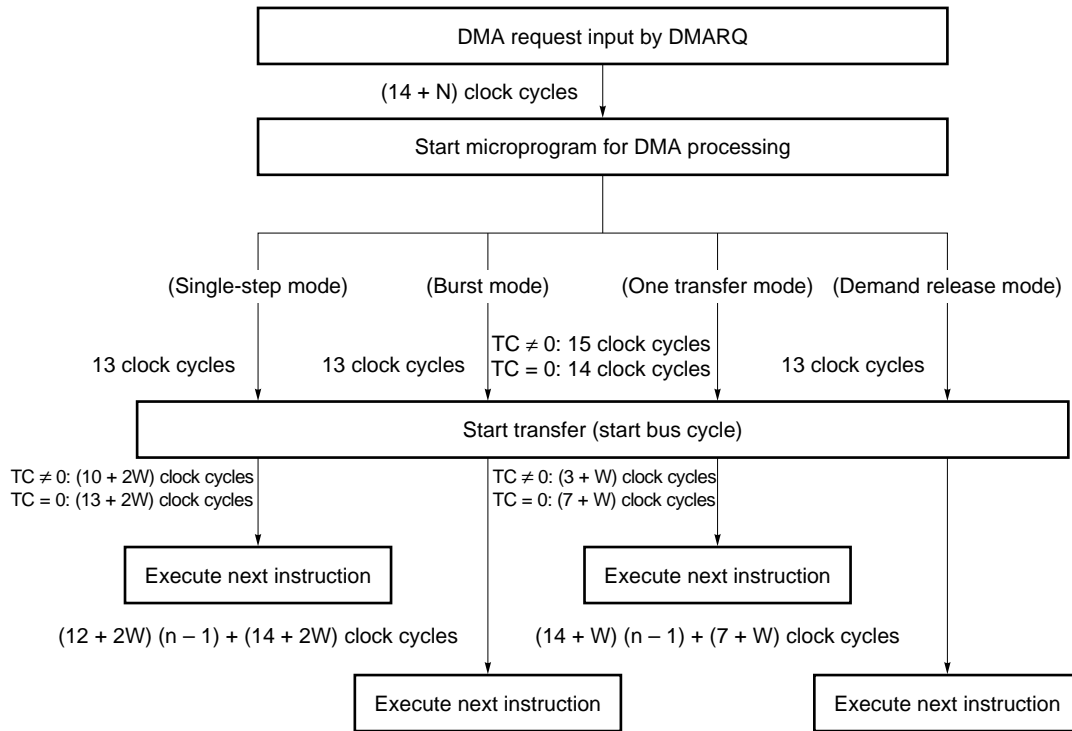
6.6.1 V25's DMA execution time (number of system clock cycles)



Remark W : Number of wait cycles per DMA bus cycle
 n : DMA transfer count
 N : Number of clock cycles still to be executed for instruction being executed

- Cautions**
1. Refresh cycles, hold requests, interrupt requests, and other DMA requests are not considered.
 2. When a DMA request occurs during acknowledgment of an interrupt, DMA is held pending until the interrupt acknowledgment terminates (that is, until the interrupt service routine's starting address is loaded into the PC). See section 4.16 Hardware Interrupt Response Time for description of the time required for interrupt acknowledgment. Depending upon the internal timing, DMA start may precede interrupt acknowledgment when an interrupt occurs between DMARQ and $\overline{\text{DMAAK}} \downarrow$.

6.6.2 V35's DMA execution time (number of system clock cycles)



Remark W : Number of wait cycles per DMA bus cycle
 n : DMA transfer count
 N : Number of clock cycles still to be executed for instruction being executed

- Cautions**
1. Refresh cycles, hold requests, interrupt requests, and other DMA requests are not considered.
 2. When a DMA request occurs during acknowledgment of an interrupt, DMA is held pending until the interrupt acknowledgment terminates (that is, until the interrupt service routine's starting address is loaded into the PC). See section 4.16 Hardware Interrupt Response Time for description of the time required for interrupt acknowledgment. Depending upon the internal timing, DMA start may precede interrupt acknowledgment when an interrupt occurs between DMARQ and $\overline{\text{DMAAK}} \downarrow$.

CHAPTER 7 PORT FUNCTIONS

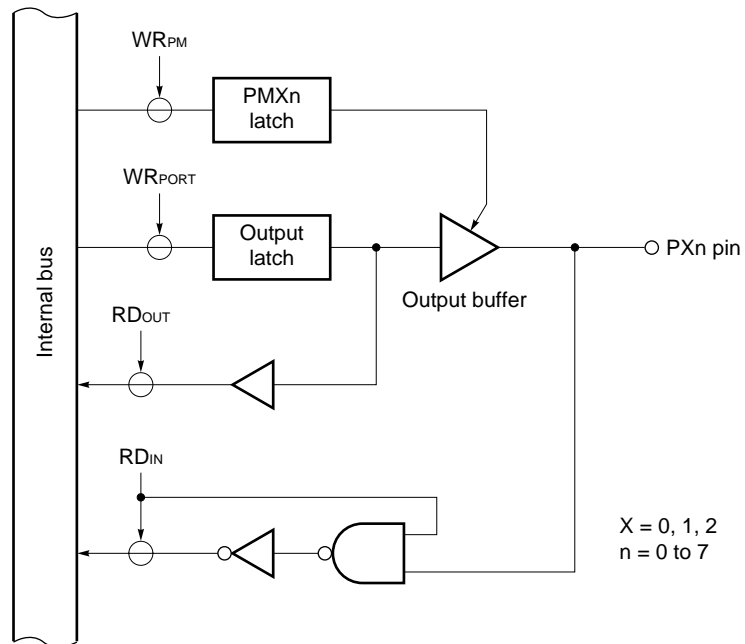
7.1 Ports 0 to 2

7.1.1 Hardware structure

Ports 0 to 2 in the V25 and V35 basically consist of three-state bidirectional ports such as shown in Figure 7-1.

When $\overline{\text{RESET}}$ is input, the port mode register bits are set to 1, specifying the input port mode. All of the port pins become high impedance. The output latch contents are not affected by $\overline{\text{RESET}}$ input.

Figure 7-1. Structure of Ports 0 to 2



Remark PMXn latch: port mode register PMX's bit n.

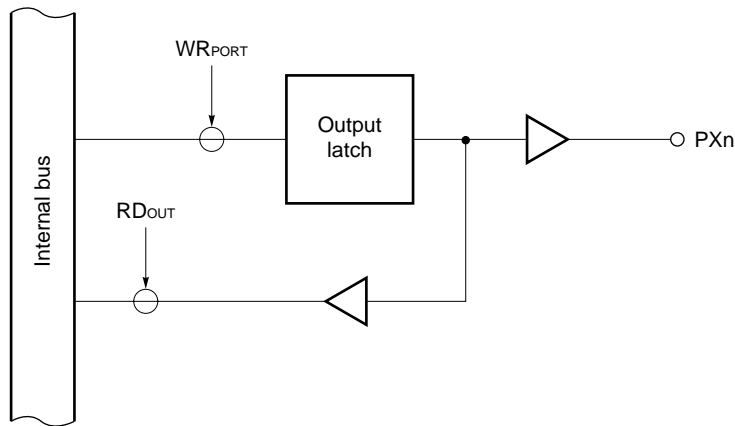
(1) When output port mode is specified ($PMXn = 0$)

The output latch becomes valid, and data can be transferred between the output latch and general-purpose register by executing a transfer instruction.

The output latch contents can be set as desired by executing a logical operation instruction. The data once written into the output latch is retained until a new port handling instruction is executed.

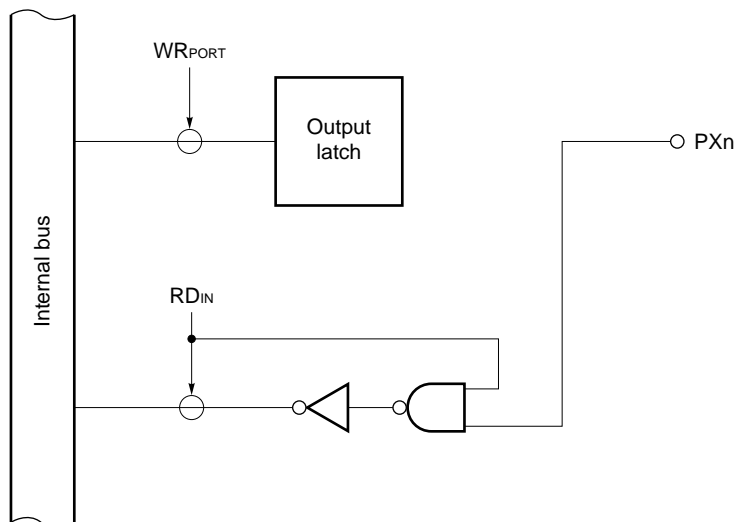
When reading port data, the output latch value is read rather than the output pin's state.

Figure 7-2. Port Set to Output Port Mode

**(2) When input port is specified ($PMXn = 1$)**

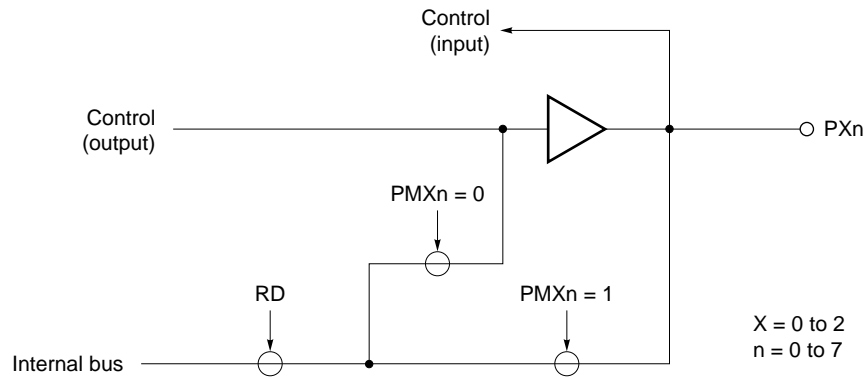
The port pin level can be loaded into the general-purpose register by executing a transfer instruction. In this case, data can be written into the output latch, and the data transferred from the general-purpose register by executing a transfer instruction is all stored in the output latch regardless of input or output port mode specification. However, since the output buffer of the bit set to the input port mode becomes high impedance, no output is made to the port pin. (When the bit set to the input port mode is changed to the output port mode, the output latch contents are output to the port pin.) The output latch contents of the bit set to the input port mode cannot be loaded into the general-purpose register.

Figure 7-3. Port Set to Input Port Mode



(3) When control mode is specified ($PMCX_n = 1$)

Ports 0 to 2 can be used as control signal input or output bit-wise by setting the port mode control register (PMCX) bits to 1 regardless of how the port mode register (PMX) is set. When using the pins as control signals, the control signal state can be read by executing a port access instruction.

Figure 7-4. Port Set to Control Mode**(a) When port is used as a control signal output**

If a port read instruction is executed when the port mode register (PMX_n) is set to 1, the control signal pin state can be read.

If a port read instruction is executed when the port mode register is reset to 0, the internal control signal state can be read.

(b) When port is used as control signal input

If a port read instruction is executed only when the port mode register is set to 1, the control signal pin state can be read.

7.1.2 Port functions

(1) P00 to P07 (port 0) ... three-state input/output

Port 0 is an 8-bit special I/O port. It also functions as a system clock output pin (also used for P07) as well as a general purpose I/O port for which the input or output mode can be selected bit-wise. The function can be selected bit-wise by setting the port 0 mode register (PM0) and port 0 mode control register (PMC0).

Table 7-1. Operation of Port 0 (n = 0 to 7)

	PMC0n = 1	PMC0n = 0	
		PM0n = 1	PM0n = 0
P00	—	Input port	Output port
P01		Input port	Output port
P02		Input port	Output port
P03		Input port	Output port
P04		Input port	Output port
P05		Input port	Output port
P06		Input port	Output port
P07	CLKOUT output ^{Note}	Input port	Output port

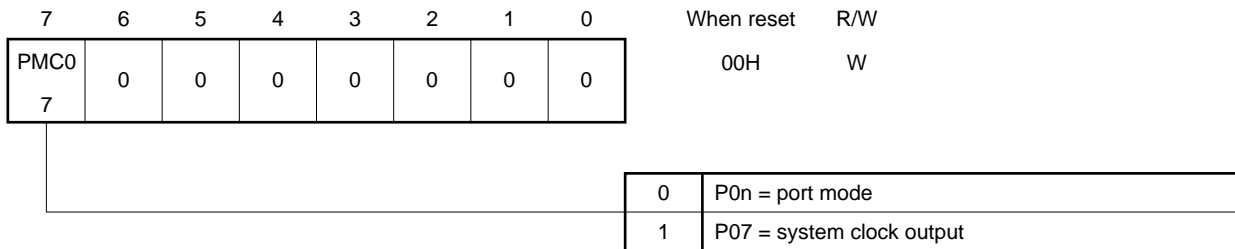
Note See section A.6 Limitation on CPU Deadlock Due to CLKOUT Output.

(a) Port 0 mode control register (PMC0)

The port 0 mode control register (PMC0) is an 8-bit register that selects the port or system clock output mode bit-wise for port 0. This register can only be written by making an 8-bit memory access. If the corresponding bit of the PMC0 register is set to 1, the system clock output mode (P07) is selected; if it is reset to 0, the port mode is selected.

When RESET is input, all of the PMC0 register bits are reset to 0, selecting the port mode.

Figure 7-5. PMC0

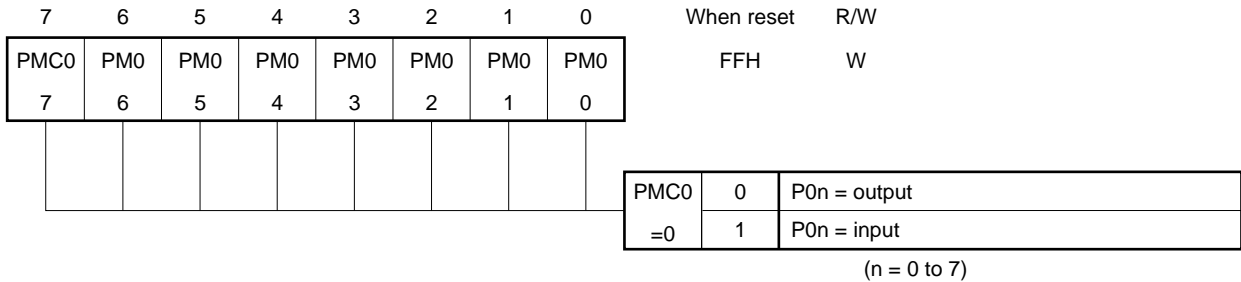


(b) Port 0 mode register (PM0)

The port 0 mode register (PM0) is an 8-bit register that selects the input or output mode bit-wise for port 0. This register can only be written by making an 8-bit memory access. When the corresponding bit of PMC0 is 0, PM0 becomes valid.

When $\overline{\text{RESET}}$ is input, all of the PM0 bits are set to 1 and port 0 becomes an input port.

Figure 7-6. PM0



(2) P10 to P17 (port 1) ... three-state input/output

Port 1 is an 8-bit special I/O port. It also functions as various control pins as well as a general purpose I/O port for which the input or output mode can be selected bit-wise like port 0. The function can be selected bit-wise by setting the port 1 mode register (PM1) and port 1 mode control register (PMC1).

The pin levels for P10 to P13 can be read by directly reading port 1 (P1).

The P10 pin cannot be used as a general-purpose input port.

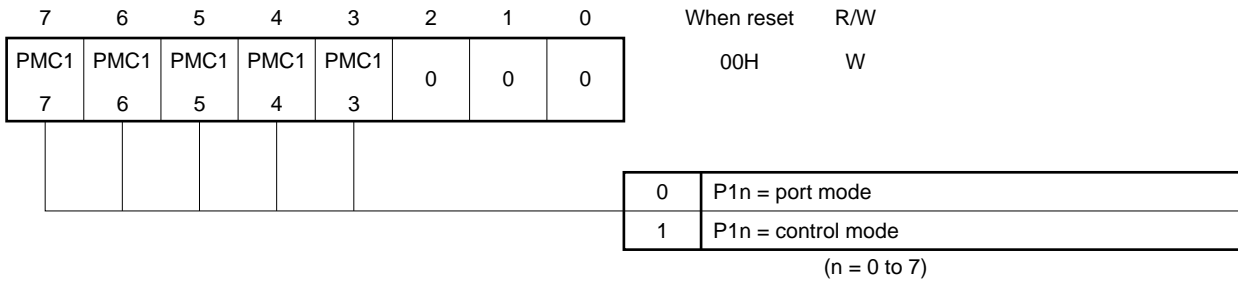
Table 7-2. Operation of Port 1 (n = 0 to 7)

	PMC1n = 1	PMC1n = 0	
		PM1n = 1	PM1n = 0
P10	—	NMI input	—
P11		$\overline{\text{INTP0}}$ input	
P12		$\overline{\text{INTP1}}$ input	
P13	$\overline{\text{INTAK}}$ output	$\overline{\text{INTP2}}$ input	
P14	INT input	Input port ($\overline{\text{POLL}}$ input)	Output port
P15	TOUT output	Input port	Output port
P16	$\overline{\text{SCK0}}$ output	Input port	Output port
P17	READY input	Input port	Output port

(a) Port 1 mode control register (PMC1)

The port 1 mode control register (PMC1) is an 8-bit register that selects the port or control signal input/output mode bit-wise for port 1. This register can only be written by making an 8-bit memory access. If the corresponding bit of the PMC1 register is set to 1, the control signal input/output mode is selected; if it is reset to 0, the port mode is selected. When $\overline{\text{RESET}}$ is input, all PMC1 register bits are reset to 0, selecting the port mode. However, P10 to P12 are fixed to the port mode.

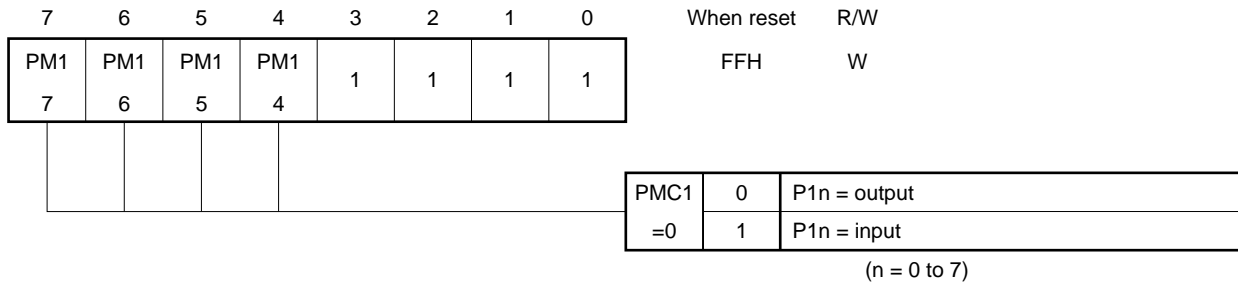
Figure 7-7. PMC1



(b) Port 1 mode register (PM1)

The port 1 mode register (PM1) is an 8-bit register that selects the input or output mode bit-wise for port 1. This register can only be written by making an 8-bit memory access. When the corresponding bit of the PMC1 register is set to 0, PM1 becomes valid. When $\overline{\text{RESET}}$ is input, all the PM1 bits are set to 1, selecting the input mode.

Figure 7-8. PM1



(3) P20 to P27 (port 2) ... three-state input/output

Port 2 is an 8-bit special I/O port. It also functions as various control pins as well as a general purpose I/O port for which the input or output mode can be selected bit-wise like port 0. The function can be selected bit-wise by setting the port 2 mode register (PM2) and port 2 mode control register (PMC2).

Table 7-3. Operation of Port 2 (n = 0 to 7)

	PMC2n = 1	PMC2n = 0	
		PM2n = 1	PM2n = 0
P20	DMARQ0 input	Input port	Output port
P21	$\overline{\text{DMAAK0}}$ input	Input port	Output port
P22	$\overline{\text{TC0}}$ output	Input port	Output port
P23	DMARQ1 input	Input port	Output port
P24	$\overline{\text{DMAAK1}}$ output	Input port	Output port
P25	$\overline{\text{TC1}}$ output	Input port	Output port
P26	$\overline{\text{HLDAK}}$ output	Input port	Output port
P27	HLDRQ input	Input port	Output port

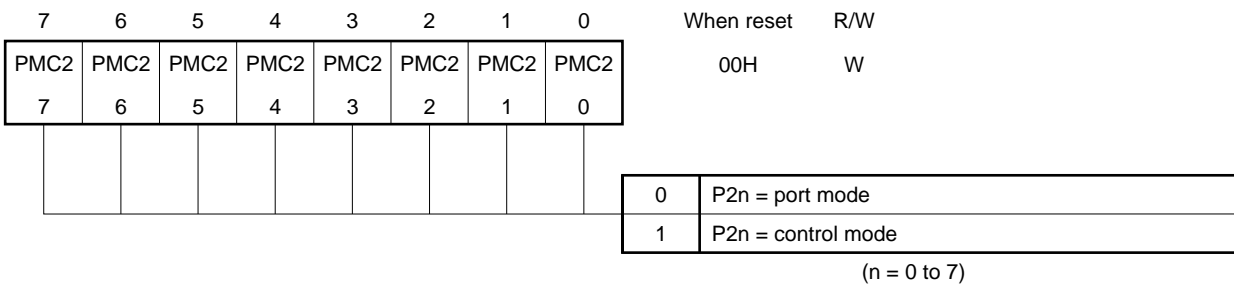
(a) Port 2 mode control register (PMC2)

The port 2 mode control register (PMC2) is an 8-bit register that selects the port or control signal input/output mode bit-wise for port 2.

This register can only be written by making an 8-bit memory access.

If the corresponding bit of the PMC2 register is set to 1, the control signal input/output mode is selected; if it is reset to 0, the port mode is selected. When $\overline{\text{RESET}}$ is input, all PMC2 register bits are reset to 0, selecting the port mode.

Figure 7-9. PMC2



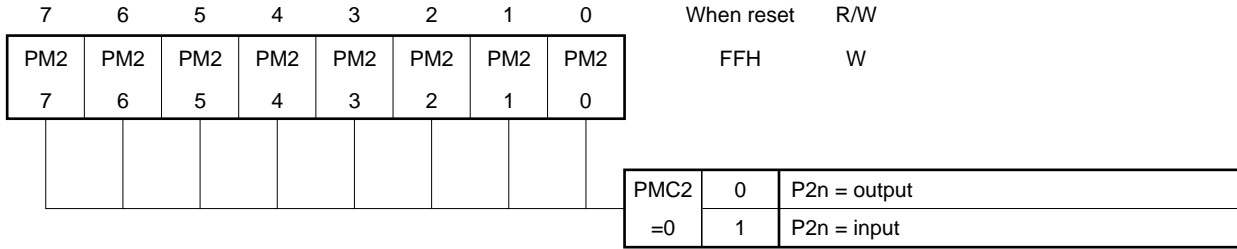
(b) Port 2 mode register (PM2)

The port 2 mode register (PM2) is an 8-bit register that selects the input or output mode bit-wise for port 2. This register can only be written by making an 8-bit memory access.

When the corresponding bit of the PMC2 register is set to 0, the various PM2 bits become valid.

When $\overline{\text{RESET}}$ is input, all of the PM2 bits are set to 1, selecting input mode.

Figure 7-10. PM2



(4) Cautions

The following cautions should be observed when using the above pins as output ports after a reset is input.

- **Power-on reset**
Because the contents of the output latch are undefined, when using these pins as an output port, first write the values to be output to the port, then switch the port mode register from input mode to output mode. If the mode is switched before setting the port, undefined values will be output from the port pins.
- **System reset**
Because the output latch value prior to reset is retained, switching the port mode register from input mode to output mode causes the output latch value prior to reset to be output from the port pins.

7.2 Port T (PT0 to PT7)

Port T is an 8-bit input port that enables the threshold voltage (reference voltage) to be changed among 16 stages. Comparator operation is performed according to the analog input.

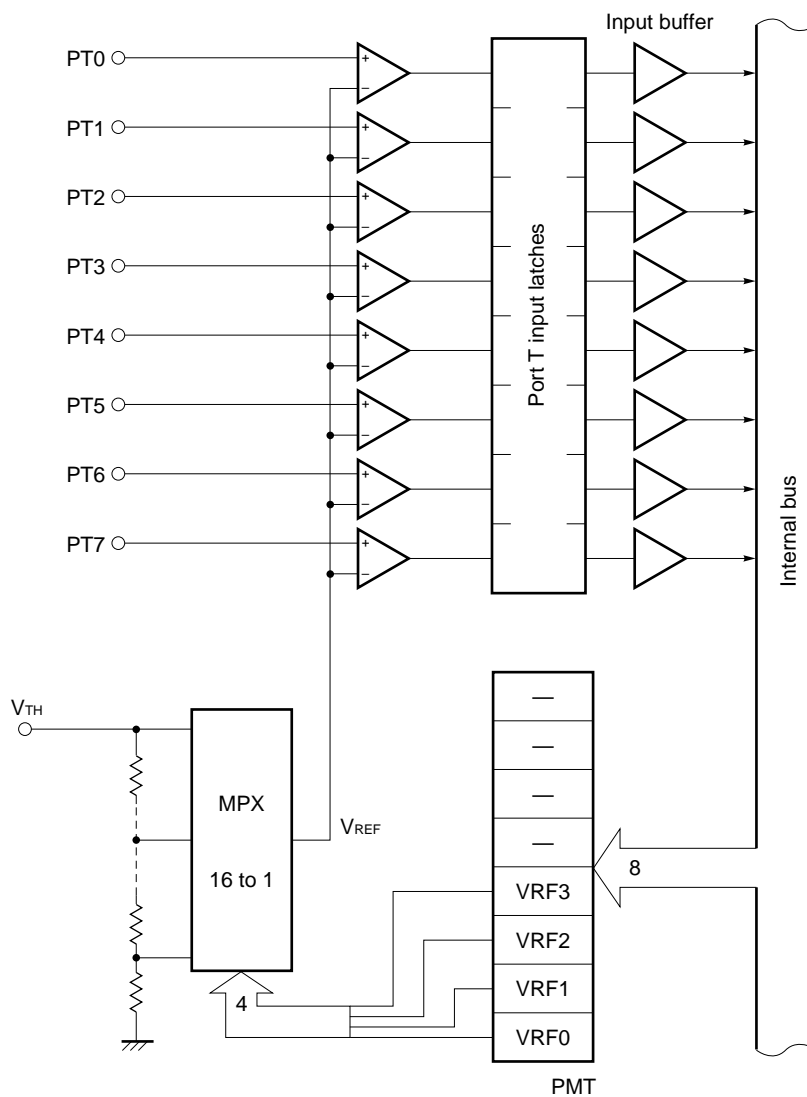
7.2.1 Hardware structure

Port T consists of comparator input for PT0 to PT7, the V_{TH} pin (reference voltage input), the multiplexer (MPX) that selects the comparison voltage (V_{REF}) among 16 types (from $1/16 \times V_{TH}$ to $16/16 \times V_{TH}$), the port mode T register (PMT) that controls the MPX, and eight latches, as shown in Figure 7-11.

The comparator compares V_{REF} , selected by setting PMT, with PT0 to PT7 input and the result is latched in the port T input latches.

$$\begin{cases} V_{REF} > PTn \rightarrow 0 \\ V_{REF} < PTn \rightarrow 1 \end{cases}$$

Figure 7-11. Block Diagram of Port T



Caution The V_{TH} pin is connected via high resistance to the GND pin. Therefore, if voltage is applied to the V_{TH} pin in the standby mode, the consumption current will increase.

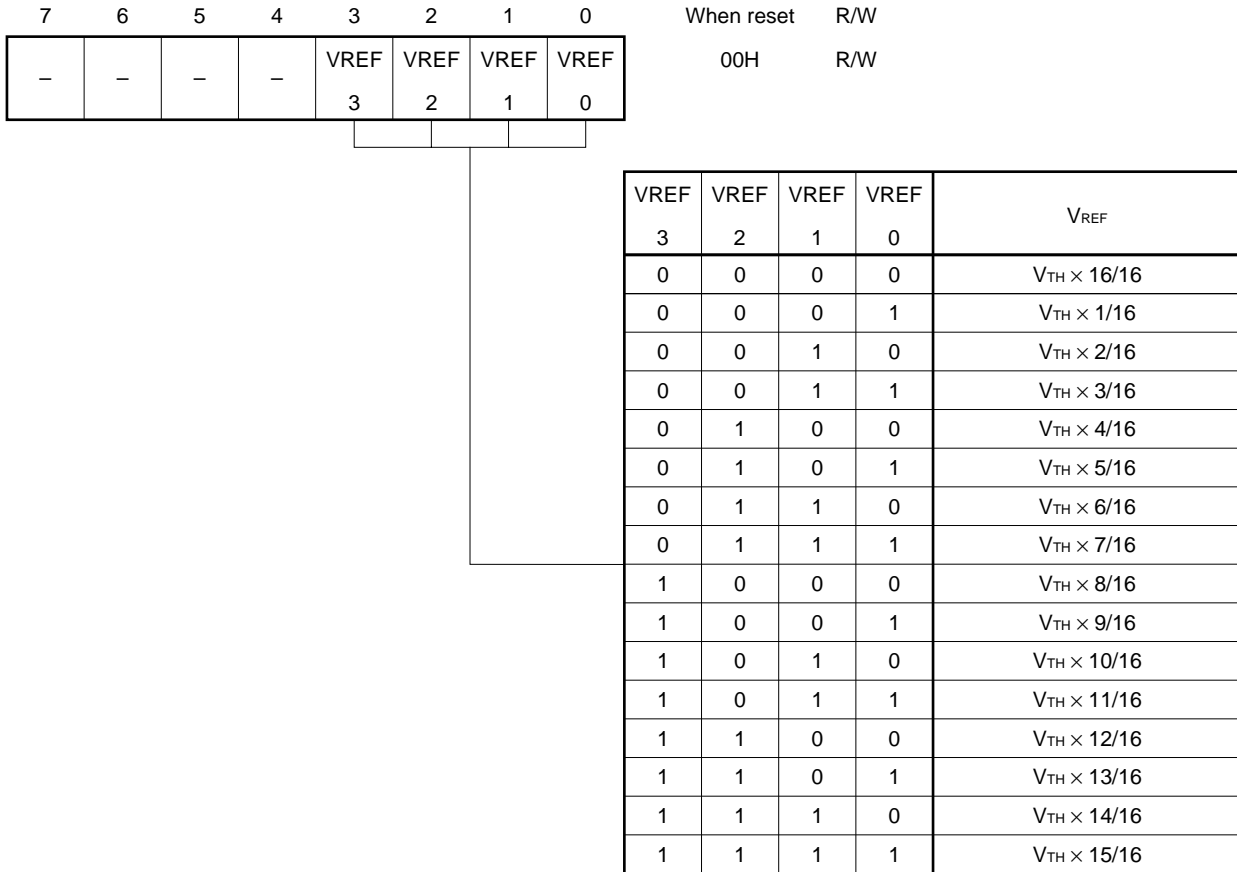
7.2.2 Port T mode register (PMT)

The port T mode register (PMT) is an 8-bit register that selects the comparator's comparison voltage (V_{REF}) among the 16 types shown in Figure 7-12.

This register can be written/read by making an 8-bit or 1-bit memory access.

When \overline{RESET} is input, all of the PMT bits are reset to 0.

Figure 7-12. PMT



7.2.3 Port T reception

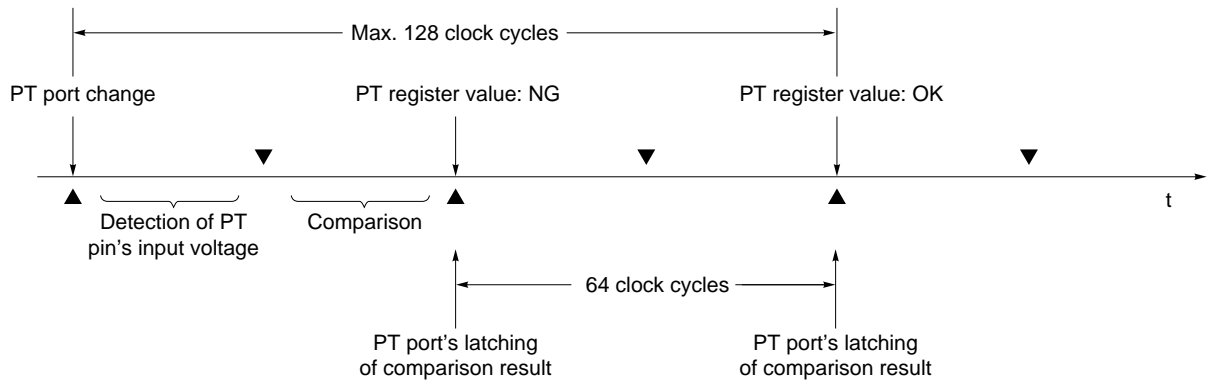
The comparison operation for PT0 to PT7 is executed in the following two stages.

First stage : PT pin voltage is detected.

Second stage : voltage detected at first stage is compared with voltage set by PMT register.

The comparison period, which is the sum of the execution times for the first and second stages, is 64 clocks.

The first and second stages are always executed alternately. The start timing cannot be controlled by software or hardware. Therefore, to obtain an accurate comparison result after the PT pin's input voltage or comparison voltage has been changed, wait until 128 clock cycles have elapsed before reading the PT register value. During this period, make sure that the PT pin's input voltage is stable.



[MEMO]

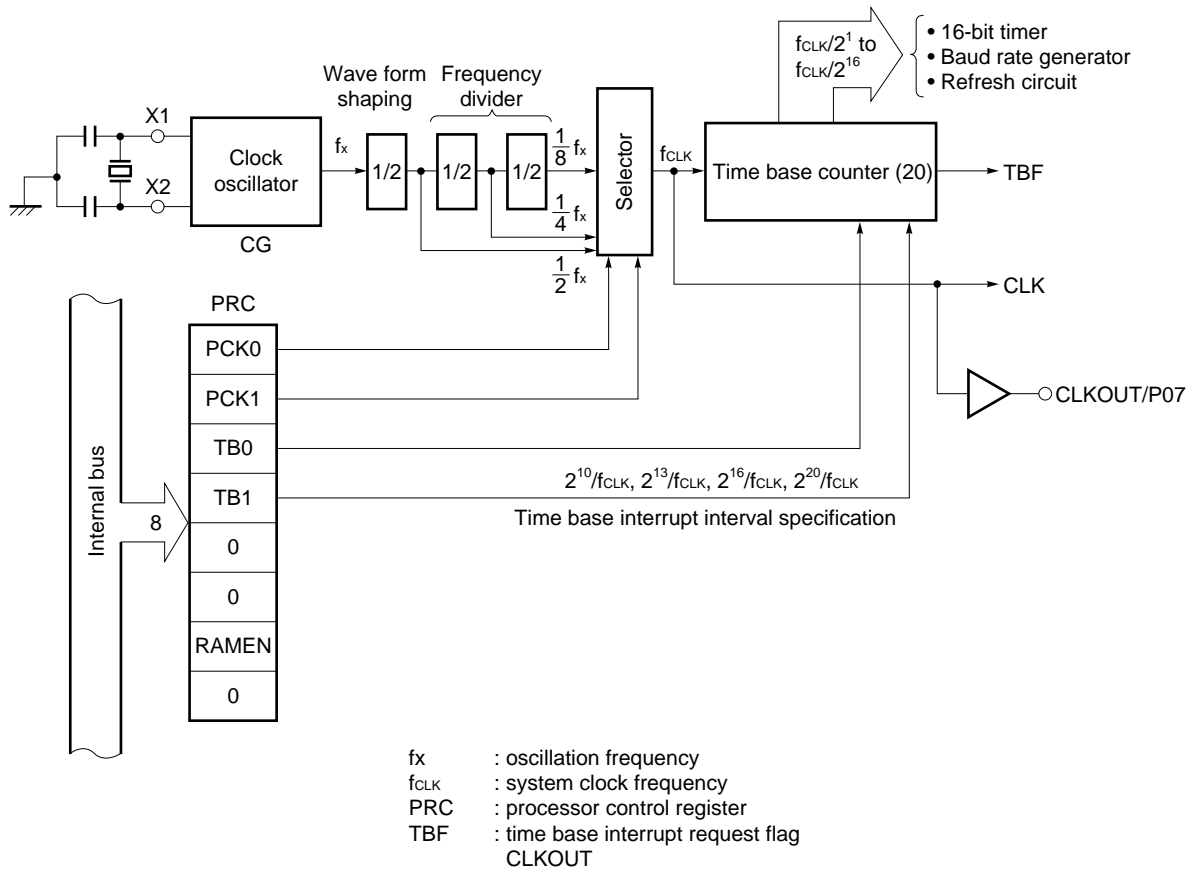
CHAPTER 8 CLOCK GENERATOR

The clock generator supplies various clocks to the CPU and peripheral hardware and also controls the CPU operation mode.

8.1 Clock Generator Structure

Figure 8-1 shows a block diagram of the clock generator.

Figure 8-1. Block Diagram of Clock Generator



The clock generator clock oscillates using a crystal or ceramic resonator connected to the X1 and X2 pins.

The clock oscillator output frequency is divided by two for wave form shaping. The dividing ratio is selected for use as a system clock (CLK).

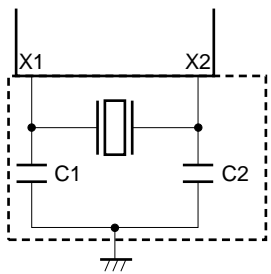
The CLK dividing ratio can be selected among 1/2, 1/4, and 1/8 of the oscillation frequency (f_x) by setting processor control register (PRC) bits 0 and 1 (PCK0 and PCK1).

As CLK is set to low speed, the system clock frequency becomes low, and long operation is possible even when the voltage is lowered in a battery-driven system.

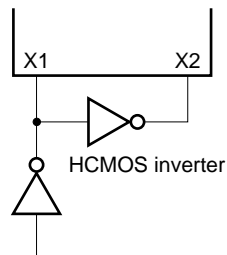
External clocks can also be input.

Figure 8-2. Clock Generator External Circuit

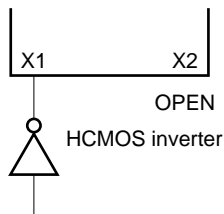
(a) Crystal or ceramic oscillation



(b) External clock input



(c) External clock input (X2: open)



- Cautions**
1. Put the external components as close as possible to the X1 and X2 pins.
 2. Do not pass any other signal through the area indicated with dotted lines.

Table 8-1. Recommended Ceramic Resonators

Manufacturer	Product name	Recommended constants	
		C1 [pF]	C2 [pF]
Kyocera	KBR-10.0M	33	33
Murata	CSA7.37MT040	100	100
	CSA10.0MT	47	47
	CSA11.0MT		
	CSA16.0MX040	30	30
TDK	FCR10.0M2S	30	30
	FCR16.0M2S	15	6
	FCR16.0M2G	22	10

Table 8-2. Recommended Crystal Resonators

Manufacturer	Product name	Recommended constants	
		C1 [pF]	C2 [pF]
Kinseki	HC-49/U (KR-100)	22	22
	HC-49/U (KR-160)	22	22

Remark Be sure to check with the resonator manufacturer regarding resonator characteristics, etc.

8.2 Processor Control Register (PRC)

The processor control register (PRC) is an 8-bit register that controls CPU and internal system control items such as the CPU operation clock, time base interrupt intervals, and enabling of internal RAM memory references.

This register can be written/read by making an 8-bit or 1-bit memory access.

When $\overline{\text{RESET}}$ is input, the PRC contents are initialized to 4EH.

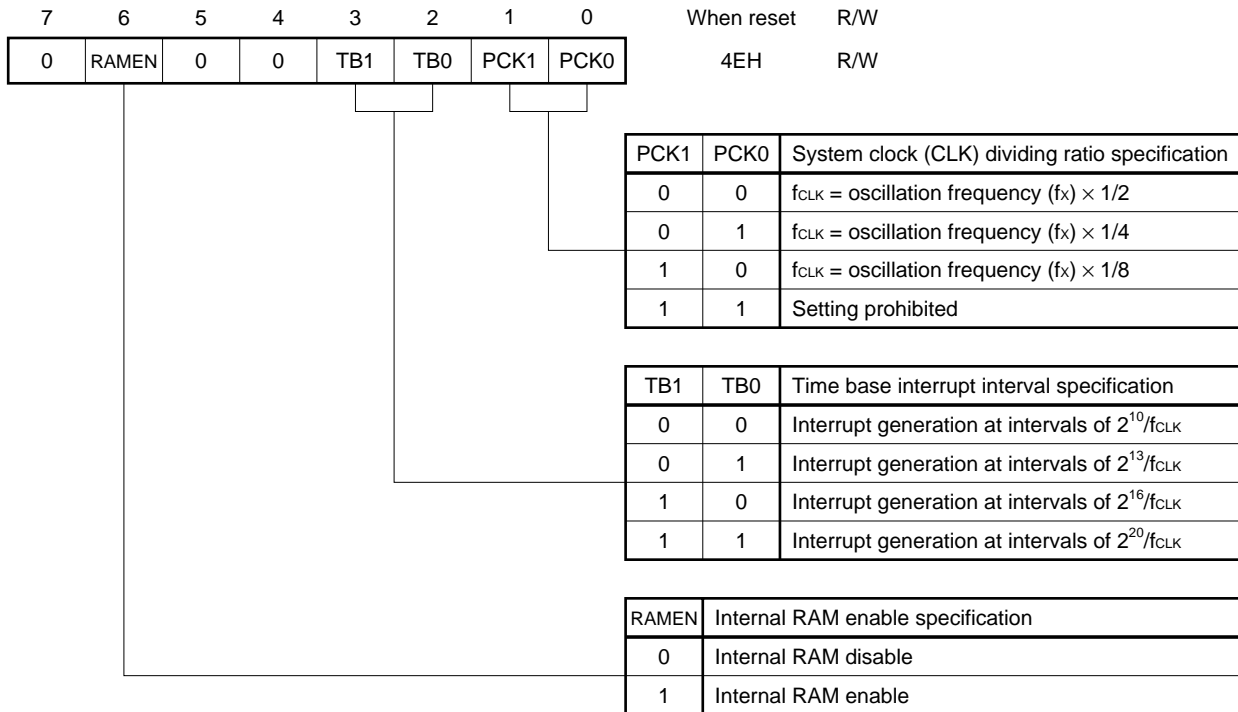
The system clock dividing ratio is specified by setting the PCK0 and PCK1 bits. The oscillation frequency is divided by the value specified in PCK0 and PCK1, and the result is used as the system clock (CLK).

The time base interrupt intervals are specified by setting the TB0 and TB1 bits. One of four long interval time types can be selected in the TB0 and TB1 bits.

Enabling of internal RAM memory references is controlled by setting the RAMEN bit. When the RAMEN bit is set to 0 (memory reference disable), address judgment of internal RAM is not made and external memory is always accessed.

When RAM is referenced as a register, internal RAM is always accessed.

Figure 8-3. PRC



CHAPTER 9 TIMER UNIT

The timer unit in the V25 and V35 can be used as an interval timer, a one-shot timer, or a square wave output.

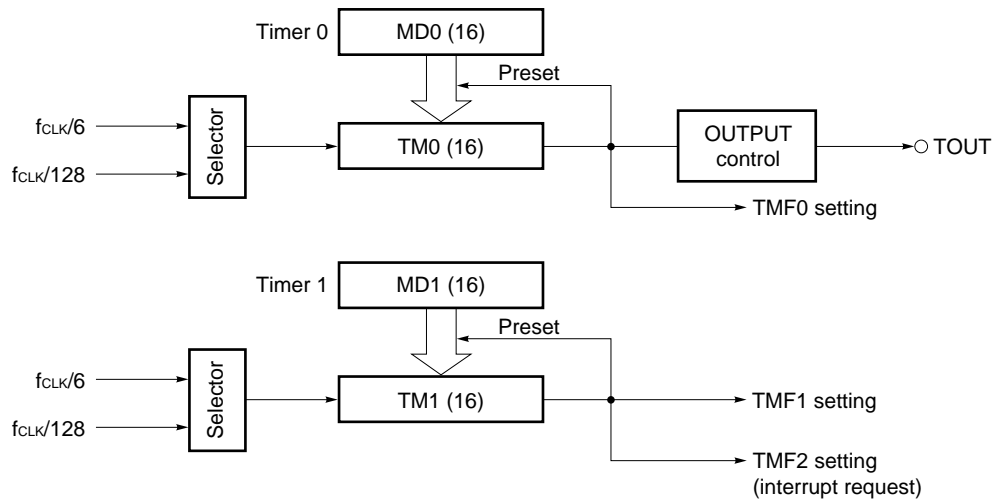
9.1 Timer Unit Structure and Operation

The timer unit consists of two 16-bit timer registers, two 16-bit modulo/timer registers, and one 8-bit timer control register. The structure and operation of each operation mode are explained below.

(1) Interval timer mode

When the timer unit is set to the interval timer mode, timers 0 and 1 can be used as shown in Figure 9-1.

Figure 9-1. Timer Unit Structure during Interval Timer Mode



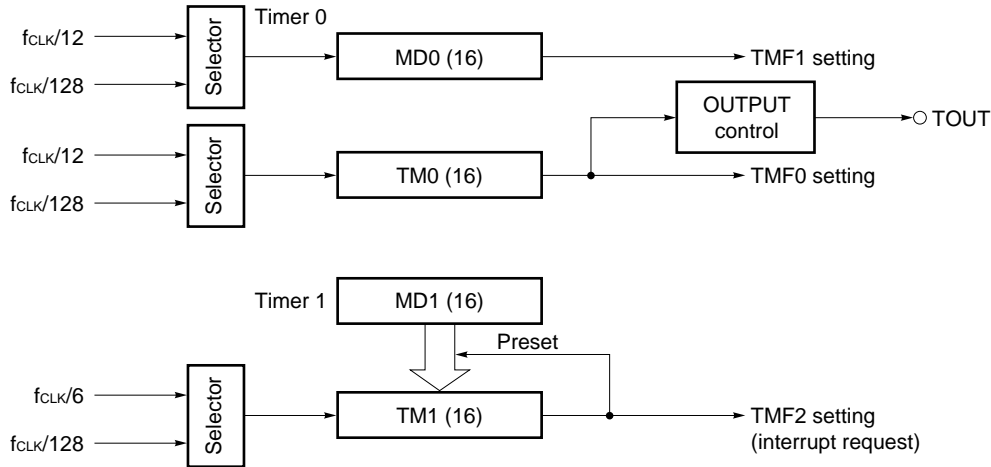
When the interval timer mode is specified in the timer control register (TMC0) and the TS0 bit is set to 1, the MD0 register value is set in the TM0 register and the clock specified in the TCLK0 bit is counted down. If the timer value is set to 0 during the countdown, the MD0 register value is again set in the TM0 register and the clock is counted down. (Count value = setting value + 1)

A similar count operation is also performed for the timer 1 registers.

(2) One-shot timer mode

When the timer unit is set to the one-shot timer mode, timer 0 is used as shown in Figure 9-2. However, timer 1 can also be operated as an interval timer at the same time.

Figure 9-2. Timer Unit Structure during One-shot Timer Mode



When the one-shot timer mode is specified in the timer control register (TMC0) and the TS0/MS0 bit is set to 1, the clock specified in the TCLK0/MCLK0 bit is counted down in the TM0/MD0 register. When the timer value is set to 0 during the countdown, the count operation stops. At that time, 0000H is retained in the TM0/MD0 register.

9.2 Timer Control Registers (TMC0 and TMC1)

The TMC0 register is an 8-bit register that controls TM0 and MD0 register operations. The TMC1 register is an 8-bit register that controls TM1 and MD1 register operations.

The registers can be written/read by making an 8-bit or 1-bit memory access, in which case up to six wait states are inserted.

When $\overline{\text{RESET}}$ is input, the contents of each register are initialized to 00H. The TMC0 and TMC1 registers differ in format as shown below.

(Interval timer mode, one-shot timer mode)

	7	6	5	4	3	2	1	0
TMC0	TS0	TCLK0	MS0	MCLK0	ENTO	ALV	MOD1	MOD0

(Interval timer mode)

	7	6	5	4	3	2	1	0
TMC1	TS1	TCLK1	0	0	0	0	0	0

The operation mode of timer 0 consisting of TM0 and MD0 (or timer 1 consisting of TM1 and MD1) is specified by setting TMC0 (or TMC1) register bits 0 and 1 (MOD0 and MOD1).

MOD0 and **MOD1**: Timer 0 (or timer 1) operation mode specification bits

Set both MOD0 and MOD1 to 0 to select interval timer operation mode. Set MOD0 to 1 and MOD1 to 0 to select one-shot timer operation mode. In the interval timer operation mode, TM0 (or TM1) serves as a timer register for counting down the setup value and MD0 (or MD1) serves as a modulo register for retaining the interval setup value. In the one-shot timer operation mode, both TM0 and MD0 serve as timer registers for counting down the setup value. TMC1 bits 0 and 1 are fixed to 0 and timer 1 operates only as an interval timer.

Therefore, timer 0 consisting of TM0 and MD0 operates as a 16-bit interval timer or 16-bit one-shot timer. Timer 1 consisting of TM1 and MD1 operates as a 16-bit interval timer.

Timer 0 can also output a square wave to the TOUT pin. Square wave output to the TOUT pin is controlled by setting the TMC0 register. However, because the TOUT pin is also used for P15, set port 1 mode control register bit 5 (PMC15) to 1 to set the pin to the control mode before outputting a square wave to the TOUT pin. At this time, square wave output to the TOUT pin depends on the CPU's internal timing and does not depend on CLKOUT output.

ALV : Active level specification bit of TOUT pin output

When the ENTO bit is reset to 0, the TOUT pin output becomes active-low when the ALV bit is set to 1 and active-high when the ALV bit is reset to 0.

ENTO : Bit specifying operation of square wave output to TOUT pin

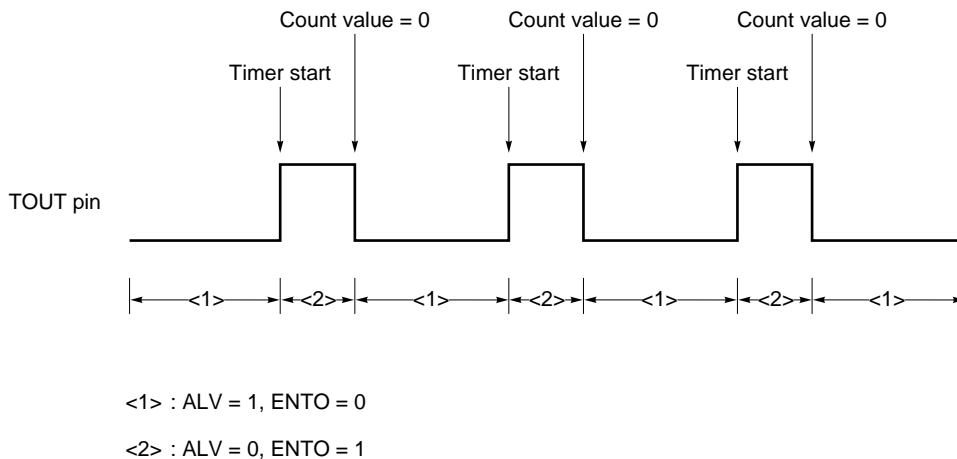
When the ENTO bit is reset to 0, the TOUT pin level becomes inactive as specified in the ALV bit when in interval timer mode.

During one-shot timer mode, the following changes occur according to the setup value in the ALV bit.

ALV	Timing of change	TOUT pin
0	TS0 = 1	1
	TMF0 = 1	0
1	TS0 = 1	0
	TMF0 = 1	1

When the ENTO bit is set to 1, the TOUT pin level is inverted according to the timing whereby the timer unit interrupt request flag (TMF0) is set to 1.

Figure 9-3. Output State of TOUT Pin (during One-shot Timer Mode)



When the timer counter is started, <2> is set.

After completion of the countdown (after inversion of TOUT output), set as shown for <1>.

Other bits in the TMC0 and TMC1 registers are explained below for each operation mode.

(1) Interval timer mode (MOD0 = 0, MOD1 = 0): Timers 0 and 1

TCLK0 and **TCLK1** : Count clock specification bits in TM0 and TM1 registers

Table 9-1 lists the reference values when the system clock frequency (f_{CLK}) is 8 MHz.

TS0 and **TS1** : Operation control bits for timers 0 and 1

When the TS0 (or TS1) bit is set to 1, the MD0 (MD1) register value is set in the TM0 (TM1) register and the TM0 (TM1) register starts counting down. When the TS0 (TS1) bit is reset to 0, the TM0 (TM1) register retains the contents of the TM0 (TM1) and MD0 (MD1) registers and stops counting down.

If the timer value is set to 0 or if the TS0 (or TS1) bit is again set to 1 during the countdown, the MD0 (MD1) register value is again set in the TM0 (TM1) register and the TM0 (TM1) register again starts counting down.

Table 9-1. TMn Count Time in Interval Timer Mode (n = 0 or 1)

When $f_{CLK} = 8 \text{ MHz}$

TCLKn	Count clock	Resolution	Full count
0	$f_{CLK}/6$	$0.75 \mu\text{s}$	49.1 ms
1	$f_{CLK}/128$	$16 \mu\text{s}$	1.04 s

(2) One-shot timer mode (MOD0 = 1, MOD1 = 0): Timer 0 only

TCLK0 : TM0 register count clock specification bit

Table 9-2 lists the reference values when the system clock frequency (f_{CLK}) is 8 MHz.

TS0 : TM0 register operation control bit

When the TS0 bit is set to 1, the countdown is started from the retained TM0 register contents. When the timer value is set to 0, the TS0 bit is reset to 0 and count operation is stopped. When the TS0 bit is reset to 0, the TM0 register retains the current value and stops counting.

MCLK0 : MD0 register count clock specification bit

Table 9-2 lists the reference values when the system clock frequency (f_{CLK}) is 8 MHz. When the interval timer mode is set, the MCLK0 bit does not affect the count operation.

MS0 : MD0 register count operation control bit

When the MS0 bit is set to 1, the countdown is started from the retained MD0 register contents. When the timer value is set to 0, the MS0 bit is reset to 0 and the count operation is stopped. When the MS0 bit is reset to 0, the MD0 register retains the current value and stops counting.

When the interval timer mode is set, the MS0 bit does not affect the count operation.

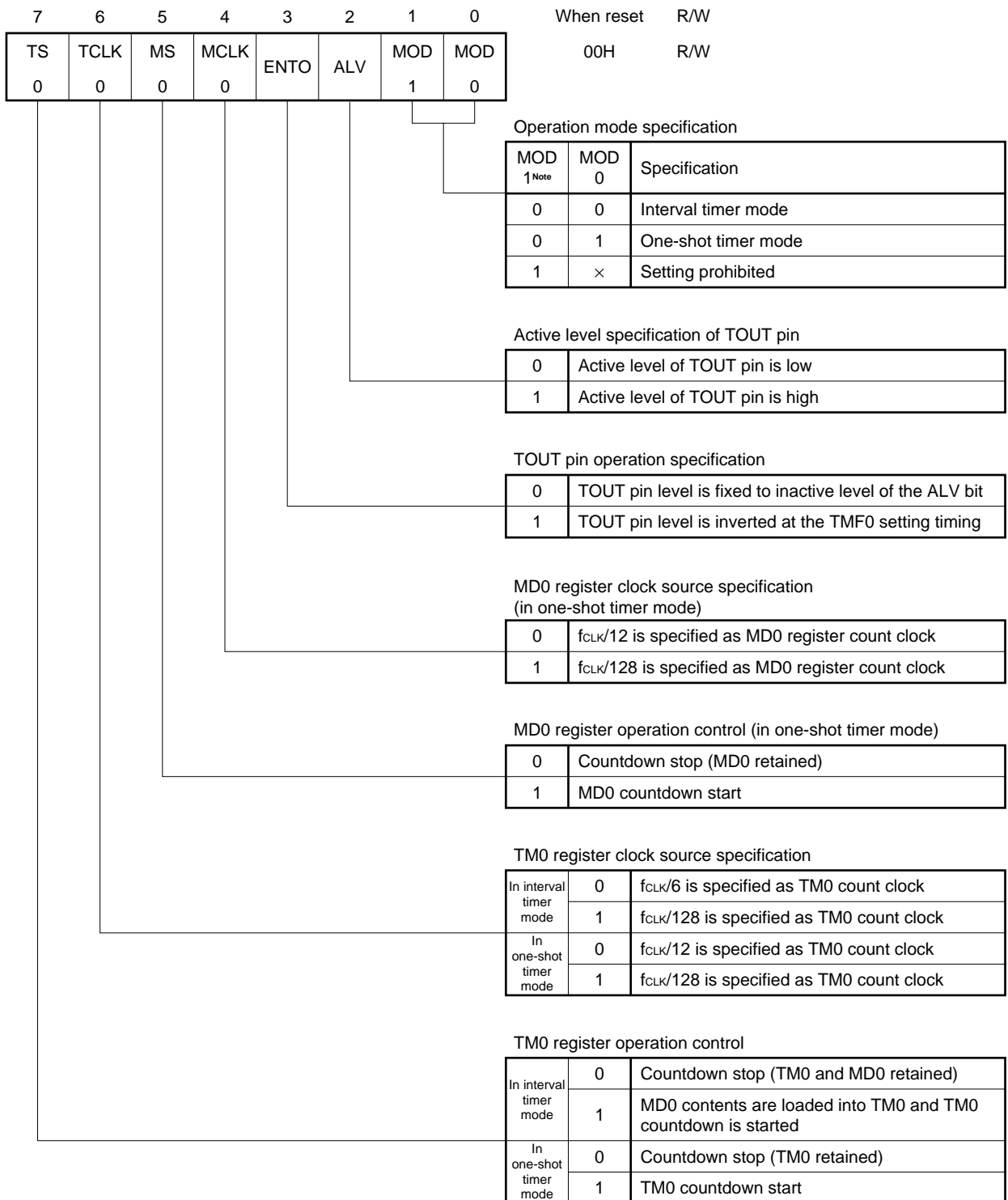
Table 9-2. TM0 and MD0 Count Time in One-shot Timer Mode

When $f_{CLK} = 8 \text{ MHz}$

TCLK0/ MCLK0	Count clock	Resolution	Full count
0	$f_{CLK}/12$	$1.5 \mu\text{s}$	98.3 ms
1	$f_{CLK}/128$	$16 \mu\text{s}$	1.04 s

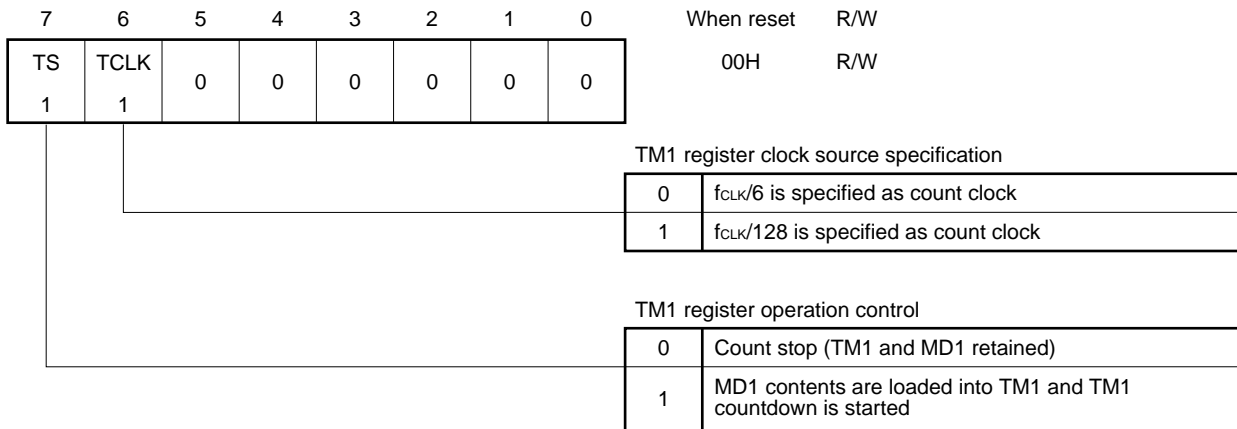
Caution The TM0 register count clock when the interval timer mode is set differs from that when the one-shot timer mode is set.

Figure 9-4. TMC0



Note Be sure to write “0” into MOD1.

Figure 9-5. TMC1



9.3 Timer Unit Interrupt Requests

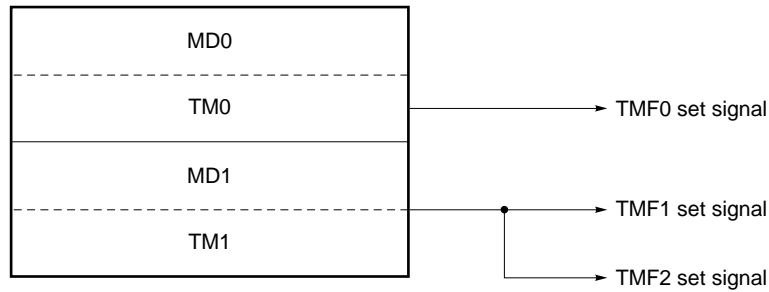
Three interrupt requests (TMF0 to TMF2) occur from the timer unit. Their occurrence conditions vary depending on the timer operation mode specification.

When the interval timer mode is set, TMF0 is set to 1 when the TM0 register value is set to 0 by counting down, and TMF1 and TMF2 are set to 1 when the TM1 register value is set to 0 by counting down. (See **Figure 9-6 (a)**.)

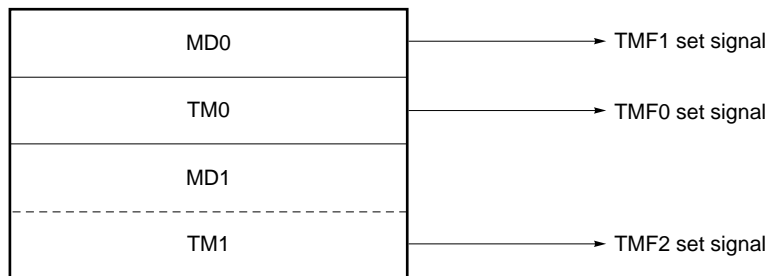
When the TM0 and MD0 registers are set to the one-shot timer mode, TMF0 is set to 1 when the TM0 register value is set to 0 by counting down, and TMF1 is set to 1 when the MD0 register value is set to 0 by counting down. In this case, TMF2 is set to 1 when the value of the TM1 register operating as interval timer changes to 0 by counting down.

Figure 9-6. Interrupt Requests Occurring from Timer Unit

(a) When TM0 and MD0 are set to interval timer mode



(b) When TM0 and MD0 are set to one-shot timer mode



TMF0 to TMF2: Timer unit interrupt request flags 0 to 2

9.3.1 Timer unit interrupt request control registers (TMIC0, TMIC1, and TMIC2)

The TMICn register (n = 0 to 2) is an 8-bit register that controls the corresponding three interrupt requests that occur from the timer unit. These three interrupt requests make up one group and the priority level is programmable as a timer unit interrupt request group. The priority levels in the group are fixed as follows.

TMF0>TMF1>TMF2

Figure 9-7. TMIC0, TMIC1, and TMIC2

	7	6	5	4	3	2	1	0
TMIC0	TMF0	TMMK0	MS/INT	ENCS	0	PR2	PR1	PR0
TMIC1	TMF1	TMMK1	MS/INT	ENCS	0	1	1	1
TMIC2	TMF2	TMMK2	MS/INT	ENCS	0	1	1	1

Caution Bits 2 to 0 of TMIC1 and TMIC2 are fixed to 1. The TMIC1 and TMIC2 interrupt request priority levels conform to the TMIC0 PR2 to PR0 settings.

For details of the TMICn register bits, see section 4.8 Interrupt Request Control Register.

The registers can be written/read by making an 8-bit or 1-bit memory access, in which case one wait state is inserted.

When $\overline{\text{RESET}}$ is input, the TMICn register contents are initialized to 47H.

9.3.2 Timer unit macro service control registers (TMMS0, TMMS1, and TMMS2)

The TMMS0 to TMMS2 registers are 8-bit registers that control macro service which is started when three types of interrupt requests occur from the timer unit.

The TMMS0 register is used to control the macro service that is started when the TMF0 flag is set.

The TMMS1 register is used to control the macro service that is started when the TMF1 flag is set, and the TMMS2 register is used to control the macro service started when the TMF2 flag is set.

The registers can be written/read by making an 8-bit or 1-bit memory access, in which case one wait state is inserted.

Figure 9-8. TMMS0, TMMS1, and TMMS2

	7	6	5	4	3	2	1	0
MSM	MSM	MSM	DIR	0	CH	CH	CH	
2	1	0			2	1	0	

For details of the TMMSn register bits, see section 4.5.4 Macro service control register.

CHAPTER 10 TIME BASE COUNTER

The V25 and V35 each contain a long interval timer function for its watch function.

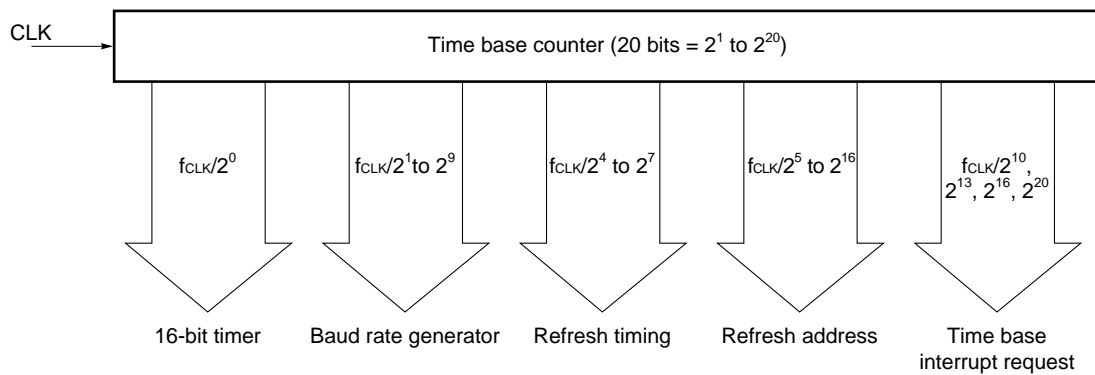
10.1 Time Base Counter Structure

Figure 10-1 shows the structure of the time base counter.

The time base counter consists of 20 frequency dividers that divide the system clock (CLK). The low-order part of the frequency divider tap output is used as a 16-bit timer count clock, a baud rate generator input clock, and for refresh timing generation and refresh address generation. Output taps at tap output bits 10, 13, 16, and 20 are used for time base interrupts.

When $\overline{\text{RESET}}$ is input, the time base counter contents are initialized to 00000H. After this, the time base counter continues to be incremented.

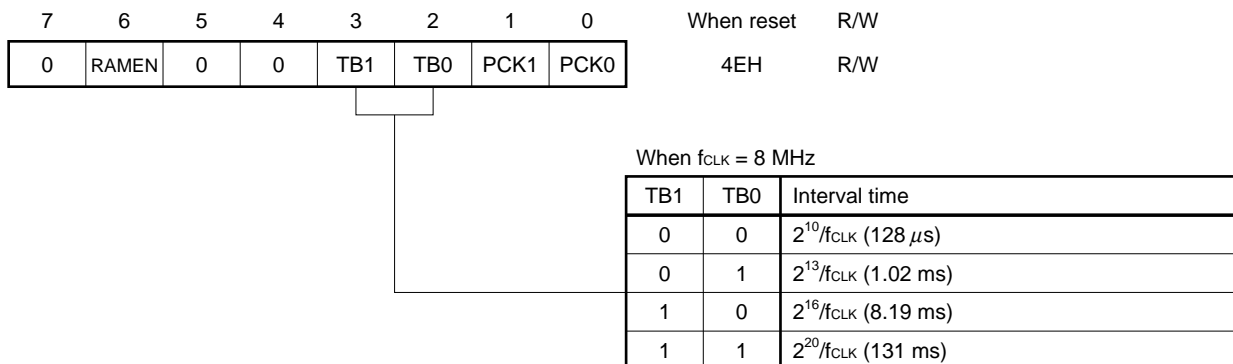
Figure 10-1. Time Base Counter Structure



10.2 Time Base Interval Specification

The interval time of an interrupt request occurring from the time base counter can be selected among the four types shown in Figure 10-2 by setting processor control register (PRC) bits 2 and 3 (TB0 and TB1).

Figure 10-2. PRC



Caution The time immediately following setting of the TB0 and TB1 bits and until the first interrupt request occurs is undefined.

10.3 Time Base Interrupt Request Control Register (TBIC)

The TBIC register is an 8-bit register for mask control of interrupt requests occurring from the time base counter.

The TBIC register can be written/read by making an 8-bit or 1-bit memory access, in which case one wait state is inserted.

When $\overline{\text{RESET}}$ is input, the TBIC contents are initialized to 47H.

Figure 10-3. TBIC

7	6	5	4	3	2	1	0	When reset	R/W
TBF	TBMK	0	0	0	1	1	1	47H	R/W

When the time base counter output tap specified in the processor control register (PRC) goes high, the interrupt request flag (TBF) is set to 1, causing an interrupt request to occur.

TBIC bits 4 and 5 are fixed to 0. The time base counter interrupts do not support the register bank switching function or macro service function. TBIC bits 0 to 2 are fixed to 1 and the time base interrupt (INTTB) priority level is fixed to 7. It is fixed to the lowest priority level even when another interrupt having priority level "7" occurs. However, the time base interrupt is subjected to multiple servicing control.

[MEMO]

CHAPTER 11 SERIAL INTERFACE

11.1 Serial Interface Configuration

The V25 and V35 each have two on-chip serial interface channels that contain dedicated baud rate generators.

The serial interface is a transfer system using start and stop bits. It provides two operation modes: the asynchronous mode, which uses a start bit for bit synchronization and character synchronization of data; and the I/O interface mode, which transfers data in synchronization with a controlled serial clock, as when using a serial data transfer system such as the μ PD7810.

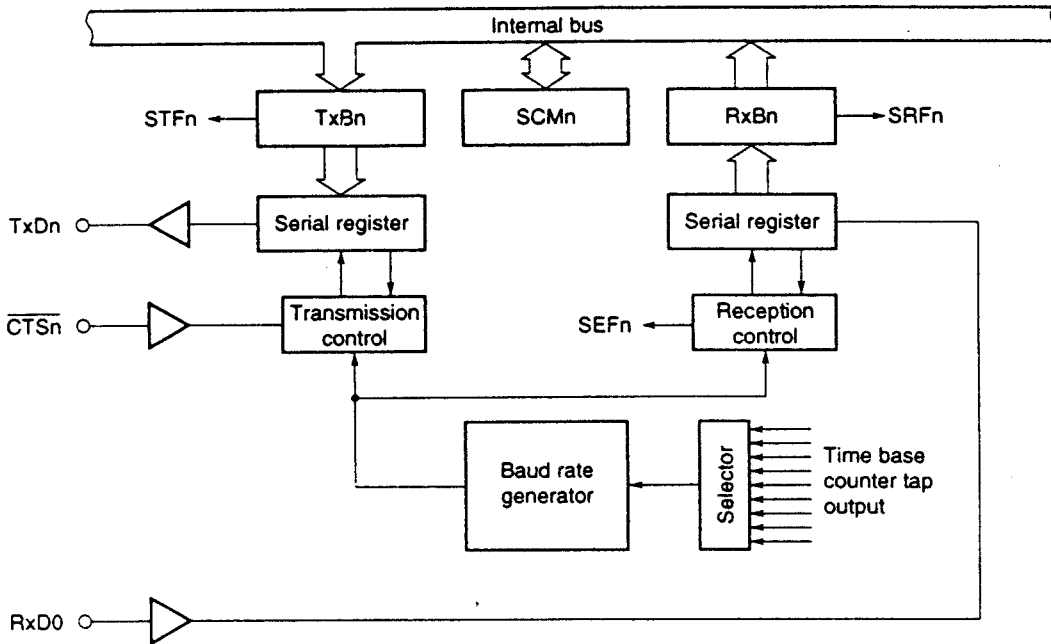
Figures 11-1 (a) and (b) show serial interface block diagrams for when the asynchronous mode and I/O interface mode are set, respectively.

The serial interface consists of a serial data input (RxDn), serial data output (TxDn), serial clock output ($\overline{\text{SCK0}}$), and clear to send ($\overline{\text{CTS}}$ n) pins^{Note}, a transfer control block, 8-bit serial registers for transmission and reception, a transmit buffer (TxBn), receive buffer (RxBn), and baud rate generator. Because serial registers and buffers are provided for independent transmission and reception, full duplex operation can be performed. In the I/O interface mode, the $\overline{\text{CTS}}$ n pin has the receive clock input/output pin function, and full duplex serial operation can also be performed in the I/O interface mode.

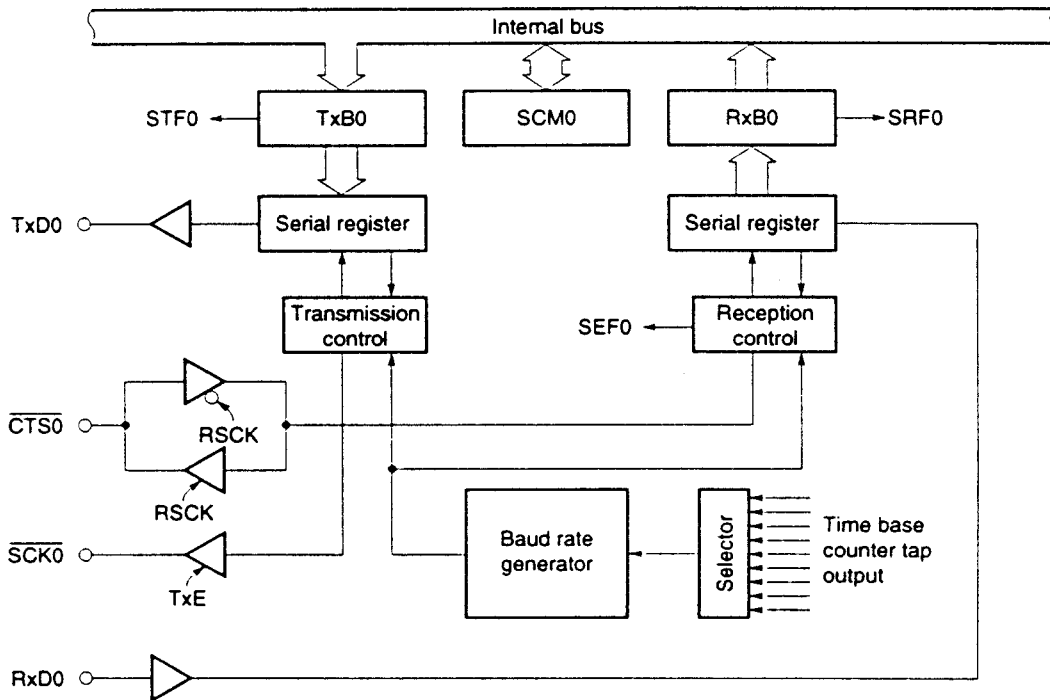
Note See section A.3 Limitations on Send Data Missing by Transmission Disable Operation during Serial Transmission.

Figure 11-1. Serial Interface Function

(a) When asynchronous mode is set ($n = 0$ or 1)



(b) When I/O interface mode is set (channel 0 only)



11.2 Asynchronous Mode

In the asynchronous mode, the serial mode register (SCMn) can control specification of the character length, number of stop bits, parity enable state, and odd or even parity.

11.2.1 Transmission

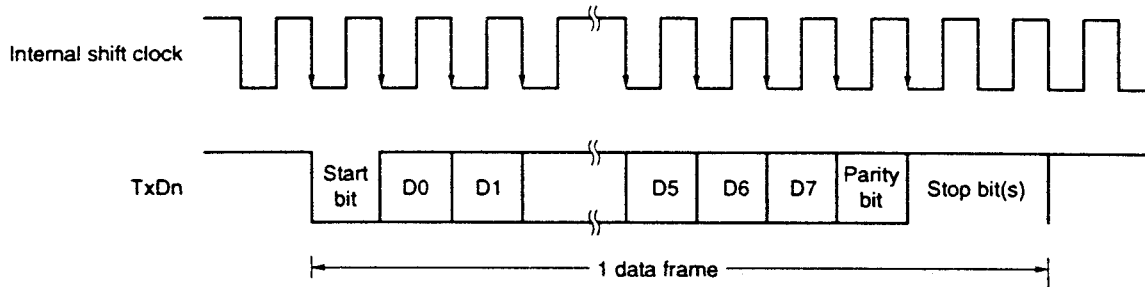
When serial mode register (SCMn) bit 7 (TxRDY) is set to 1 and the $\overline{\text{CTS}}_n$ pin is active (0), transmission is enabled^{Note}. Transmission can be started in any of the following three ways:

- (a) A transmission completion interrupt is generated by enabling transmission when the transmit buffer (TxB) is empty, and the interrupt routine writes send data into the transmit buffer.
- (b) If send data is transferred to the transmit buffer when transmission is enabled, the send data is transmitted consecutively after the immediately preceding transmission operation terminates.
- (c) If transmission is enabled after send data is prewritten into the transmit buffer in the transmission disable state, the data retained in the transmit buffer is transmitted.

Note Transmission may be enabled by either activating the $\overline{\text{CTS}}_n$ pin with TxRDY = 1 or by setting TxRDY to 1 with the $\overline{\text{CTS}}_n$ pin activated.

The send data format is shown below. A start bit, character bit, and a parity bit, and one or two stop bits make up one data frame (see Figure 11-2). Serial data is sent out in synchronization with the transmission clock's falling edge, starting at the least significant bit (LSB) from the TxDn pin. When transmission is disabled or when there is no data transmitted to the serial register, the TxDn pin becomes the mark state (1).

Figure 11-2. Format of Send Data



- Start bit : 1
- Character bit : 7 or 8
- Parity bit : Odd, even, 0, or no parity
- Stop bit(s) : One or two

11.2.2 Transmission completion interrupt

A transmission completion interrupt request occurs immediately when the transmit buffer (TxBn) becomes empty.

When $\overline{\text{RESET}}$ is input, the transmit buffer (TxBn) becomes empty. If transmission is enabled at this time, a transmission completion interrupt request occurs. When transmission operation is started and send data in the transmit buffer is transferred to the shift register, the transmit buffer becomes empty and a transmission completion interrupt request occurs.

Consecutive data transmission is enabled without inserting the mark state (1) by writing send data into the transmit buffer each time a transmission completion interrupt request occurs. When transmission is disabled, transfer from the transmit buffer to the shift register is also disabled.

If the transmission enable state is switched to disable during the transmission operation, the data being transmitted is transmitted up to the end of the current frame. However, if new send data is already written into the transmit buffer, transfer from the transmit buffer to the shift register is disabled and the transmit buffer contents are retained. When transmission is again enabled, the transmit buffer contents are transferred to the shift register in synchronization with this timing, and a transmission completion interrupt request occurs concurrently with the start of transmission.

Caution When a transmission completion interrupt is used in a macro service and while a macro service which will set macro service counter (MSC) to 0 is held pending for a long time, another macro service may be processed and the macro service completion interrupt may be acknowledged with the serial register and transmit buffer empty.

In this case, the TxRDY flag of SCM0 and SCM1 must be set from 1 to 0 then 1 (enable → disable → enable) within the macro service completion interrupt service routine. If this operation is not executed, subsequent transmission completion interrupts will not occur and the transmission operation will stop. (For example, this occurs when the interrupt servicing time is long for an interrupt having a higher priority level than the transmission interrupt.) However, all other operations will continue normally.

11.2.3 Reception

- ★ When serial mode register (SCMn) bit 6 (RxE) is set to 1, reception is enabled. When reception is disabled (RxE = 0), the reception hardware stands by in the initial state.

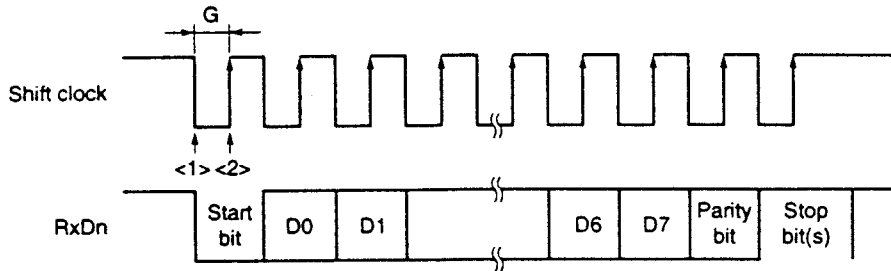
When RxDn pin input is sampled using the input clock to the baud rate generator and the falling edge is detected, reception operation starts and the baud rate generator for reception starts counting. If RxDn pin input is detected to be low with the first timing signal from the baud rate generator for reception, it is recognized as a start bit and subsequent reception operations are performed. If it is detected to be high with the first timing signal, it is not recognized as a start bit and the baud rate generator is initialized and stops operation.

Receive data is sampled in synchronization with the shift clock's rising edge after a start bit is detected (see Figure 11-3).

The start bit is detected by the input clock (selected by setting the SCC register) to the baud rate generator. The baud rate is generated by counting the clocks generated by dividing the input clocks by two, as many times as the value (G) set in the BRG register.

In other words, the start bit is detected when G is counted using the input clock's edge to the baud rate generator after the reception pin (RxD) is detected low when sampling the input clock to the baud rate generator. Once the start bit is detected, receive data is read each time 2G is counted by the input clock's edge to the baud rate generator.

Figure 11-3. Receive Data Sampling Timing



<1> Reception operation starts when the falling edge of the RxD pin is detected when sampling the input clock's edge to the baud rate generator.

<2> The baud rate generator for reception starts counting.

11.2.4 Reception completion interrupt

Upon reception completion of data having the character length specified in the serial mode register bit 3 (CL), receive data in the shift register is transferred to the receive buffer (RxBn) and a reception completion interrupt request is generated.

During reception, an odd-even parity check is made (when PRTY1 bit^{Note} = 1). When a mismatch is found (parity error), when the stop bit is low (framing error), or when the receive buffer is full and the next data is transferred to the receive buffer (overrun error), a reception error flag is set, causing a reception error interrupt request to occur. (See section 11.7).

Note The PRTY1 bit is serial mode register bit 5.

Caution If the RxE bit is cleared to set the reception disable state during reception, the character being received at the time is not guaranteed.

11.3 I/O Interface Mode

The I/O interface mode is the same as the serial interface mode of the μ PD7810, etc. It is useful for external I/O addition or connection of an I/O controller (such as an A/D converter or liquid crystal controller).

In the I/O interface mode, data is transferred starting at the most significant bit (MSB) with the character length fixed to eight bits and with no parity bit.

The I/O interface mode can be used only with channel 0.

11.3.1 Transmission

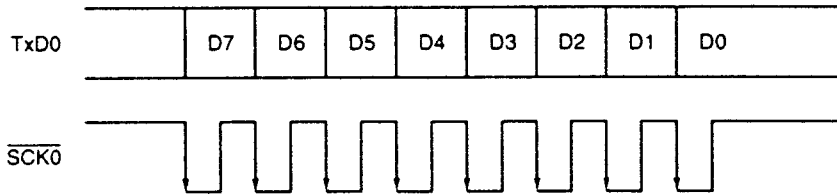
When serial mode register bit 7 (TxE) is set to 1, transmission is enabled. In the I/O interface mode, the $\overline{\text{SCK0}}$ pin is used as a transmit clock output pin.

The transmit clock cannot be externally input.

As with the asynchronous mode, transmission operation in the I/O interface mode can be started in any of the following three ways:

- (a) A transmission completion interrupt request is generated by enabling transmission when the transmit buffer (TxB0) is empty, and the interrupt service routine writes send data into the transmit buffer.
- (b) If send data is transferred to the transmit buffer (TxB0) when transmission is enabled, the send data is transmitted consecutively after the immediately preceding transmission operation terminates.
- (c) If transmission is enabled after send data is prewritten into the transmit buffer (TxB0) in the transmission disable state, the data retained in the transmit buffer (TxB0) is transmitted.

The send data format is shown below. The data is transmitted starting at the most significant bit (MSB) with the character length fixed to eight bits.



11.3.2 Transmission completion interrupt

A transmission completion interrupt request occurs immediately when the transmit buffer (TxB0) becomes empty.

When $\overline{\text{RESET}}$ is input, the transmit buffer (TxB0) becomes empty. If transmission is enabled at this time, a transmission completion interrupt request occurs. When transmission operation starts and send data in the transmit buffer (TxB0) is transferred to the shift register, the transmit buffer becomes empty and a transmission completion interrupt request occurs.

Caution When a transmission completion interrupt is used in a macro service and while a macro service which will set macro service counter (MSC) to 0 is held pending for a long time, another macro service may be processed and the macro service completion interrupt may be acknowledged with the serial register and transmit buffer empty.

In this case, the TxRDY flag of SCM0 and SCM1 must be set from 1 to 0 then 1 (enable \rightarrow disable \rightarrow enable) within the macro service completion interrupt service routine. If this operation is not executed, subsequent transmission completion interrupts will not occur and the transmission operation will stop. (For example, this occurs when the interrupt servicing time is long for an interrupt having a higher priority level than the transmission interrupt.) However, all other operations will continue normally.

11.3.3 Reception

When serial mode register (SCM0) bit 6 (RxE) is set to 1, reception is enabled. Receive data is input to the serial register on the reception clock's rising edge. When the serial register receives 8-bit data, the data is transferred from the serial register to the receive buffer (RxB0) and a reception completion interrupt request is generated.

During I/O interface mode, the reception clock can be selected as either an external reception clock or an internal reception clock by setting the serial mode register (SCM0) bit 2 (RSCK).

When using an internal reception clock, the operation is started by writing a value of 1 into the SCM0 TSK bit. This means that the macro service function cannot be used.

During I/O interface mode, the $\overline{\text{CTS0}}$ pin functions as a reception clock input/output pin.

Caution If the RxE bit is cleared to set the reception disable state during reception, the character being received at the time is not guaranteed.

11.3.4 Reception error interrupt

During reception, if the receive buffer (RxB0) becomes full and the next data is transferred to the receive buffer (overrun error), a reception error flag is set, causing a reception error interrupt request to occur.

11.3.5 Serial register clear

To clear the serial register, perform either of the following operations.

- Write into the SCC0 register
- Change the SCM0 register's MD0 bit
(I/O interface mode → asynchronous mode → I/O interface mode)

11.4 Starting Transmission without Using Interrupts

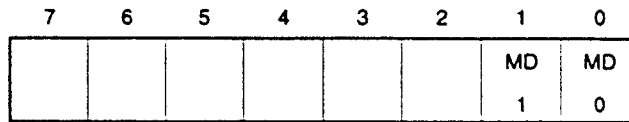
Transmission can be started without using interrupts in the following two ways.

- (1) If the transmission enable state is set while the transmit buffer (TxB) is empty, the interrupt control register (STICn) bit 7 (STFn) will be set to 1, and should be cleared to 0 once. Next, the send data is written into the transmit buffer, and then the STICn bit is polled to start transmission.
- (2) After the first byte of data is written into the transmit buffer (TxB) during the transmission disable state, setting the transmission enable state causes the data retained in the transmit buffer to be downloaded to the serial register, from where the data is output.

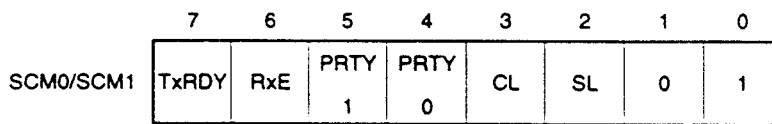
Remark The STICn register's bit 7 (STFn) is not cleared to 0 until a transmission completion interrupt is acknowledged. Accordingly, when not using interrupts, use software to clear the STFn flag to 0 when writing data to the transmit buffer.

11.5 Serial Mode Registers (SCM0 and SCM1)

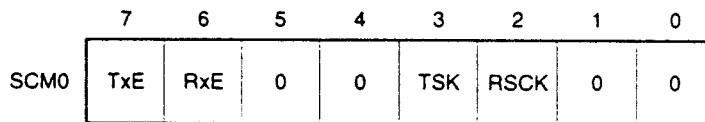
The serial mode registers (SCMn, n = 0 or 1) are 8-bit registers that specify the serial interface transfer mode. The SCM0 register can be set for channel 0 and the SCM1 register for channel 1. The meanings of SCMn bits 2 to 7 vary depending on how bits 1 and 0 (MD1 and MD0) are set.



MD1, MD0 = 0, 1 (asynchronous mode)



MD1, MD0 = 0, 0 (I/O interface mode)



The MD1 and MD0 bits are serial interface transfer mode specification bits. The asynchronous mode is selected by setting MD1 = 0 and MD0 = 1. The I/O interface mode is selected by setting MD1 = 0 and MD0 = 0. However, the I/O interface mode can be set only in SCM0.

The registers can be written/read by making an 8-bit or 1-bit memory access.

When $\overline{\text{RESET}}$ is input, the register contents are reset to 00H.

During serial transmission, transfer of send data from the transmit buffer (TxB) to the serial register causes an interrupt request flag (STFn) to be set, after which data is sent to the TxD pin one bit at a time.

This means that if the serial mode register (SCMn) is used during serial transmission and immediately after STFn has been set, subsequent transmission operations cannot be performed normally.

Therefore, be sure to allow sufficient time for the transmission operation after data is set into the serial mode register (SCMn).

(1) When asynchronous mode is set

RxE : Reception enable control bit

When RxE is set to 0 (reception disable) during the reception operation, reception processing is stopped and no reception completion interrupt occurs.

SL : Bit specifying the number of stop bits

When the SL bit is reset to 0, one stop bit is specified; when the bit is set to 1, two stop bits are specified.

CL : Character length specification bit

When the CL bit is reset to 0, the character length is set to seven bits; when the bit is set to 1, the character length is set to eight bits.

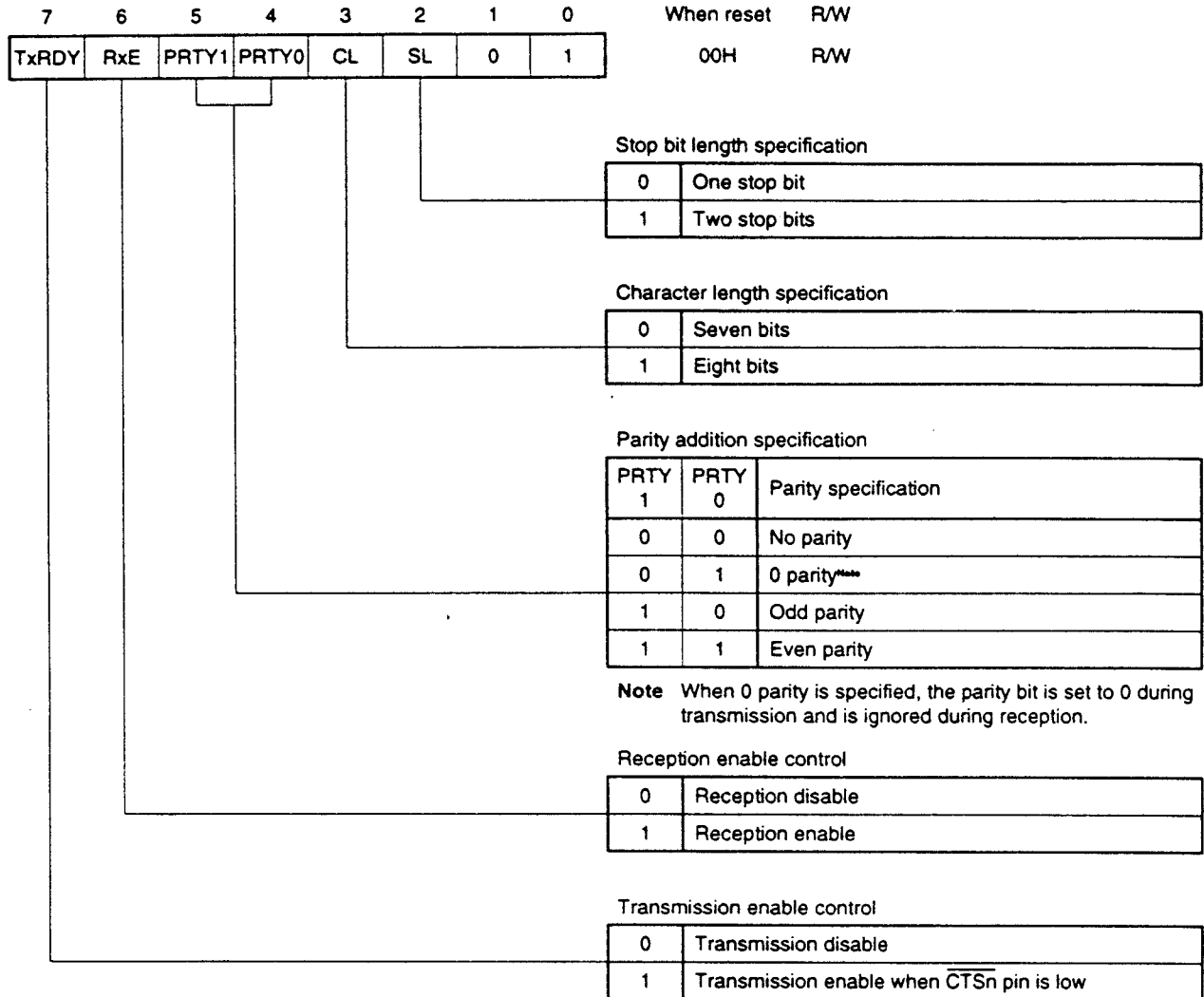
PRTY0 and **PRTY1** : Parity addition specification bits

No parity, odd parity, even parity or 0 parity is specified by setting the PRTY0 and PRTY1 bits. When 0 parity is specified, the parity bit is set to 0 during transmission and is ignored during reception.

TxRDY : Transmission enable state control bit

When the $\overline{\text{CTS}}_n$ pin is low and TxRDY is set to 1, transmission is enabled.

Figure 11-4. SCM0 and SCM1 (when asynchronous mode is set)



(2) When I/O interface mode is set

RSCK : Serial reception clock source specification bit

When the RSCK bit is reset to 0, the external reception clock is selected for the reception operation; when the bit is set to 1, the internal reception clock is selected for the reception operation. The $\overline{CTS_0}$ pin is used as the reception clock input/output pin.

TSK : Reception clock output trigger bit

This bit is valid only when the RSCK bit is set to 1. When 1 is written into the TSK bit, eight reception shift clocks are output from the $\overline{CTS_0}$ pin.

When serial clock is output, the TSK bit is automatically reset to 0.

RxE : Reception enable control bit

When RxE is set to 1, reception is enabled; when the bit is reset to 0, reception is disabled. If the bit is reset to 0 (reception disable) during the reception operation, reception processing is stopped and no reception completion interrupt request occurs.

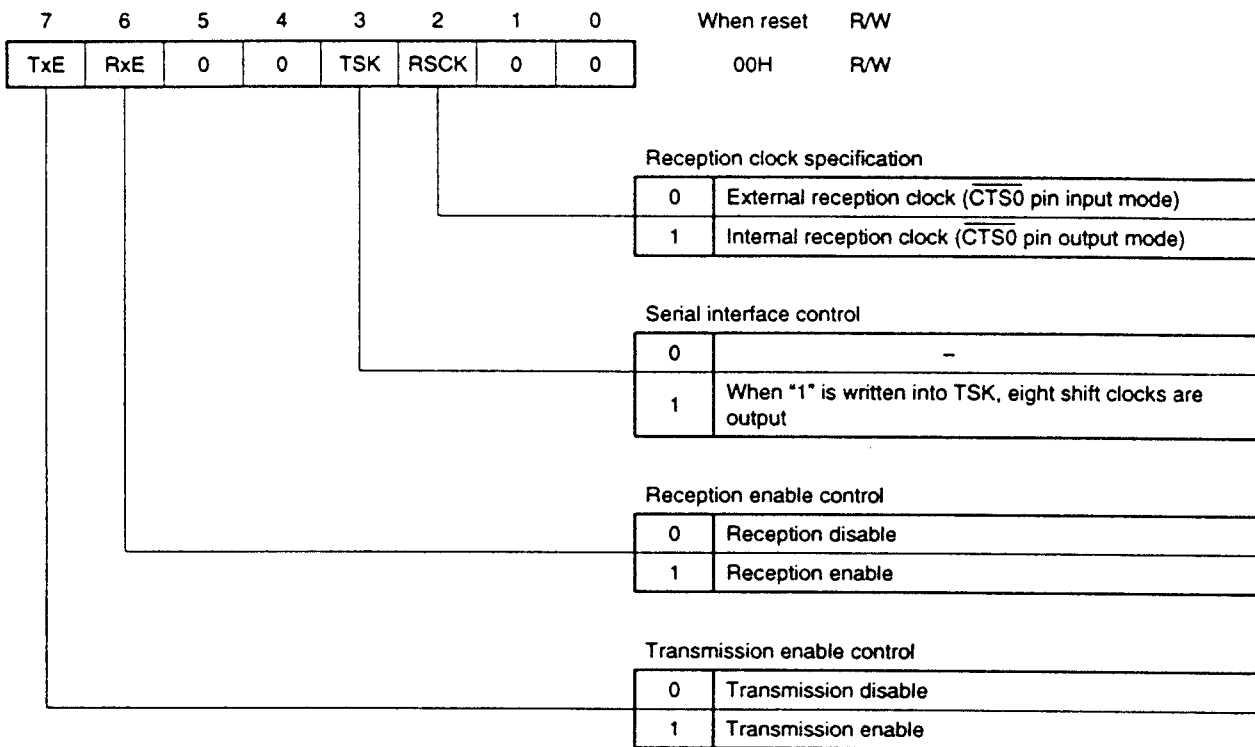
TxE : Transmission enable control bit

When the TxE bit is set to 1, transmission is enabled, when the bit is reset to 0, transmission is disabled. If transfer data is written into the transmit buffer when transmission is enabled (TxE = 1), the corresponding serial transmission is started after the transmission being performed or immediately if a transmission is not being performed.

If send data is written into the transmit buffer when transmission is disabled (TxE = 0), serial transmission is not performed and the data in the transmit buffer is retained. Subsequently, transmission processing of the send data held in the transmit buffer is started at the same time as transmission is enabled.

Although the TxE bit is reset to 0 (transmission disable) during the transmission operation, the transmission operation is performed to the end. However, if the next send data is already stored in the transmit buffer when transmission is disabled by resetting TxE to 0, the next send data transmission is held pending and the data is retained in the transmit buffer.

Figure 11-5. SCM0 (when I/O interface mode is set)



11.6 Baud Rate Generators

The baud rate generators are 8-bit timers dedicated to the serial interface that generates shift clocks for transmission and reception. Transmission and reception baud rate generators are provided for each channel. The transmission baud rate is the same as the reception baud rate. The baud rate is determined by writing a value to the baud rate generator register (BRGn). However, the maximum baud rate is 750 kbps. Data transfer at more than 750 kbps is enabled by inserting idle time of one shift clock or more between data units.

The input clock to the baud rate generator is specified by selecting a time base counter output tap (see section 10.1) in serial control register (SCCn) bits PRS3 to PRS0. The baud rate generator output signal is used for the serial interface shift clock. The relationship between the baud rate and parameters is shown below.

$$B \cdot G = 10^6 \times \frac{f_{CLK}}{2^{n+1}}$$

These parameters are defined as follows.

B : Transfer baud rate (bps)

B=110, 150,, 9600, 19200,

G : Value set in baud rate generator register (BRGn) ($2 \leq G \leq 255$)

n : Input clock specification number to baud rate generator specified in serial control register ($0 \leq n \leq 8$)

CLK : System clock frequency (MHz)

The two modes are described as follows.

- When in I/O interface mode (clock synchronization)

No baud rate error is allowed. The data timing is determined by the setup and hold times corresponding to the transmission and reception clock's rising edge.

- During asynchronous mode (asynchronously)

The baud rate is defined as $(1/G) \times 10^6 \times f_{CLK}/2^{n+1}$ and the baud rate error depends on the f_{CLK} value.

When the setup value G is small, the effect of f_{CLK} 's frequency deviation becomes greater. Therefore, the baud rate's error tolerance is reduced as the setup value G becomes smaller.

Table 11-1 lists the baud rate generator setup values for each standard baud rate when an external 16-MHz crystal is used (when $f_{CLK} = f \times 12$).

Table 11-1. Baud Rate Generator Setup Values (for reference)

When $f_{CLK} = 8 \text{ MHz}$

Baud rate	n	BRGn register setup value G	Error (%)
110	8	142	0.03
150	7	208	0.16
300	6	208	0.16
600	5	208	0.16
1200	4	208	0.16
2400	3	208	0.16
4800	2	208	0.16
9600	1	208	0.16
19200	0	208	0.16
38400	0	104	0.16

n: Specification number of input clock to baud rate generator

The baud rate error is calculated as follows.

$$\sqrt{\left\{ \frac{\left(\text{Baud rate calculated based on parameters} \right) - \left(\text{Requested baud rate value} \right)}{\text{Baud rate value for requested value.}} \right\}^2} \times 100 (\%)$$

The baud rate error values listed in Table 11-1 are errors corresponding to the requested values for parameter-based settings.

11.6.1 Serial control registers (SCC0 and SCC1)

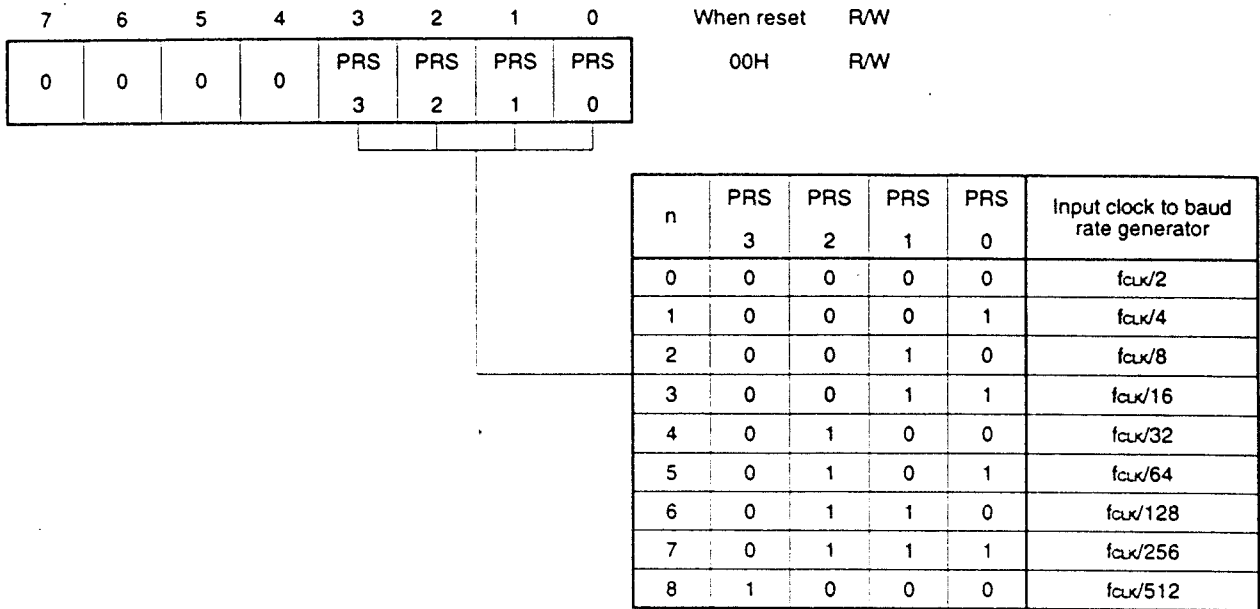
The serial control registers (SCC0 and SCC1) are serial interface transfer rate control registers.

The registers can be written/read by making an 8-bit or 1-bit memory access.

When RESET is input, the register contents are initialized to 00H.

Time base counter output tap input to the baud rate generator is specified by setting bits PRS3 to PRS0.

Figure 11-6. SCC0 and SCC1



f_{clk} : System clock frequency

n : Specification number of input clock to baud rate generator

11.7 Serial Error Handling

The following three types of serial interface reception errors can be detected.

(a) Parity error (in asynchronous mode)

The parity operation result does not match the reception parity.

(b) Framing error (in asynchronous mode)

Stop bit is not detected.

(c) Overrun error (in asynchronous or I/O interface mode)

Before the preceding receive data is received from RxB, the next reception is complete.

11.7.1 Serial error registers (SCE0 and SCE1)

The serial error registers are 8-bit registers that indicate three types of error flag states corresponding to reception errors. These registers are provided for both channel 0 and channel 1.

These registers can be only read by making an 8-bit memory access.

The error flags are updated when the next data reception terminates. The previous flag contents are retained until then.

When $\overline{\text{RESET}}$ is input, the SCE_n contents are initialized to 00H.

ERP_n : Parity error flag

When transmission parity does not match reception parity, the ERP flag is set to 1. If they match at the next data reception, the flag is reset to 0.

ERF_n : Framing error flag

When a stop bit is not detected, the ERF flag is set to 1. If a stop bit is detected at the next data reception, the flag is reset to 0.

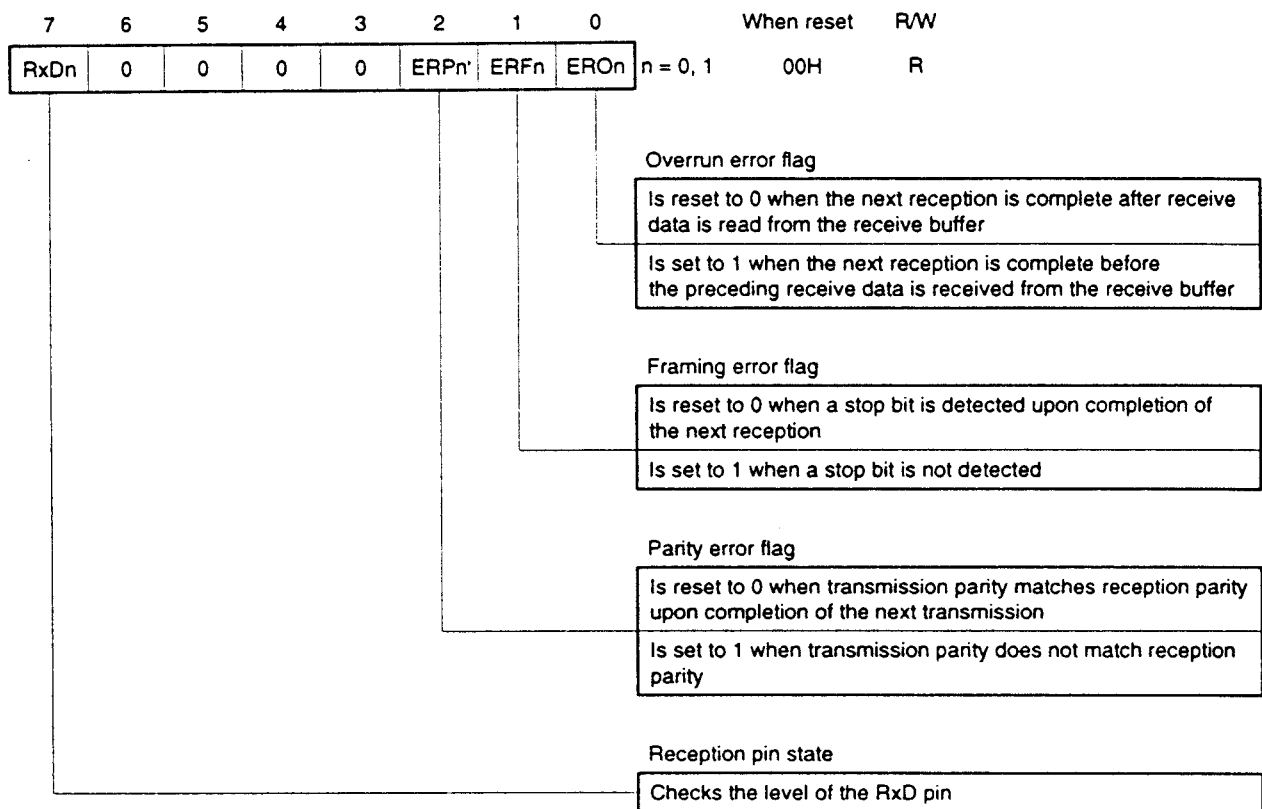
EROn : Overrun error flag

When the next reception is completed before the preceding receive data is received from the receive buffer (RxB), the ERO flag is set to 1. When data reception terminates after receive data is read from the receive buffer (RxB), the flag is reset to 0.

RxD_n : Bit for checking the RxD pin for reception state

The RxD pin level is set in this bit.

Figure 11-7. SCE0 and SCE1



11.8 Break State Detection Function

The V25 and V35 can detect the line break state by software (only in the asynchronous mode). The break state detection procedure is described below.

(1) Reception error interrupt caused by the first framing error

The reception error handling routine checks receive data to ensure that it is 00H. At the same time, the routine checks the reception error flag for framing errors.

(2) Reception error interrupt caused by the second framing error

During the break state, a framing error occurs again.

The line's break state can be determined by the fact that the receive data is again 00H and consecutive 00H data is received with a framing error, and by directly checking the pin state in serial error register (SCEn) bit 7 (RxDn).

11.9 Serial Interface Interrupt Requests

Three types of interrupt requests that occur from the serial interface for both channels (0 and 1) are transmission completion, reception completion, and reception error interrupt requests.

11.9.1 Interrupt request control registers (SEICn, SRICn, and STICn: n = 0 or 1)

The interrupt request control registers are 8-bit registers that control reception error interrupt requests (SEFn), reception completion interrupt requests (SRFn), and transmission completion interrupt requests (STFn); all of which occur from the serial interface.

The three interrupt request control registers make up one group and the priority level can be specified as the serial interface interrupt request group. The priority levels within the group are fixed as follows.

$$SEFn > SRFn > STFn$$

When a reception error occurs, a reception error interrupt request is acknowledged preferentially because it has a higher priority level than a reception completion interrupt. When reception error interrupt servicing terminates, the reception completion interrupt is acknowledged.

Figure 11-8. SEICn, SRICn, and STICn (n = 0 or 1)

	7	6	5	4	3	2	1	0
SEIC0/SEIC1	SEFn	SEMKn	MS/INT	ENCs	0	PR2	PR1	PR0
SRIC0/SRIC1	SRFn	SRMKn	MS/INT	ENCs	0	1	1	1
STIC0/STIC1	STFn	STMKn	MS/INT	ENCs	0	1	1	1

Caution Bits 2 to 0 of SRICn and STICn are fixed to 1. The SRICn and STICn interrupt request priority levels conform to the settings for SEICn bits PR2 to PR0.

The SEFn, SRFn, and STFn bits are interrupt request flags. When a reception error occurs or when reception or transmission is completed, the SEFn, SRFn, or STFn bit is set to 1. It is reset to 0 by the software or when the corresponding interrupt request is acknowledged.

See section 4.8 **Interrupt Request Control Register** for descriptions of the other bits.

The registers can be written/read by making an 8-bit or 1-bit memory access, in which case one wait state is inserted.

When RESET is input, the contents of SEICn, SRICn, and STICn are initialized to 47H.

11.9.2 Macro service control registers (SRMSn and STMSn: n = 0 or 1)

The SRMSn register is an 8-bit register that specifies the macro service servicing mode and the channel used when serial interface reception is complete. The STMSn register is an 8-bit register that specifies the macro service servicing mode and the channel used when serial interface transmission is complete. The SRMS0 and STMS0 registers correspond to serial interface channel 0 and the SRMS1 and STMS1 registers correspond to serial interface channel 1.

The registers can be written/read by making an 8-bit or 1-bit memory access, in which case one wait state is inserted.

See section 4.5.4 Macro service control register for descriptions of the macro service control register bits.

Figure 11-9. SRMSn and STMSn (n = 0 or 1)

7	6	5	4	3	2	1	0
MSM2	MSM1	MSM0	DIR	0	CH2	CH1	CH0

CHAPTER 12 STANDBY FUNCTION

The V25 and V35 each contain the following two operation clock control modes for the standby function intended for low power consumption.

- HALT mode Stops CPU operation clock only. The CPU status and the data contents and internal RAM contents are all retained. Peripheral hardware operation is continued.
Total system power consumption can be lowered by intermittent operation using the HALT mode in combination with the normal operation mode.
- STOP mode ... Stops the oscillator and the entire internal circuitry.
The internal RAM contents and output data on ports are retained by consuming very low power.

The HALT or STOP mode is set by executing the HALT or STOP instruction.

12.1 Standby Control Register (STBC)

The standby control register (STBC) is an 8-bit register which contains a standby flag (SBF). The high-order seven bits are fixed to 0.

The SBF flag is used to decide a return from the STOP mode.

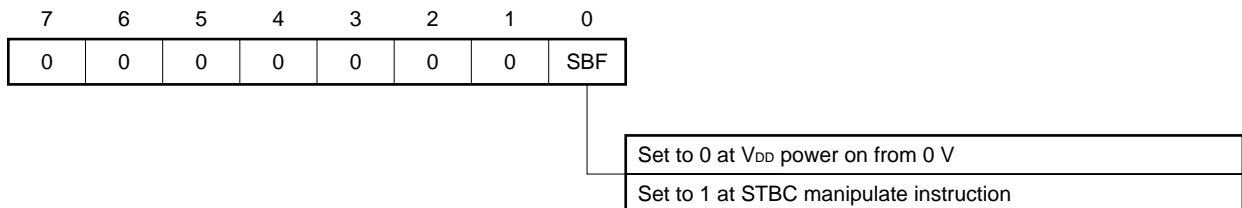
The SBF flag is set to 1 only by executing an instruction. The flag is cleared (to 0) only when the supply voltage (V_{DD}) rises from 0 V; it is not cleared by executing any instruction.

Therefore, the SBF flag can be tested to determine a reset start when the power is turned on (SBF = 0) or a return from the STOP mode (SBF = 1 **Note**).

The STBC register is not initialized when the $\overline{\text{RESET}}$ signal is input.

Note Preset the SBF flag to 1 before entering the STOP mode.

Figure 12-1. STBC



12.2 HALT Mode

The HALT mode stops the CPU operation clock.

The entire system's power consumption can be reduced by setting the HALT mode at the CPU idle time. The HALT mode is set by executing the HALT instruction.

In the HALT mode, the CPU clock stops and program execution is stopped, but the contents of all registers and on-chip RAM just before the HALT mode is entered are retained. Table 12-2 lists the hardware states. Also, see section **A.5 Limitations on Instruction Slips during HALT Mode**.

12.2.1 HALT mode release

The HALT mode is released when a nonmaskable interrupt (NMI) or an unmasked maskable interrupt request occurs or when $\overline{\text{RESET}}$ is input. (See **Figure 12-2**.)

If an unmasked interrupt request, macro service request, or DMA request occurs just after the transition to the HALT state is made by executing the HALT instruction, the instruction that immediately follows the HALT instruction is executed. Therefore, place one or more NOP instructions immediately after the HALT instruction.

When a macro service request or DMA request occurs, the HALT mode is released and macro service or DMA processing is started (see **Figure 12-3**.) When the macro service or DMA processing terminates, a return is again made to the HALT mode. However, the HALT mode is released when the condition described in Table 12-1 arises during the macro service or DMA processing.

(1) HALT mode release when an interrupt occurs

(a) When interrupt service routine sets HALT mode

The HALT mode is released when a nonmaskable interrupt request or an unmasked maskable interrupt request having a higher priority level than the interrupt being serviced occurs.

(b) Other than (a)

The HALT mode is released when a nonmaskable interrupt request or an unmasked maskable interrupt request occurs regardless of the priority level.

(2) HALT mode release when $\overline{\text{RESET}}$ is input

Same as the normal reset operation.

Figure 12-2. HALT Mode Release when Interrupt Request Occurs

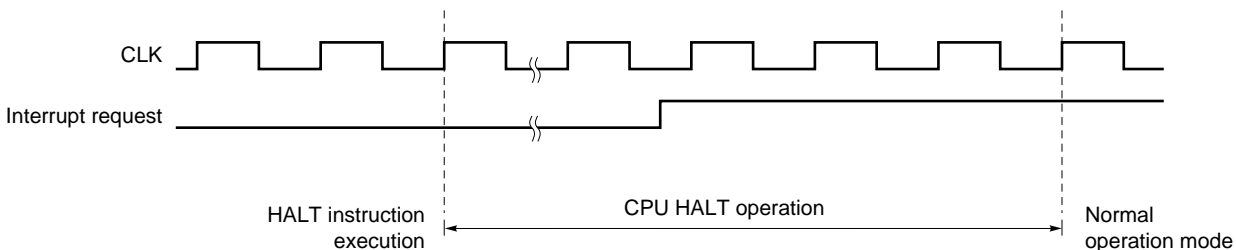


Figure 12-3. Macro Service or DMA Start during HALT Mode

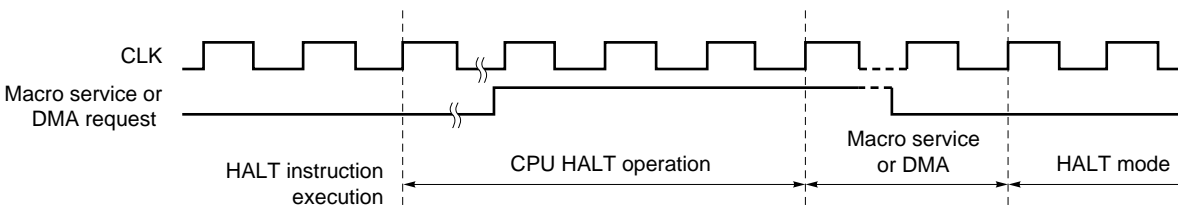


Table 12-1. Operation after HALT Mode Is Released when Interrupt Request Occurs

Release source	EI state	DI state
Nonmaskable interrupt request	A branch is taken to the vector address after the HALT mode is released.	A branch is taken to the vector address after the HALT mode is released.
Maskable interrupt request	A branch is taken to the vector address after the HALT mode is released.	The next instruction is executed after the HALT mode is released.
Macro service request	If the macro service is started and the macro service counter is set to 0H or a transfer data match is found in the character search mode, a branch is taken to the vector address. If the macro service counter is not set to 0H or a transfer data match is not found in the character search mode, the HALT mode is entered again.	If the macro service is started and the macro service counter is set to 0H or a transfer data match is found in the character search mode, the HALT mode is released and the next instruction is executed.
DMA request	If DMA is started and the terminal counter is set to 0H, a branch is taken to the vector address. If the terminal counter is not set to 0H, the HALT mode is entered again.	If DMA is started and the terminal counter is set to 0H, the HALT mode is released and the next instruction is executed.

12.3 STOP Mode

The STOP mode stops the oscillator.

The STOP mode is useful when the entire application system stops. This mode enables very low power consumption. The STOP mode is set by executing the STOP instruction. In the STOP mode, all clocks stop. Although program execution is stopped, the contents of all registers and on-chip RAM just before the STOP mode is set are retained. Table 12-2 lists the hardware states.

Caution In the STOP mode, the X1 and X2 pin levels are fixed. Therefore, do not use the STOP mode when using an external clock. Use a crystal or ceramic resonator when using the STOP mode.

12.3.1 STOP mode release

The STOP mode is released when an NMI request occurs or when $\overline{\text{RESET}}$ is input.

(1) STOP mode release when NMI request occurs (Figure 12-4)

When a valid edge is input to the NMI pin, the oscillator restarts oscillation. The time base counter (TBC) also starts operation. After the STOP mode is released, the clock is not immediately supplied. Instead, clock supply is started after the oscillation stabilization time is counted by the TBC. To count the oscillation stabilization time, the TBC uses a half of the interval time specified in the processor control register (PRC) bits 2 and 3 (TB0 and TB1). (However, any interrupt request occurring from the TBC at the time is disabled.) Set half of the interval time as 30 ms or more.

(2) STOP mode release when $\overline{\text{RESET}}$ is input

Same as normal reset operation.

Figure 12-4. STOP Mode Release by Input to NMI Pin

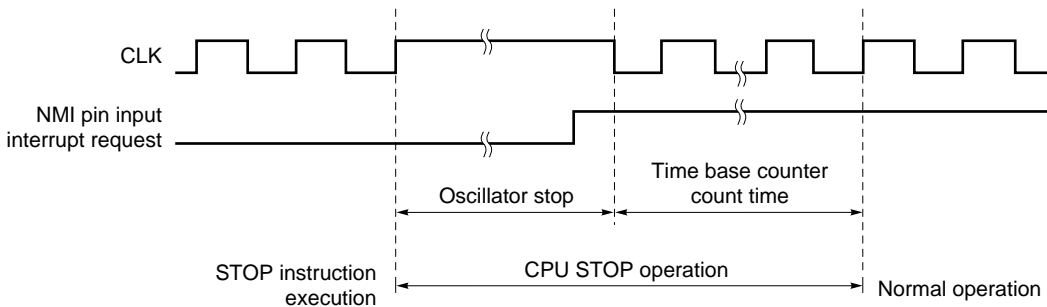


Table 12-2. HALT Mode and STOP Mode

Item		HALT mode	STOP mode
Oscillator		Operation	Stop
Internal system clock		Stop	
16-bit timer		Operation	
Time base counter			
HOLD circuit			
Serial interface			
Interrupt request controller			
DMA controller			
I/O line		Retention	Retention
Bus line	A0 to A19	Retention	Retention
	D0 to D15	High impedance	High impedance
R/W output		High	High
Refresh operation		Operation or stop	Stop
Data retention		All internal data, such as CPU status and RAM contents, is retained.	All internal data, such as CPU status and RAM contents, is retained.
Release method		<ul style="list-style-type: none"> • Nonmaskable interrupt request • Maskable interrupt request • $\overline{\text{RESET}}$ input • Macro service request^{Note} • DMA^{Note} 	<ul style="list-style-type: none"> • Nonmaskable interrupt request • $\overline{\text{RESET}}$ input

Note After the macro service or DMA processing terminates, a return is again made to the HALT mode.

[MEMO]

CHAPTER 13 RESET FUNCTION

When a low pulse is input to the $\overline{\text{RESET}}$ input pin, the system is reset and the hardware devices are placed in the state listed in Table 13-1. When the low-to-high transition of the $\overline{\text{RESET}}$ input is made, the reset state is released and program execution is started. Initialize the register contents within the program as required.

Table 13-1. Hardware States after Reset (1/2)

Hardware (symbol)		Address ^{Note} (Low-order 12 bits: xx□□□H)	State after reset	
Program counter		PC	—	0000H
Program status word		PSW	—	F002H
On-chip RAM	Data memory	—	—	Undefined
	General purpose registers	AW, CW, DW, BW, SP, BP, IX, IY	EFEH to EF0H	Undefined
	Segment registers (bank 7 only)	DS1, SS, DS0	EEEH, EEAH, EE8H	0000H
PS		EECH	FFFFH	
Ports	Port registers	P0, P1, P2	F00H, F08H, F10H	Undefined
		PT	F38H	
	Port mode registers	PM0, PM1, PM2	F01H, F09H, F11H	FFH
		PMT	F3BH	00H
Port mode control registers	PMC0, PMC1, PMC2	F02H, F0AH, F12H	00H	
Timer unit	Timer registers	TM0, TM1	F80H, F88H	Undefined
	Modulo/timer registers	MD0, MD1	F82H, F8AH	Undefined
	Timer control registers	TMC0, TMC1	F90H, F91H	00H
	Interrupt request control registers	TMIC0 to TMIC2	F9CH to F9EH	47H
	Macro service control registers	TMMS0 to TMMS2	F94H to F96H	Undefined
DMA controller	DMA mode registers	DMAM0, DMAM1	FA1H, FA3H	00H
	DMA control registers	DMAC0, DMAC1	FA0H, FA2H	Undefined
	Interrupt request control registers	DIC0, DIC1	FACH, FADH	47H
Serial interface	Serial mode registers	SCM0, SCM1	F68H, F78H	00H
	Serial control registers	SCC0, SCC1	F69H, F79H	00H
	Baud rate generator register	BRG0, BRG1	F6AH, F7AH	00H
	Receive buffer registers	RxB0, RxB1	F60H, F70H	Undefined

Note xx in the high-order eight bits of an address is a value specified in the IDB register.

Table 13-1. Hardware States after Reset (2/2)

Hardware (symbol)		Address ^{Note 1} (Low-order 12 bits: xx□□□H)	State after reset	
Serial interface	Transmit buffer registers	TxB0, TxB1	F62H, F72H	Undefined
	Serial error registers	SCE0, SCE1	F6BH, F7BH	00H
	Interrupt request control registers	(Error) SEIC0, SEIC1	F6CH, F7CH	47H
		(Reception) SRIC0, SRIC1	F6DH, F7DH	
		(Transmission) STIC0, STIC1	F6EH, F7EH	
	Macro service control registers	(Reception) SRMS0, SRMS1	F65H, F75H	Undefined
(Transmission) STMS0, STMS1		F66H, F76H		
Timer base interrupt request control register		TBIC	FECH	47H
User flag register		FLAG	FEAH	00H
Internal data area base register		IDB	FFFH	FFH
Processor control register		PRC	FEBH	4EH
Wait control register		WTC	FE8H	FFFFH
Refresh mode register		RFM	FE1H	FCH
Standby control register		STBC	FE0H	Retention ^{Note 2}
External interrupt	External interrupt mode register	INTM	F40H	00H
	Interrupt request control registers	EXIC0 to EXIC2	F4CH to F4EH	47H
	Macro service control register	EMS0 to EMS2	F44H to F46H	Undefined
Interrupt priority register		ISPR	FFCH	00H

- Notes**
1. xx in the high-order eight bits of an address is a value specified in the IDB register.
 2. After power on reset: 00H

CHAPTER 14 ADDRESS GENERATION

14.1 Instruction Address

In addition to automatic address incrementation each time an instruction is executed, a number of instruction execution order control modes can be used, as described below.

14.1.1 Direct addressing

Two-byte or four-byte immediate data among the instruction's bytes is directly loaded into the PC or both the PS and PC to become a branch address.

This mode is used to execute the following instructions.

```
CALL far-proc
CALL memptr16
CALL memptr32
BR far-label
BR memptr16
BR memptr32
```

14.1.2 Relative addressing

One-byte or two-byte immediate data among the instruction's bytes is added to the PC as a signed displacement value to become a branch address.

When eight-bit displacement is given, a sign extension is made and the resulting 16-bit data is added to the PC. The PC contents when the addition is made indicate the starting address of the next instruction.

This mode is used to execute the following instructions.

```
CALL near-proc
BR near-label
BR short-label
Conditional branch instruction short-label
```

14.1.3 Register addressing

The contents of any desired 16-bit register specified in the 3-bit register specification field among the instruction's bytes are loaded into the PC as a branch address.

Unlike data, all eight of the 16-bit registers (AW, BW, CW, DW, IX, IY, SP, and BP) can be used.

This mode is used to execute the following instructions.

		Description	example
CALL	regptr16	CALL	AW
BR	regptr16	BR	BW

14.1.4 Register indirect addressing

The word or double word contents of the memory addressed by the 16-bit register (IX, IY, or BW) specified in the register specification field among the instruction's bytes are loaded into the PC or both the PC and PS as a branch address.

		Description	example
CALL	memptr16	CALL	WORD PTR [IX]
CALL	memptr32	CALL	DWORD PTR [IY]
BR	memptr16	BR	WORD PTR [BW]
BR	memptr32	BR	DWORD PTR [IX]

Remark When WORD PTR is described, memptr16 opcode is generated by the assembler. When DWORD PTR is described, memptr32 opcode is generated by the assembler.

14.1.5 Index addressing

One-byte or two-byte immediate data among the instruction's bytes is the signed displacement value, and is added to the 16-bit register (IX or IY) which serves as an index register. The word or double word contents of the memory addressed by the result are loaded into the PC as a branch address.

This mode is used to execute the following instructions.

		Description	example
CALL	memptr16	CALL	var [IX] [2]
CALL	memptr32	CALL	var [IY]
BR	memptr16	BR	var [IY]
BR	memptr32	BR	var [IX+4]

Remark If variable var has a word attribute, memptr16 opcode is generated by the assembler. If it has a double word attribute, memptr32 opcode is generated by the assembler.

14.1.6 Based addressing

One-byte or two-byte data among the instruction's bytes is the signed displacement value and is added to the 16-bit register (BP or BW) which serves as a base register. The word or double word contents of the memory addressed by the result are loaded into the PC as a branch address.

This mode is used to execute the following instructions.

		Description example
CALL	memptr16	CALL var [BP+2]
CALL	memptr32	CALL var [BP]
BR	memptr16	BR var [BW] [2]
BR	memptr32	BR var [BP]

Remark If variable *var* has a word attribute, `memptr16` opcode is generated by the assembler. If it has a double word attribute, `memptr32` opcode is generated by the assembler.

14.1.7 Based index addressing

One-byte or two-byte data among the instruction's bytes is used as a signed displacement value. This value, the contents of the 16-bit register (BP or BW) which serves as a base register, and the contents of the 16-bit register (IX or IY) which serves as an index register are added together. The word or double word contents of the memory addressed by the result are loaded into the PC as a branch address.

This mode is used to execute the following instructions.

		Description example
CALL	memptr16	CALL var [BP] [IX]
CALL	memptr32	CALL var [BW+2] [IY]
BR	memptr16	BR var [BW] [2] [IX]
BR	memptr32	BR var [BP+4] [IY]

Remark If variable *var* has a word attribute, `memptr16` opcode is generated by the assembler. If it has a double word attribute, `memptr32` opcode is generated by the assembler.

14.2 Memory Operand Address

A number of addressing modes for registers and memory to be handled when executing instructions can be used, as described below.

14.2.1 Register addressing

The register to be handled is addressed by the contents of the register specification field (reg = 3-bit field or sreg = 2-bit field) among the instruction's bytes.

One of eight word registers (AW, BW, CW, DW, BP, SP, IX, and IY) and eight byte registers (AL, AH, BL, BH, CL, CH, DL, and DH) is specified in a combination of reg and one bit (W) among the instruction's bytes that specifies a word or byte.

One of four segment registers (PS, SS, DS0, and DS1) is specified in sreg.

The opcode of an instruction may specify a particular register.

This mode is used to execute instructions having the following operand identifiers.

Identifier	Description
reg	AW, BW, CW, DW, SP, BP, IX, IY, AL, AH, BL, BH, CL, CH, DL, DH
reg16	AW, BW, CW, DW, SP, BP, IX, IY
reg8	AL, AH, BL, BH, CL, CH, DL, DH
sreg	PS, SS, DS0, DS1
acc	AW, AL

Description example

MOV	When reg, reg'
MOV	BP, SP
MOV	AL, CL

14.2.2 Immediate addressing

One-byte or two-byte immediate data among the instruction's bytes is to be handled as it is.

This mode is used to execute instructions having the following operand identifiers.

Identifier	Description
imm	8-bit or 16-bit immediate data
imm16	16-bit immediate data
imm8	8-bit immediate data
pop-value	16-bit immediate data

For imm, 8-bit or 16-bit data is determined by the assembler which decides the imm value described in the operand or another operand attribute described at the same time. Word/byte specification bit W is also determined.

Description example

MOV	When reg, imm
MOV	AL, 5; byte
MUL	When reg16, reg16', imm16
MUL	AW, BW, 1000H

14.2.3 Direct addressing

Immediate data among the instruction's bytes addresses the memory to be manipulated.

This mode is used to execute instructions having the following operand identifiers.

Identifier	Description
mem	16-bit variable specifying 8-bit or 16-bit memory data
dmem	16-bit variable specifying 8-bit or 16-bit memory data
imm4	4-bit variable indicating the bit field data's bit length

Description example

MOV	When mem, imm
MOV	WORD_VAR, 2000H
MOV	When acc, dmem
MOV	AL, BYTE_VAR

14.2.4 Register indirect addressing

The 16-bit register (IX, IY, or BW) specified in the memory specification field (mod or mem) among the instruction's bytes addresses the memory to be handled.

This mode is used to execute instructions having the following operand identifiers.

Identifier	Description
mem	[IX], [IY], [BW]

Description example

SUB	When mem, reg
SUB	[IX], AW

14.2.5 Automatic increment/decrement addressing

This mode belongs to register indirect addressing. After the default register contents address the memory to be manipulated, the register contents are automatically incremented or decremented (by one for byte processing or by two for word processing).

In other words, this addressing function enables automatic address updating for the next processing of a byte or word operand.

The direction flag (DIR) specifies increment or decrement addressing. If DIR = 0, increment addressing is specified; if DIR = 1, decrement addressing is specified.

This addressing mode is always applicable to default registers, and is used to execute instructions having the following operand identifiers.

Identifier	Default register
dst-block	IY
src-block	IX

The addressing mode and the counter which counts the number of byte/word operand processing repetitions (CW) are used in combination for block data processing control.

14.2.6 Index addressing

One-byte or two-byte immediate data among the instruction's bytes is the signed displacement value, which is added to the 16-bit register (IX or IY) that serves as an index register, and the result addresses the memory operand to be manipulated.

This addressing mode is useful for accessing array-type data. The displacement indicates the array start address and the index register contents determine the array element location.

This mode is used to execute instructions having the following operand identifiers.

Identifier	Description
mem	var [IX], var [IY]
mem16	var [IX], var [IY]
mem8	var [IX], var [IY]

Description example

```
TEST    When mem, imm
TEST    BYTE_VAR [IX], 7FH
TEST    BYTE_VAR [IX+8], 7FH
TEST    WORD_VAR [IX] [8], 7FFFH
```

Remark When variable var has a byte attribute, byte operand is specified and when it has a word attribute, word operand is specified. In either case, the corresponding opcode is generated by the assembler.

14.2.7 Based addressing

One-byte or two-byte immediate data among the instruction's bytes is the signed displacement value, which is added to the 16-bit register (BP or BW) that serves as a base register, and the result addresses the memory operand to be manipulated.

This addressing mode is useful for accessing structure-type data stored in a number of memory locations. The base register indicates the structure start address and the displacement selects one element in the structure.

This mode is used to execute instructions having the following operand identifiers.

Identifier	Description
mem	var [BP], var [BW]
mem16	var [BP], var [BW]
mem8	var [BP], var [BW]

Description example

```
SHL    When mem, 1
SHL    BYTE_VAR [BP], 1
SHL    WORD_VAR [BP+2], 1
SHL    BYTE_VAR [BP] [4], 1
```

Remark When variable var has a byte attribute, byte operand is specified and when it has a word attribute, word operand is specified. In either case, the corresponding opcode is generated by the assembler.

14.2.8 Based index addressing

One-byte or two-byte immediate data among the instruction's bytes is the signed displacement value. This value is added to the contents of the 16-bit register (BP or BW) which serves as a base register and the contents of the 16-bit register (IX or IY) which serves as an index register. The result addresses the memory operand to be manipulated.

This addressing can point to one data unit by changing both the base register contents and index register contents, and thus is very useful for accessing structure-type data that contains array-type data. In other words, the base register indicates the structure's starting address, the displacement value indicates the offset from the structure's starting address to the starting address of the array-type data, and the index register indicates the array data element location.

This mode is used to execute instructions having the following operand identifiers.

Identifier	Description
mem	var [base register] [index register]
mem16	var [base register] [index register]
mem8	var [base register] [index register]

Description example

PUSH	When mem16
PUSH	WORD_VAR [BP] [IX]
PUSH	WORD_VAR [BP+2] [IX+6]
PUSH	WORD_VAR [BP] [4] [IX] [8]

14.2.9 Bit addressing

Three-bit or four-bit immediate data among the instruction's bytes or the low-order three or four bits of the CL register specifies one bit of the 8-bit or 16-bit register or the memory to be manipulated.

If an instruction using this addressing mode is executed, only a particular bit of the specified register or memory can be tested (to determine 0 or 1), set, cleared, or inverted without considering other bit contents. This means that byte or word data need not be provided for one-bit manipulation such as set or reset when using the AND or OR instruction.

This mode is used to execute instructions having the following operand identifiers.

Identifier	Description
imm4	Word operand bit number
imm3	Byte operand bit number
CL	CL

Description example

TEST1	reg8, CL
TEST1	AL, CL
NOT1	reg8, imm3
NOT1	CL, 5
CLR1	mem16, CL
CLR1	WORD_VAR [IX], CL
SET1	mem16, imm4
SET1	WORD_VAR [BP], 9

14.2.10 Special function register addressing

One-byte immediate data among the instruction's bytes addresses the special function register to be manipulated as offset (unsigned) from the top of the special function register area.

This addressing mode is applicable only to the BTCLR instruction.

Identifier	Description
sfr	8-bit variable specifying 8-bit special function register

Description example

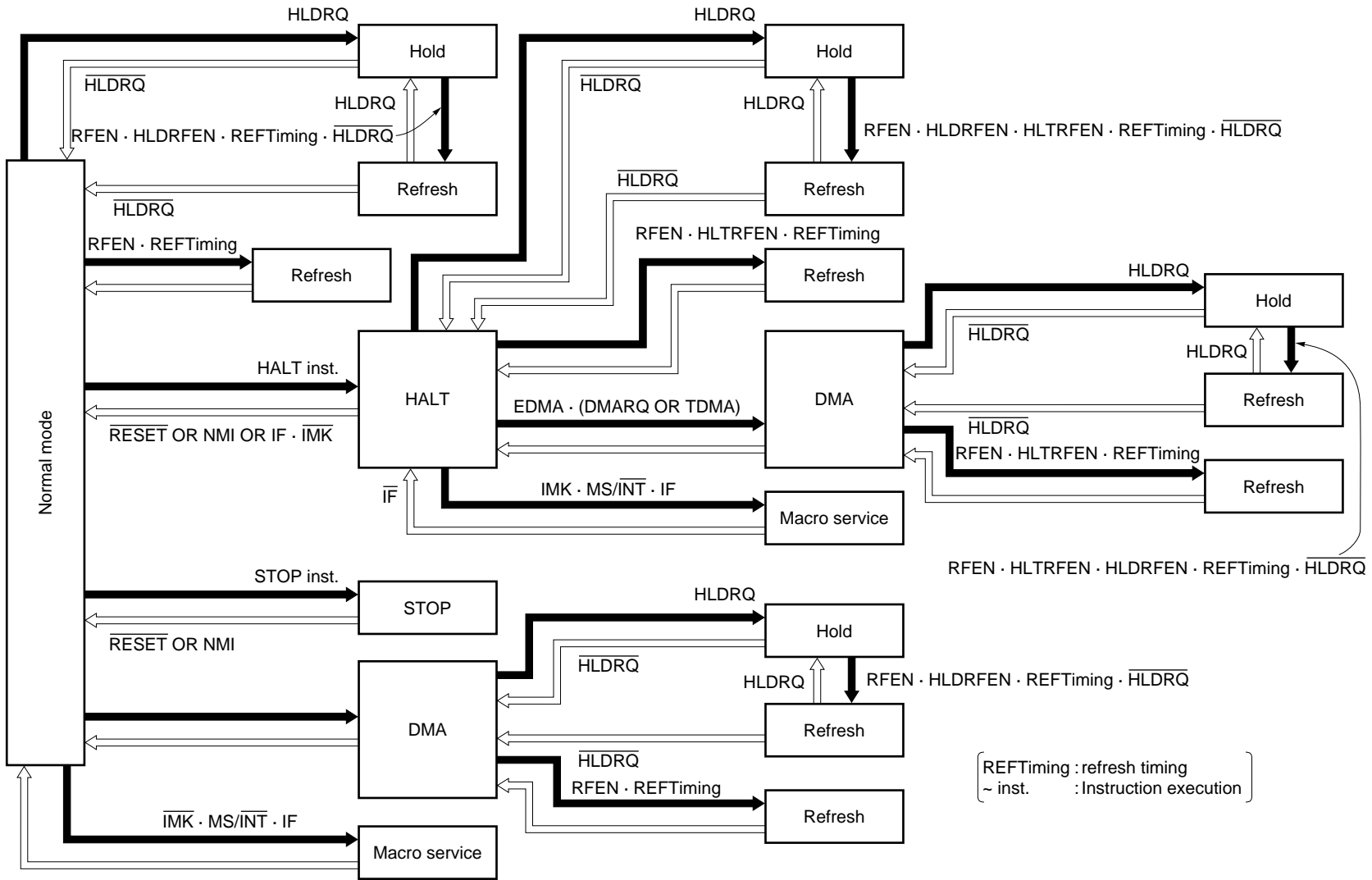
BTCLR	EXIC, 7, 45
-------	-------------

Remark The special function registers are mapped in memory space (between addresses $\times\times F00H$ and $\times\times FFFH$ where $\times\times$ is the IDB register value). For details, see section 3.5.3 **Special function register area**.

CHAPTER 15 OPERATION STATE TRANSITION

The V25 and V35 perform state transitions among various operation modes (macro service, DMA, refresh, hold, HALT, and STOP) under the conditions and via the flow shown in Figure 15-1.

Figure 15-1. Operation State Transition Diagram



APPENDIX A LIMITATIONS

A.1 Limitation on Conflict between Macro Service and INT

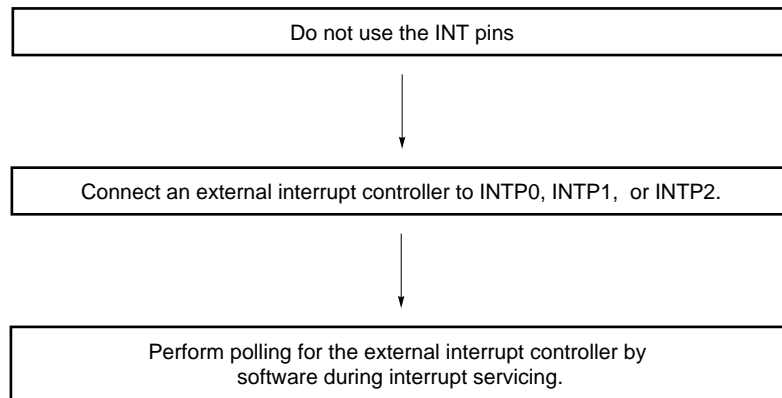
A.1.1 Description of limitation

If an INT request occurs while a macro service request is held pending under interrupt priority control, the macro service which should be pending may instead be executed first.

A.1.2 Avoidance methods

If this symptom adversely affects your system when using the macro service function, do not use the macro service function and INT input at the same time.

If this symptom adversely affects your system when an external interrupt controller (μ PD71059) is connected and the macro service function is used, perform interrupt servicing as shown below.

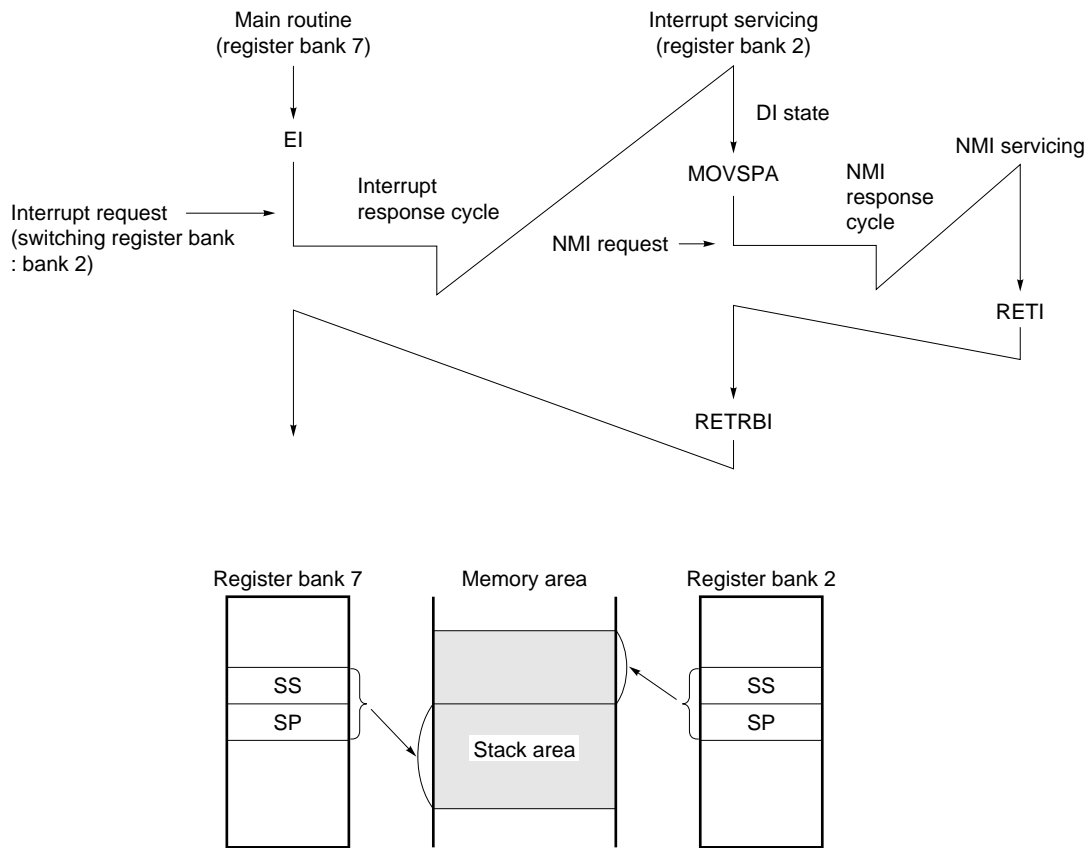


A.2 Cautions on Use of the MOVSPA Instruction

A.2.1 Device operation description (symptom)

In applications where register banks before and after switching use contiguous stack areas when register bank switching is performed in an asynchronously occurring interrupt, the MOVSPA instruction can be used to copy SS and SP between the register banks before and after switching. In this case, the stack areas used before and after register bank switching can be made contiguous by executing the MOVSPA instruction at the beginning of program processing of the register bank after switching. (See **Figure A-1.**)

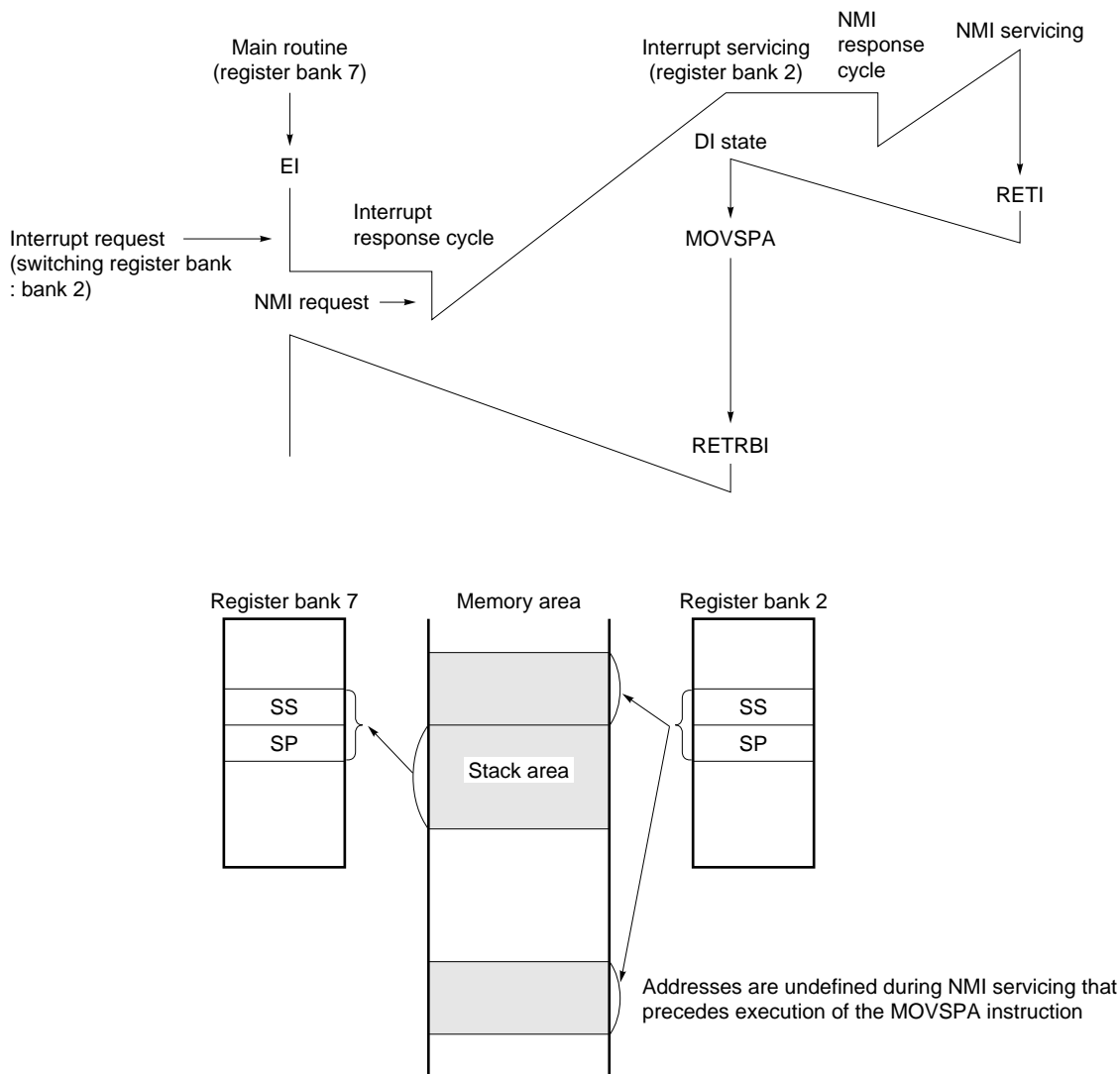
Figure A-1. Normal Operation



If an NMI request occurs just after response interrupt INT_{xx} is acknowledged by register bank switching, NMI is acknowledged before the INT_{xx} service routine is executed. Thus, when NMI is acknowledged, PS, PC, and PSW are saved in the stack area indicated by SS and SP in the new register bank.

Therefore, in applications where the stack areas of the register banks before and after switching are made contiguous by the MOVSPA instruction, if an NMI request occurs just after response interrupt by register bank switching is acknowledged, the NMI is acknowledged before SS and SP are copied and an undefined memory area is accessed (written) as the stack area. (See **Figure A-2.**)

Figure A-2. Abnormal Operation



A.2.2 Avoidance methods

When using response interrupt and NMI at the same time by register bank switching, avoid the symptom described above by either of the following methods.

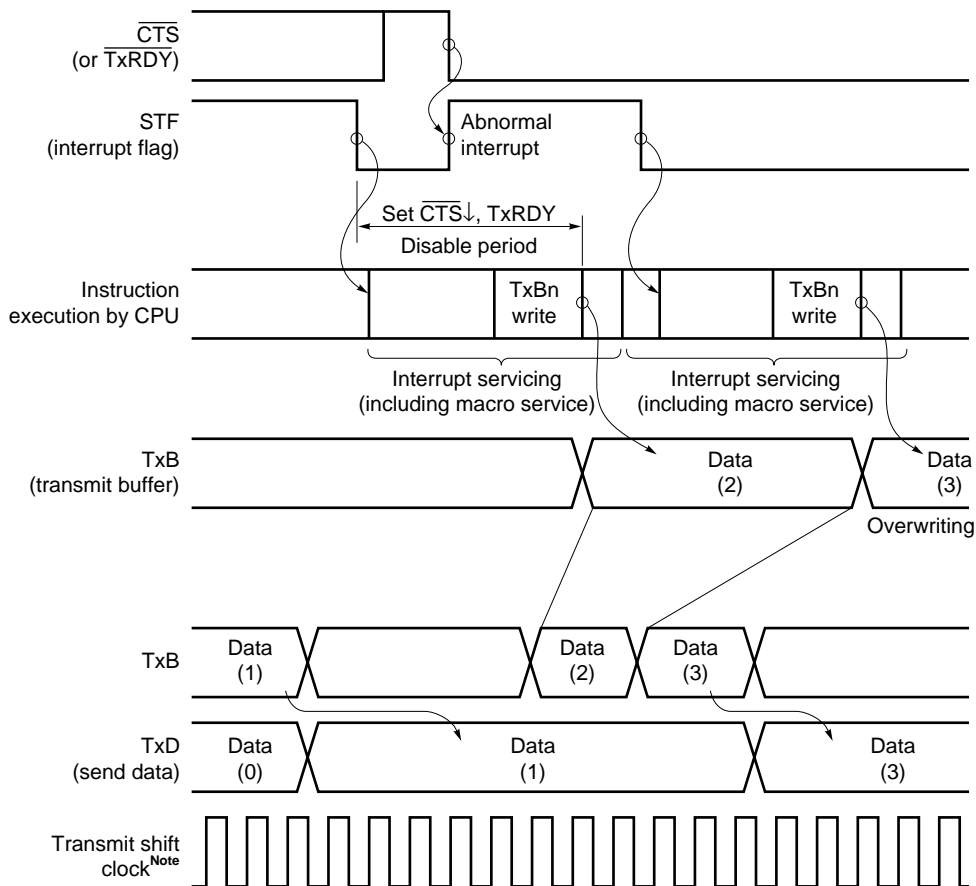
- (1) Do not use the MOVSPA instruction and set a separate stack area for each switched register bank when an interrupt occurs.
- (2) Prevent the stack area from being undefined by providing a spare stack to avoid the symptom described above.
 - (a) Reserve the area used as the spare stack and preset SS and SP of the switch register bank when an interrupt occurs in the area during the initialization routine.**Note**
 - (b) Set the values set in (a) just before the register bank is restored from the new register bank to the former register bank.

Note If (b) is not performed, the stack area becomes undefined again when the second or subsequent interrupt occurs.

A.3 Limitations on Send Data Missing by Transmission Disable Operation during Serial Transmission

A.3.1 Device operation description (symptom)

If the transmission enable state is changed by making the high-to-low transition of $\overline{\text{CTS}}$ input or by setting the TxRDY flag to 1 between the acknowledgment of a transmission completion interrupt request and the next write operation to the transmit buffer (TxB) on the on-chip serial interface, an extra transmission completion interrupt request occurs. (This also applies when macro service is used.) As a result, the send data (one character) written just before may be skipped by the interrupt service routine (or macro service) which overwrites into TxB.



Note When one character consists of 11 bits for transfer (for example, eight data bits, two stop bits, and no parity bit)

A.3.2 Avoidance methods

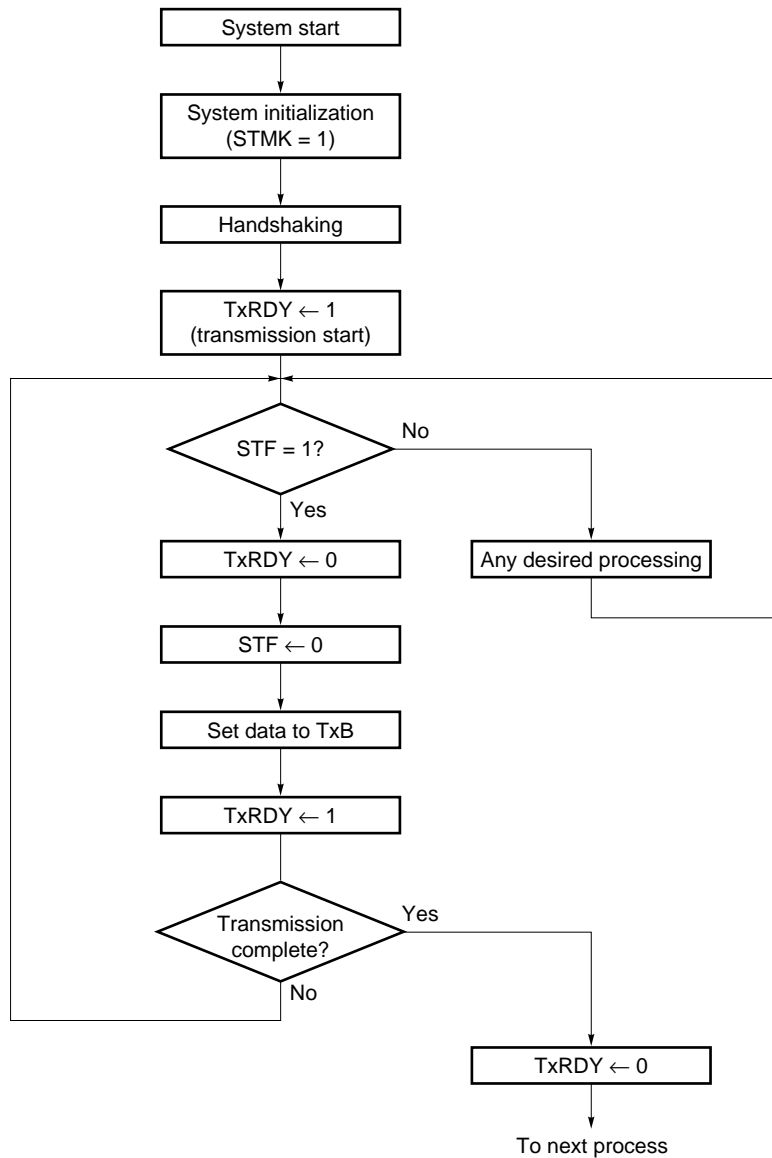
Perform the following two steps.

- (a) When using a transmission completion interrupt (including macro service), perform send/receive handshaking as described below, without using the $\overline{\text{CTS}}$ input pin.
 - Handshaking by polling general purpose ports
 - Handshaking by send character

To perform handshaking using $\overline{\text{CTS}}$ input, do not use transmission completion interrupts but instead process the transmission by polling via the steps described below. This process is shown in Figure A-3.

- 1) After system initialization, set the TxRDY flag (with the transmission completion interrupt masked).
- 2) Perform program polling of the STF flag. If the STF flag has been set, clear the TxRDY flag.
- 3) Write the send data into transmit buffer TxB and clear the STF flag.
- 4) Set the TxRDY flag.

Figure A-3. Example of Avoidance Process Flow



- (b) For transmission completion interrupt servicing (including interrupt servicing in which nesting is enabled for transmission completion interrupt servicing), do not execute an instruction to set the TxRDY flag before writing the send data to the transmit buffer TxB.

Setting the TxRDY flag is an operation that should be done only when starting transmission. There is usually no need to set or reset the TxRDY flag during the transmission operation.

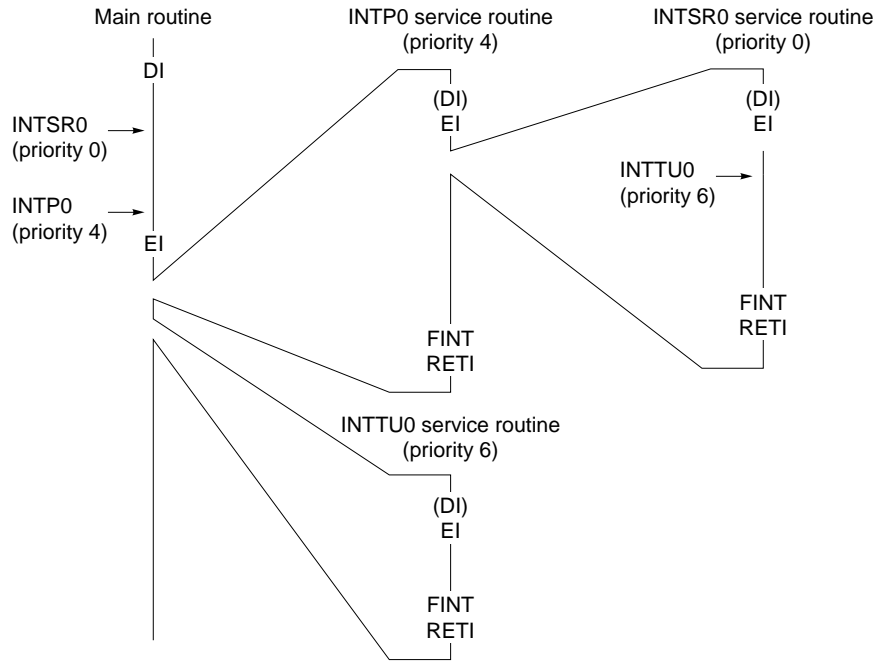
A.4 Cautions on Interrupt Priority Levels and Servicing Sequence

When the CPU state transits from DI to EI, an interrupt having a lower programmable priority among the pending interrupts in the DI state may be acknowledged before an interrupt having a higher programmable priority. (Because the generation timing depends only on the internal timing, the generation timing cannot be externally limited.)

However, if an EI instruction is executed during servicing of an interrupt that has a lower priority level, interrupts that have higher priority levels can be consecutively acknowledged under the multiple servicing control according to the priority levels. For the interrupt that has a lower priority level, execute an EI instruction at the beginning of interrupt servicing to enter the EI state during servicing.

The state should be EI to service the interrupt having the lower priority level. If the DI state is prolonged while servicing the interrupt having the lower priority level, the interrupt requests having higher priority levels that occur in the meantime are held pending during the period, and therefore the priority setting is insignificant. Execute an EI instruction as soon as possible while servicing the interrupt having the lower priority level.

Example of normal interrupt servicing under multiple servicing control



A.5 Limitations on Instruction Slips during HALT Mode

A.5.1 Description of limitation

Immediately after execution of the HALT instruction, if there is a request for an NMI, vectored interrupt, register bank switching interrupt, macro service, or DMA, the instruction immediately after the HALT instruction is executed after completion of the HALT instruction. After that, the requested service is executed.

A.5.2 Avoidance method

During HALT mode, when a request occurs for any of the above interrupts, macro service, or DMA, immediately after the HALT instruction add an instruction (such as an NOP instruction) that does not affect program operation.

A.6 Limitation on CPU Deadlock Due to CLKOUT Output

A.6.1 Description of limitation

When the P07/CLKOUT pin is at low-level output (when bit 7 = 0 in the P0 register and PM0 register) or when it is in the input state (high impedance state: bit 7 = 1 in the PM0 register), changing the PMC0 register's bit 7 from 0 to 1 will cause a system clock to be output from the P07/CLKOUT pin, which may cause a CPU deadlock.

A.6.2 Avoidance method

This symptom can be avoided via the following two methods.

- Avoidance using hardware
Externally connect a pull-up resistor to the P07/CLKOUT pin.
- Avoidance using software
Before switching the P07/CLKOUT pin to control mode, set it for high level output as an output port.

A.7 Caution on Starting Timer Countdown

A.7.1 Device operation description (symptom)

If a countdown is started when the timer register TM value is 0, the following symptoms may occur just after the countdown is started.

- Occurrence of timer interrupt request
- Inversion of TOUT pin level (when ENTO bit = 1)

If such symptoms occur when the interval timer is used, the timer register TM which is counting down can be stopped by clearing (to 0) the TS bit of the TMC register. At that time, if the timer stop timing overlaps with the timing at which TM is set to 0, similar symptoms will occur just after the next countdown is started (the TMC register's TS bit is set to 1).

A.7.2 Avoidance method

After stopping the countdown of the TM timer register (by clearing the TS bit), write word data other than 0000H into the TM register. This will suppress any unnecessary occurrence of TOUT pin inversion or a timer interrupt request just after the countdown start and the operation will proceed normally.

Example

```

MOV  TMC0, 00H
MOV  TMIC0, 00H
MOV  MD0, 0FE0H

MOV  TMC0, 88H   Start countdown

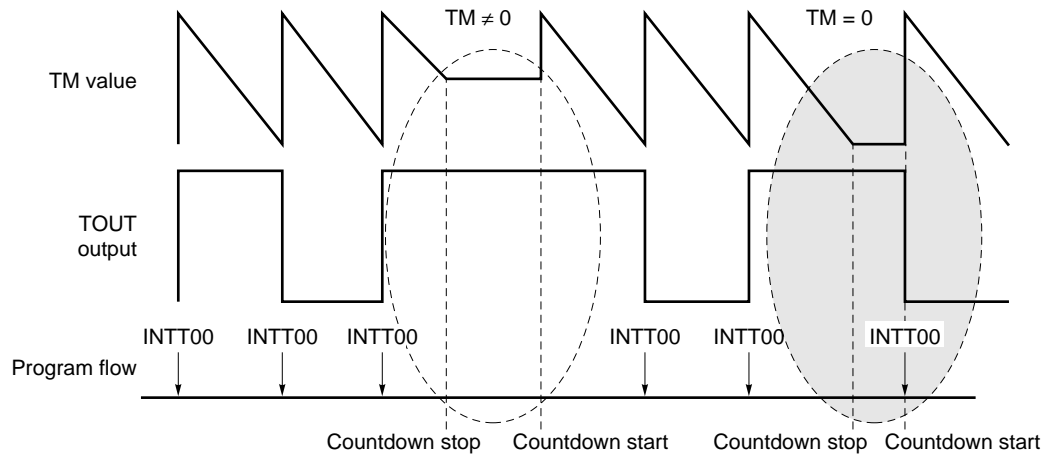
MOV  TMC0, 00H   Stop countdown

MOV  TM0, 0001H  Write a value other than 0000H into TM register

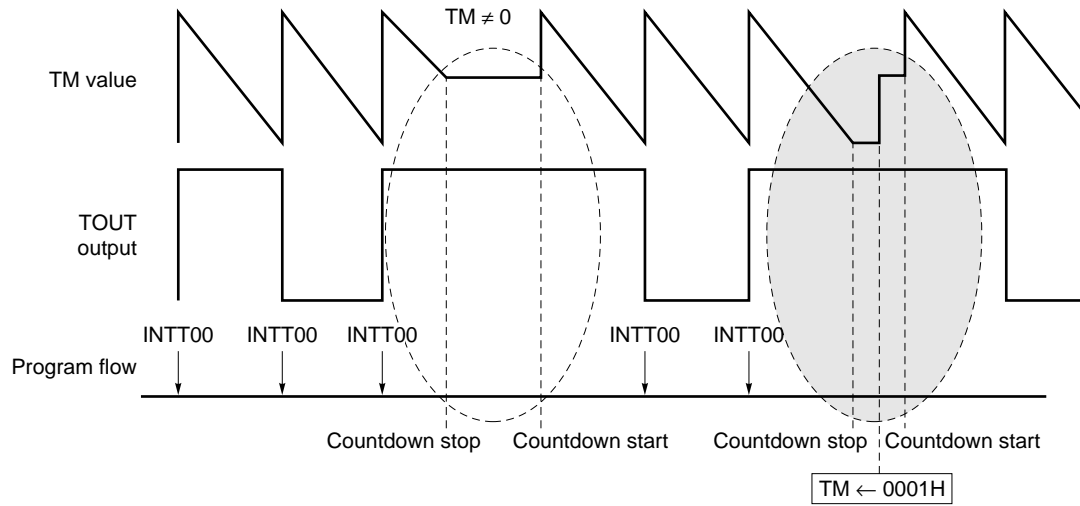
MOV  TMC0, 88H   Start countdown

```

When avoidance method is not performed



When avoidance method is performed



A.8 Limitations on Macro Service Masks

A.8.1 Device operation description (symptom)

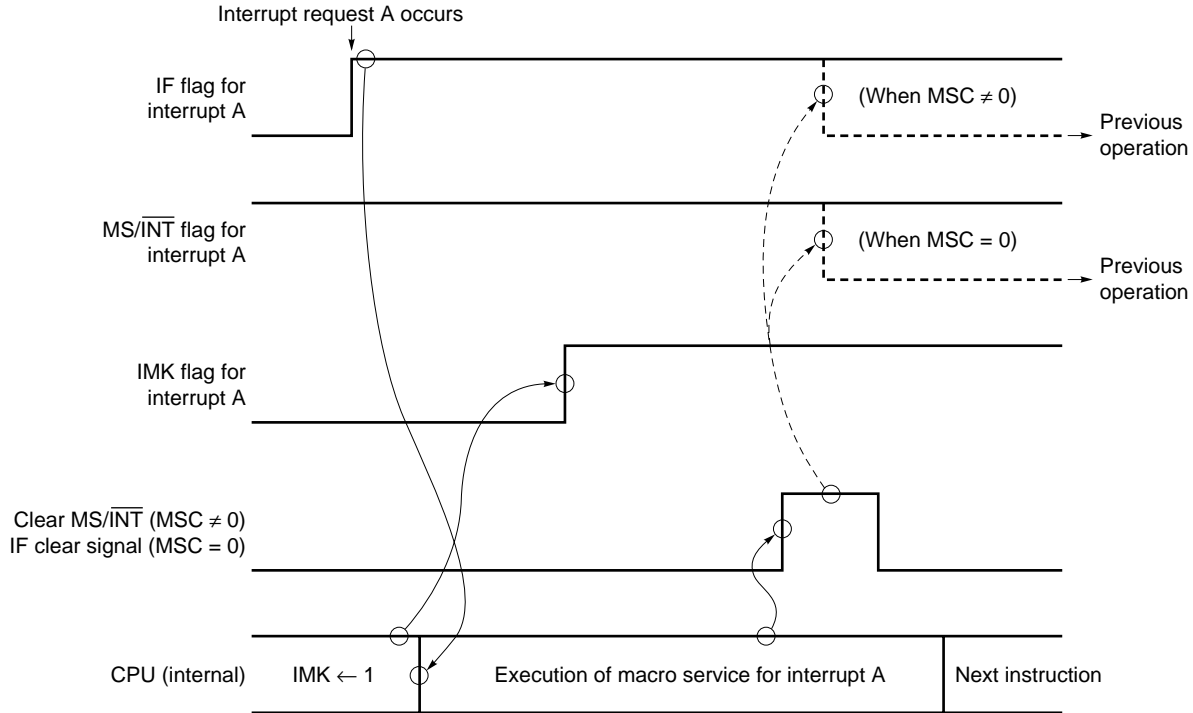
Immediately before execution of interrupt A (macro service A) that was set as a macro service response, if an instruction is executed to set macro service A's IMK flag, the IMK flag will be set during execution of the subsequent macro service A. At that time, abnormal execution of macro service A may occur, such as the symptoms described below.

- **When macro service counter (MSC) = 0**

Because the IF flag is not cleared after execution of macro service A, macro service A is executed again after the mask is released, even if no interrupt request has occurred.

- **When macro service counter (MSC) = 0**

Because the MS/INT flag is not cleared after execution of macro service A, a completion interrupt does not occur for macro service A and macro service A is executed again after the mask is released. As a result, MSC is decremented from 0 to 0FFH and macro service is executed each time an interrupt occurs until MSC = 00H.



A.8.2 Avoidance method

Follow the steps shown below when setting or resetting the mask flag set for macro service.

- **Set**

```
DI
CLR1    xxIC, 05H                ; Reset MS/ $\overline{\text{INT}}$  flag
NOP
NOP     }
NOP     } Be sure to insert at least four NOP instructions.
NOP     }
SET1    xxIC, 06H                ; Set IMK flag
EI
```

- **Reset**

```
DI
SET1    xxIC, 05H                ; Set MS/ $\overline{\text{INT}}$  flag
CLR1    xxIC, 06H                ; Reset IMK flag
EI
```

A.9 Limitations on CVTBD/CVTDB Instructions

A.9.1 Device operation description (symptom)

When the CVTBD or CVTDB instruction is executed immediately after executing one of the instructions listed in Table A-1, data will not be written correctly during the last memory write operation of the instruction (from Table A-1). Instead, undefined data will be written. At that time, only the write data will change and flags will operate normally.

However, this symptom will not occur if internal RAM access is enabled (RAMEN = 1) or if the number of waits for the memory being accessed by the instruction (from Table A-1) is less than two clocks (for the V25) or less than one clock (for the V35).

A.9.2 Avoidance method

Do not execute the CVTBD or CVTDB instruction immediately after executing one of the instructions listed in Table A-1. Instead, execute one or more NOP instructions immediately before the CVTBD or CVTDB instruction.

Example OR mem, reg
 NOP **Be sure to insert at least one NOP instruction.**
 CVTBD

Table A-1. Instructions Subject to Limitations Concerning CVTBD and CVTDB Instructions

Instruction group	Mnemonic	Operand	Instruction group	Mnemonic	Operand
Data transfer instructions	MOV	mem, reg	Bit manipulate instructions	CLR1	mem16, CL
		mem, imm			mem8, imm3
		mem16, sreg			mem16, imm4
	XCH	mem, reg		SET1	mem8, CL
		reg, mem			mem16, CL
Primitive block transfer instructions	MOVBK	dst_block, src_block			mem8, imm3
	STM	dst_block			mem16, imm4
Input/output instructions	OUT	imm8, acc	Shift instructions	SHL	mem, 1
		DW, acc			mem, CL
Primitive input/output instructions	OUTM	DW, src_block			
Add and subtract instructions	ADD	mem, reg		SHR	mem, 1
		mem, imm			mem, CL
	ADDC	mem, reg	mem, imm8		
		mem, imm	SHRA		mem, 1
	SUB	mem, reg			mem, CL
		mem, imm			mem, imm8
	SUBC	mem, reg	Rotate instructions	ROL	mem, 1
mem, imm		mem, CL			
BCD operation instructions	ROL4	mem8			
	ROR4	mem8		ROR	mem, 1
Increment and decrement instructions	INC	mem			mem, CL
	DEC	mem	mem, imm8		
Complement operation instructions	NOT	mem	ROLC	mem, 1	
	NEG	mem		mem, CL	
Logical operation instructions	AND	mem, reg			mem, imm8
		mem, imm	RORC	mem, 1	
	OR	mem, reg		mem, CL	
		mem, imm		mem, imm8	
	XOR	mem, reg	Stack manipulate instructions	PUSH	mem16
mem, imm		reg16			
Bit manipulate instructions	NOT1	mem8, CL			sreg
		mem16, CL			PSW
		mem8, imm3			R
		mem16, imm4	imm		
	CLR1	mem8, CL	POP	mem16	

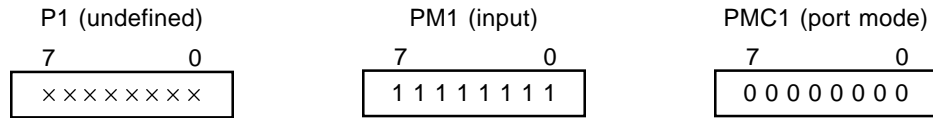
[MEMO]

APPENDIX B SETTING EXAMPLES

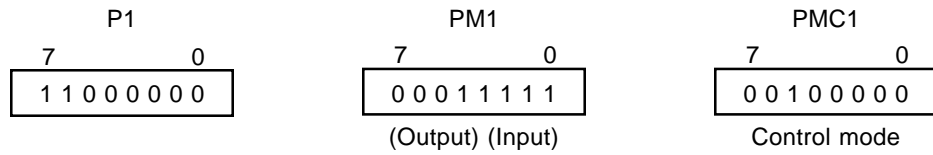
B.1 Ports

Shown below is an example of port 1 initialization executed after a power-on reset.

When reset



After setup



```

.....
;
;***                                     ***
;                                     Special function register setting
;
;*****
;
;

```

```

MOV     P1,      11000000B
MOV     PM1,     00011111B
MOV     PMC1,    00100000B ; TOUT output
;
;

```

P10 to P14 : Changes when data is input via input port
 P15 : TOUT output occurs when in control mode
 P16 and P17 : "1" is output from the output port

B.2 Programmable Wait, Processor Control, and Refresh Function

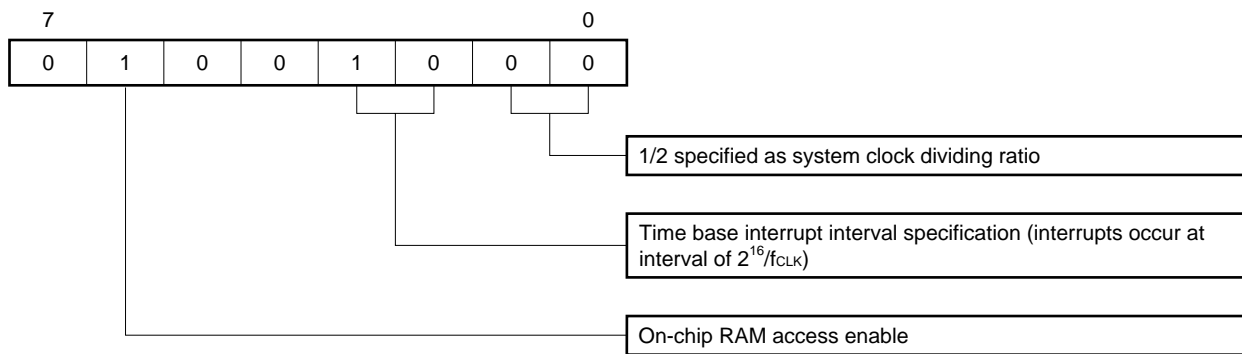
Mode setting method for wait insertion, refresh, etc.

```

*****
;
;***               Initialization of special function register               ***
;
;*****
;
;
;
;
MOV     PRC,  01001000B       ; <1>
MOV     WTCH, 00011010B       ; <2>
MOV     WTCL, 01010101B       ; <2>
MOV     RFM,  11110101B       ; <3>
;
;
MOV     STBC, 00000001B       ; <4>
;
;
HALT
;
;
;
;
MOV     RFM,  00000000B       ; <5>
STOP
;
;

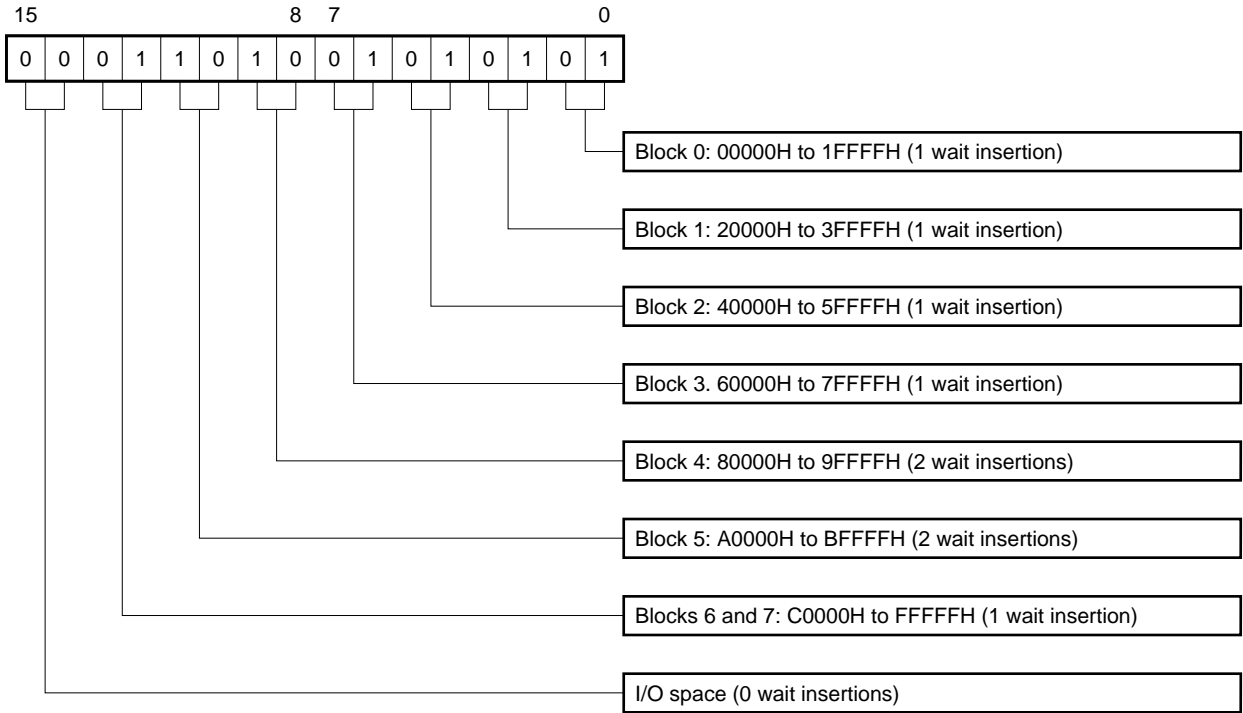
```

<1> PRC (Processor control register)



<2> WTC (Wait control register)

Separate wait insertions for 8-block memory space and I/O space



These can be organized as shown below.

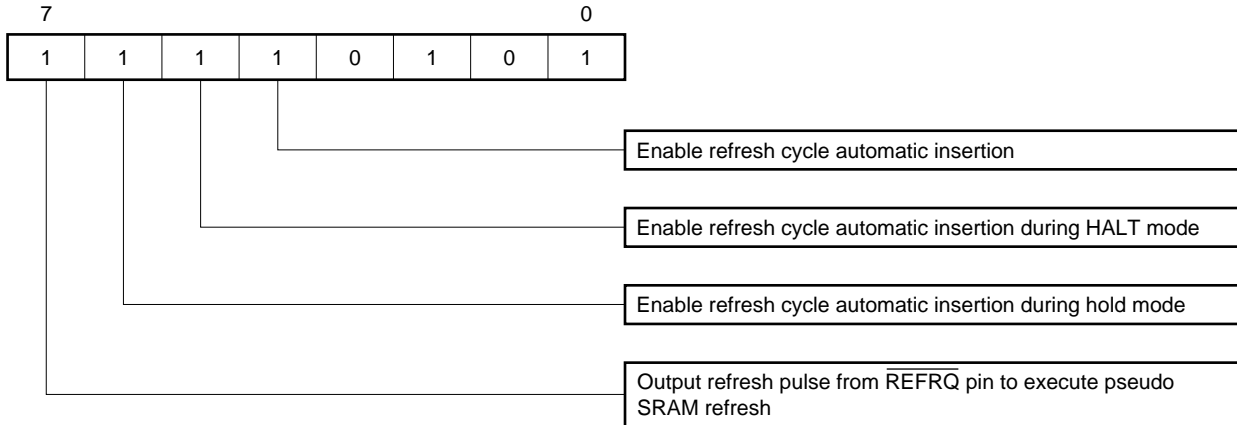
```
MOV WTC, 1A55H
```

Relation between memory and wait states

Memory location	Capacity	Block(s)	Wait state(s)
00000H to 07FFFH	32 Kbytes	Block 0	1 state
40000H to 47FFFH	32 Kbytes	Block 2	1 state
80000H to BFFFFH	256 Kbytes	Blocks 4 and 5	2 states
F8000H to FFFFFH	32 Kbytes	Blocks 6 and 7	1 state

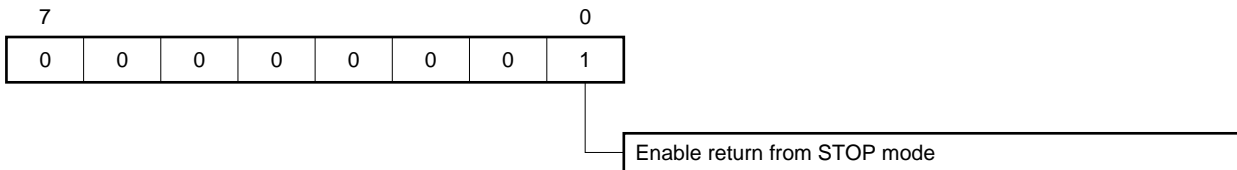
<3> RFM (Refresh mode register)

Inserts a refresh cycle into a series of bus cycles.



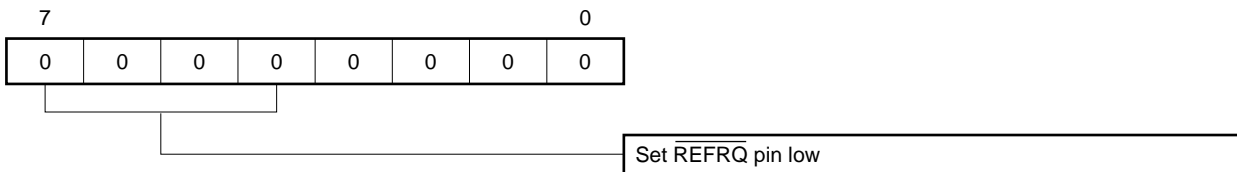
<4> STBC (Standby control register)

Controls return from STOP mode.

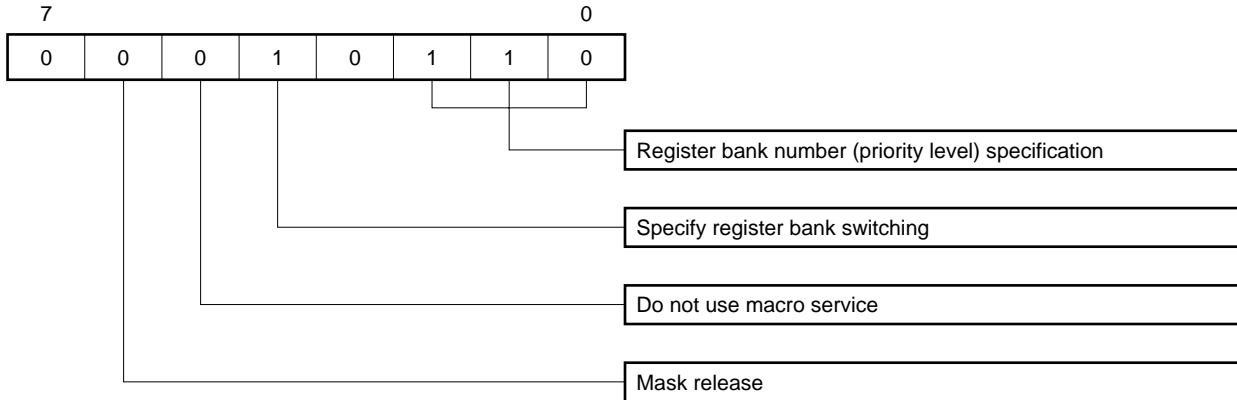


<5> RFM (Refresh mode register)

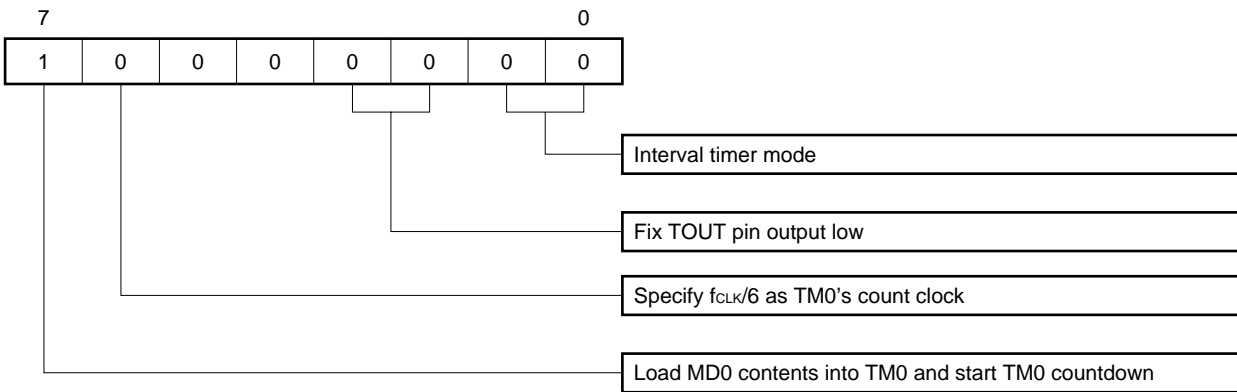
Sets $\overline{\text{REFRQ}}$ pin low to use power-down self-refresh function for pseudo SRAM.



- <1> The PS and vector PC for register bank 6 to be switched to must be already initialized.
When necessary, initialize other registers such as DS0 and DS1.
- <2> TMIC0 (Timer unit 0's interrupt request control register)



- <3> TMC0 (Timer control register 0)



- <4> Transfers contents of SS and SP in register bank 7 prior to switching to SS and SP in register bank 6 to be switched to.
- <5> When interrupt servicing from the peripheral hardware terminates, the FINT instruction must be executed before the RETI instruction or RETRBI instruction. Use the RETRBI instruction to return from register bank switching.
When the countdown results in a count value of 0, an interrupt request (INTTU0) occurs.

B.3.3 Register bank switching by instruction (MOVSPB or TSKSW instruction)

Switch from register bank 7 to register bank 6.

```

*****
;
;***                               Switch using MOVSPB reg16 and TSKSW reg16                               ***
;*****
;
;
;
;
;
;
;
MOV      REGBANK. BK6. BPC,  OFFSET BANK6   ; <1> Initialization of PC save
MOV      REGBANK. BK6. BPSW, 0E002H        ; <1> PSW save
MOV      REGBANK. BK6. BPS,   SEG BANK6    ; <1> PS
MOV      REGBANK. BK6. BDS0,  0             ; <1> DS0
MOV      REGBANK. BK6. BDS1,  0             ; <1> DS1
;
;
MOV      AW, 6                              ;
MOVSPB   AW                                ; <2>
TSKSW    AW                                ; <3>
;
;
;
;=====
;====                                Processing of register bank 6                                ====
;=====
;
;
BANK6:
;
;
MOV      AW, 7                              ; <4>
TSKSW    AW
;
;
;

```

- <1> To execute the TSKSW instruction, the PS, PC save area, SS, SP, and PSW save area in the register bank to be selected must be previously initialized. (In section B.3.3, this is done for the SS and SP by using the MOVSPB instruction.) When using the TSKSW instruction, the selected register bank’s PSW save area and PC save area values are loaded.
- <2> Use the MOVSPB instruction to set the SS and SP values before switching to the SS and SP switching destinations. In this case, you are switching to register bank 6, so set these values to register bank 6’s AW register.
- <3> Use the TSKSW instruction to select register bank 6 as the register bank to be switched to and load the previously stored PC save area contents into the PC for a branch. Execute the MOVSPB instruction and TSKSW instruction to switch to register bank 6.
- <4> If, as in <1>, initialization was required before register bank switching, the TSKSW instruction would be executed to select register bank 7 for a branch. However, in this case, you are not using the MOVSPB instruction and so the stack cannot be used continuously.

- Cautions**
- 1. If the TSKSW instruction is used during register bank switching caused by a BRKCS instruction or an interrupt request occurrence, the contents of the PSW save area are destroyed and a return to the previous bank cannot be made. However, there is no problem if it is used during register bank switching caused by the TSKSW instruction.**
 - 2. The values for RB0, RB1, and RB2 in the PSW save area to be switched to must match the number of the register bank to be switched to.**

B.4 Access to Internal Data Area

Indicates access when the internal data area (consisting of on-chip RAM and special function registers) is from 0FE00H to 0FFFFH. However, this is when setup of assembler (RA70116-I) pseudo instructions such as ASSUME and ASGNSFR has been completed.

```

;*****
;***
;*****
;
;
START:
    SETIDB    0FHNote          ;Set physical address base address (0F00H) in IDB register

    MOV      AW, DATA        ;Set DATA (0000H) in segment register
    MOV      DS0, AW          ;
;
;=====
;===                          Initialization of special function register                          ===
;=====
;
    MOV      P0,    00001111B ;Set to output 1 from port 0 (P00 to P03) and
    MOV      PM0,  00000000B ;to output 0 from P04 to P07
    MOV      PMC0, 00000000B ;
;
    MOV      PRC,  01000100B ;On-chip RAM access enable (set bit 6 to 1)
    ;
    MOV      AL, [FE00]      ;Fetch one byte of on-chip RAM contents
    ;                          ;AL←[0FE00]
    MOV      AL, P0          ;Read P0 contents to AL
    ;

```

Note This pseudo instruction indicates the internal data area address in the assembler (RA70116-I). The assembler generates the following instructions corresponding to this pseudo instruction description.

```

    PUSH    DS0
    PUSH    0FFFFH
    POP     DS0
    MOV     DS0 : BYTE PTR [0FH], xx
    POP     DS0

```

“xx” is set as the expression value (0FH) description for the operand.

B.5 Timer Unit

The interval timer (timer 1) and one-shot timer (timer 0) are used when timer unit interrupts occur.

(1) Interval timer mode

```

:
:
:
*****
;
;
;
Interval timer mode setting (about 4.9 ms)
;
;
;
;
:
:
MOV      TMIC2, 0000111B      ; <1>
MOV      TMC1,  00000000B     ; <2>
MOV      MD1,   0FF0H         ; <3>
:
:

```

(2) One-shot timer mode

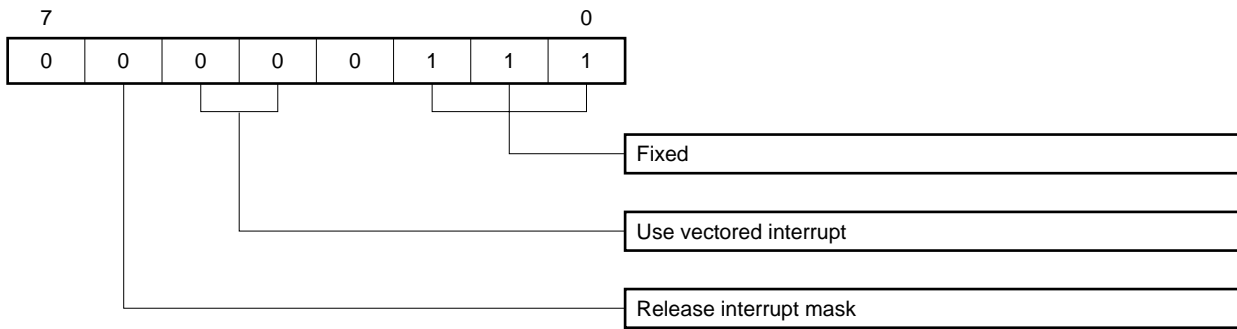
```

:
:
*****
;***                               One-shot timer mode setting (about 9.8 ms)                               ***
;*****
;
:
:
MOV      TMIC0, 00000001B          ; <4>
MOV      TMC0, 00001001B          ; <5>
MOV      MD0, 0FF0H                ; <6>
:
:
:
=====
;===                               Vector address No. 28 (INTTU0) setting                               ===
=====
;
;
MOV      IY, 28*4
MOV      WORD PTR[IY],            OFFSET INTTU0
MOV      WORD PTR[IY+2],          SEG INTTU0
:
:
:
=====
;===                               Vector address No. 30 (INTTU1) setting                               ===
=====
;
;
MOV      IY, 30*4
MOV      WORD PTR[IY],            OFFSET INTTU1
MOV      WORD PTR[IY+2],          SEG INTTU1
;
EI
SET1     TMC1, 7                    ; <7>
:
:
:
=====
;===                               Timer start                                                       ===
=====
;
;
SET1     TMC0, 5                    ; <8>
:
:
:
=====
;===                               Timer unit interrupt servicing                                       ===
=====
;
;
INTTU0:
:
FINT
RETI                                         ;Return from interrupt
;
INTTU1:
:
FINT
RETI                                         ;Return from interrupt

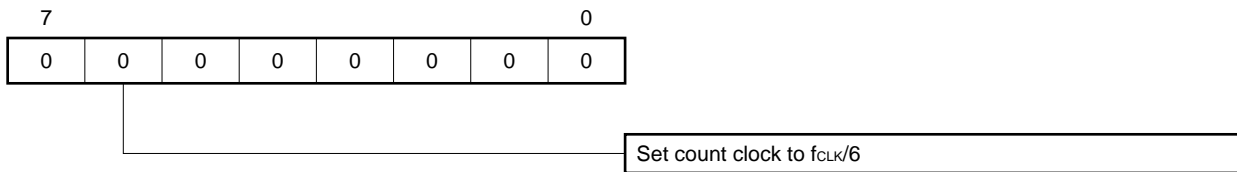
```

(1) Interval timer mode

<1> TMIC2 (timer unit interrupt request control register 2)



<2> TMC1 (timer control register 1)

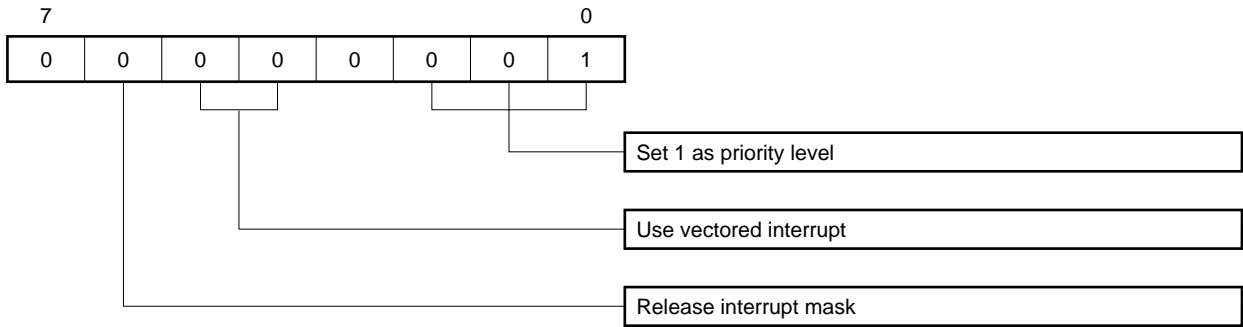


<3> MD1

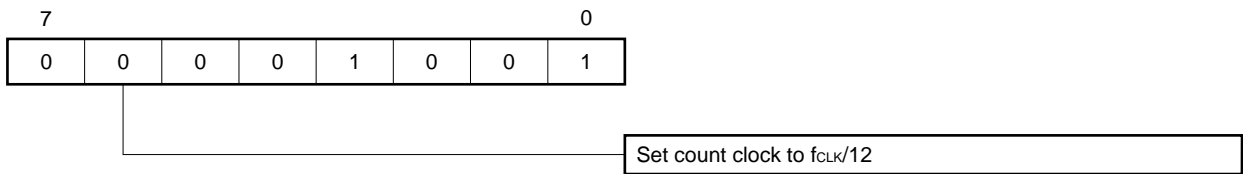
Sets the count value (FF0H). Interrupts occur at an interval of about 4.9 ms.

(2) One-shot timer mode

<4> TMIC0 (timer unit interrupt request control register 0)



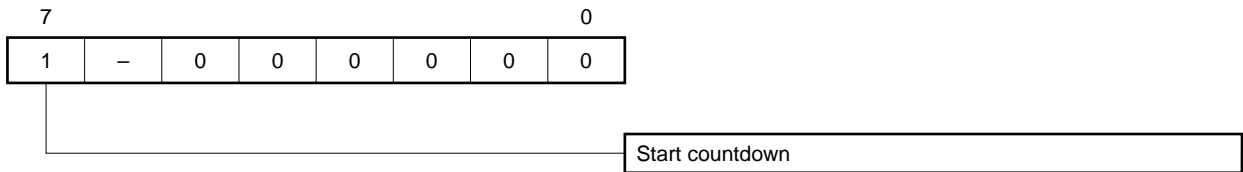
<5> TMC0 (timer control register 0)



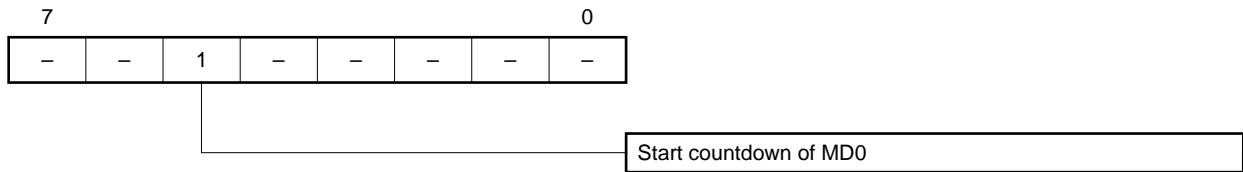
<6> MD0

Sets the count value (FF0H). One interrupt occurs after an interval of about 9.8 ms.

<7> TMC1 (timer control register 1)



<8> TMC0 (timer control register 0)



B.6 I/O Interface Mode

The I/O interface mode (reception) is used with vectored interrupts.

```

;+++++
;+++                               Special function register setting          +++
;+++++
;
;      MOV      SCM0, 00000000B      ; <1>
;      MOV      SCC0, 00000011B      ; <2>
;      MOV      BRG0, 130             ; <2>
;
;
;      MOV      SEIC0, 01000011B     ; <3>
;      MOV      SRIC0, 10000111B     ; <3>
;
;
;      MOV      P1, 11111111B        ;Set port 1 (P16) to control mode and execute  $\overline{SCK0}$  output
;      MOV      PM1, 01000000B        ;
;      MOV      PMC1, 00000000B        ;
;
;
;      SET1     SCM0, 6                ; <4>
;
;      EI                                ;SRIC0 (setting bit 7 to 1) enables interrupts and
;                                       ;reception completion interrupt occurs.

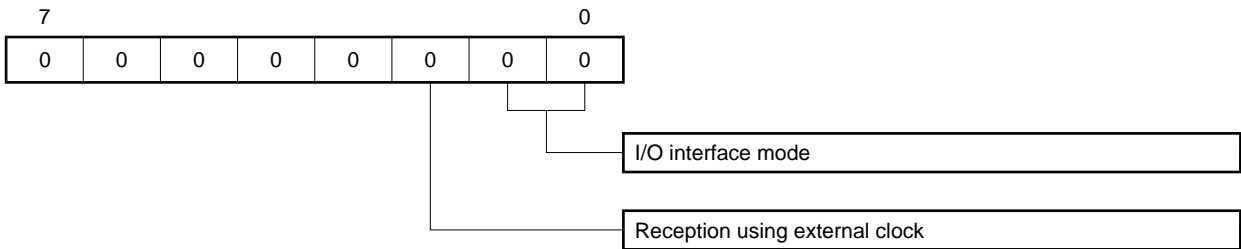
```

```

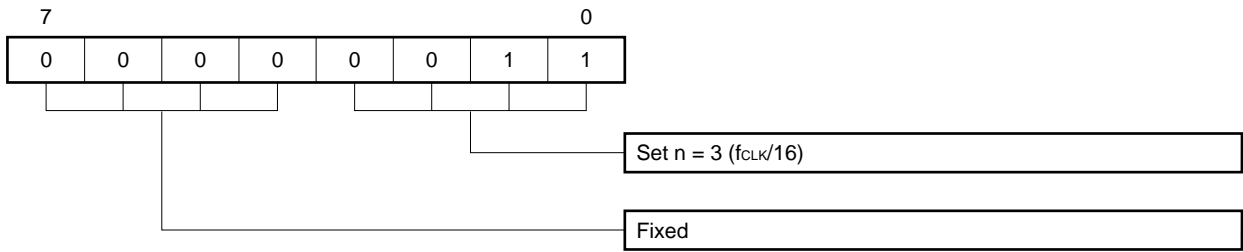
;*****
;***      Serial data reception processing (reception completion interrupt being serviced)      ***
;*****
;
;
INTSR0:
;
;      MOV      AL, RXB0                ;Fetch receive data from receive buffer (RxB0) and load to AL.
;
;
;      FINT
;      RETI

```

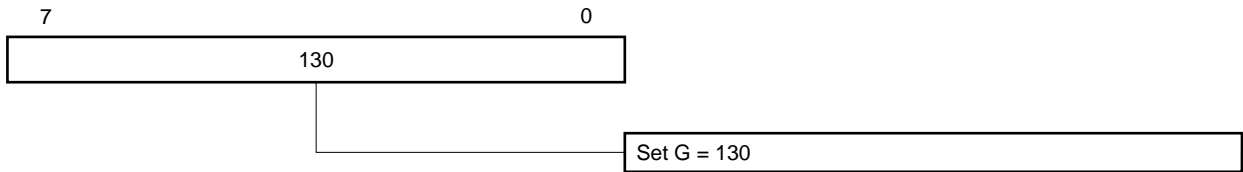
<1> SCM0 (serial mode register channel 0)



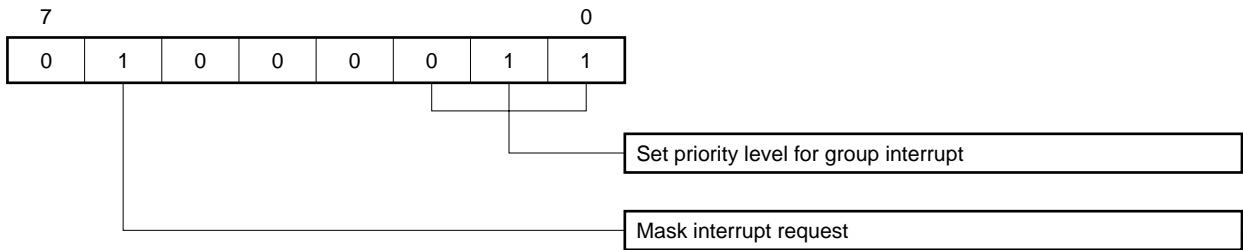
<2> SCC0 (serial control register 0)



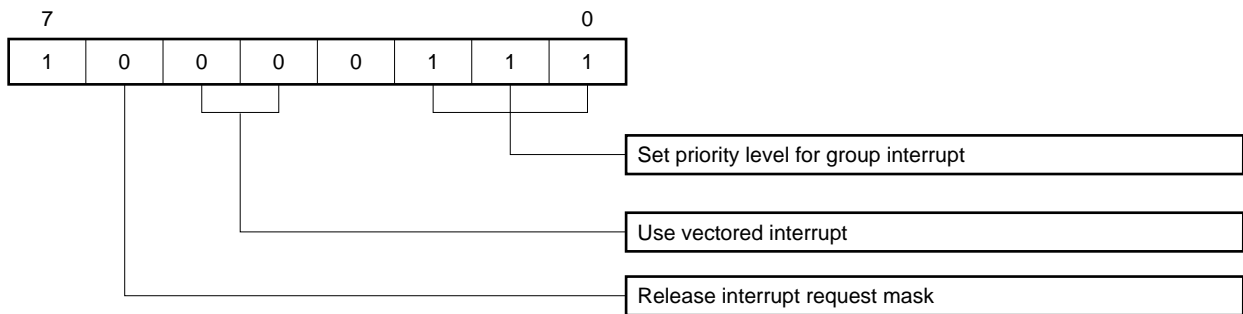
BRG0 (baud rate generator register 0)



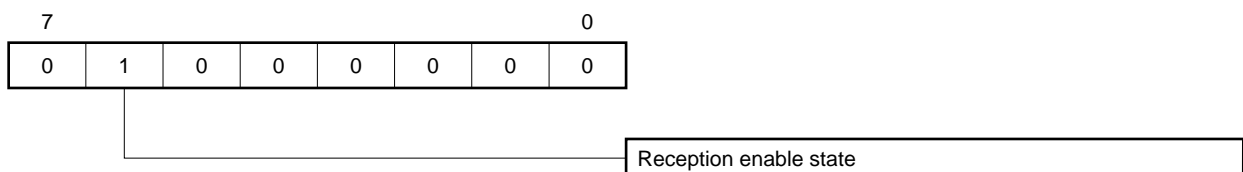
<3> SEIC0 (serial error interrupt request control register 0)



SRIC0 (serial reception interrupt request control register 0)



<4> SCM0 (serial mode register channel 0)



B.7 Macro Service

Shown below is an example in which (BAR_CODE) is the starting address of the memory space where the send data are stored and (REGBANK) is the starting address of macro service channel 0.

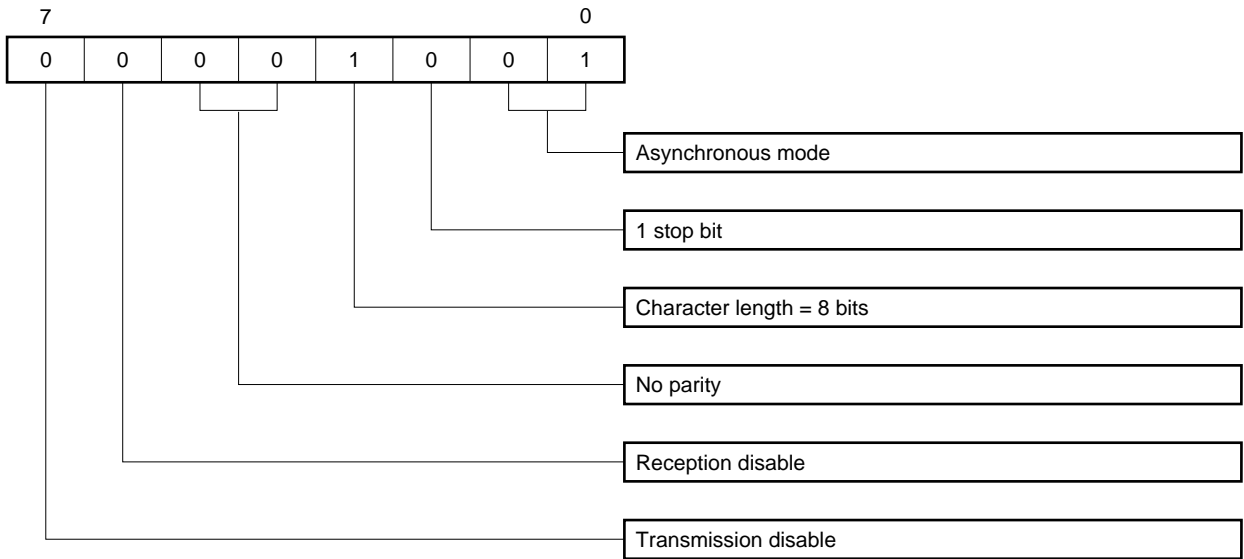
B.7.1 Normal mode (serial interface UART transmission)

```

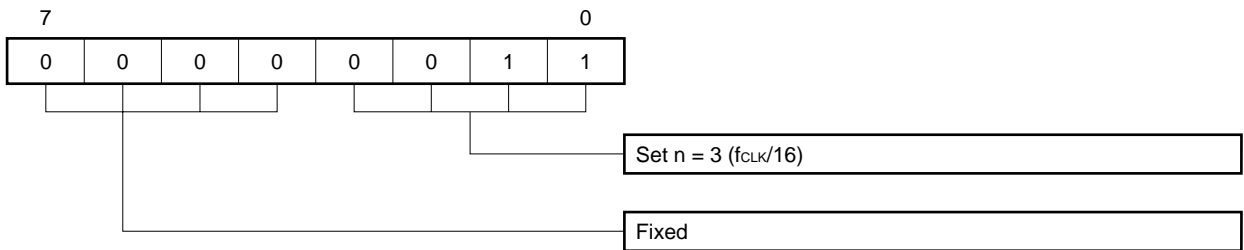
;
;
;+++++
;+++                               Special function register setting                               +++
;+++++
;
;
;      MOV      SCM0, 00001001B      ; <1>
;      MOV      SCC0, 00000011B      ; <2>
;      MOV      BRG0, 130             ; <2>
;
;
;      MOV      SEIC0, 01000011B     ; <3>
;      MOV      STIC0, 00110111B     ; <4>
;
;
;*****
;***                               Initialization of macro service (channel 2) normal mode                               ***
;*****
;
;      MOV      STMS0, 00000010B      ; <5>
;
;
;      MOV      IY, OFFSET REGBANK+8*2 ; <6>
;      MOV      BYTE PTR [IY], 15     ; <7>
;      MOV      BYTE PTR [IY+1], LOW TXB0 ; <8>
;      MOV      WORD PTR[IY+4], OFFSET BAR_CODE ; <9>
;      MOV      WORD PTR[IY+6], SEG BAR_CODE ; <10>
;
;
;*****
;***                               Register bank 3 setting                               ***
;*****
;
;                                           ;See section B.3
;
;*****
;
;      SET1     SCM0, 7                ; <11>
;
;
;

```

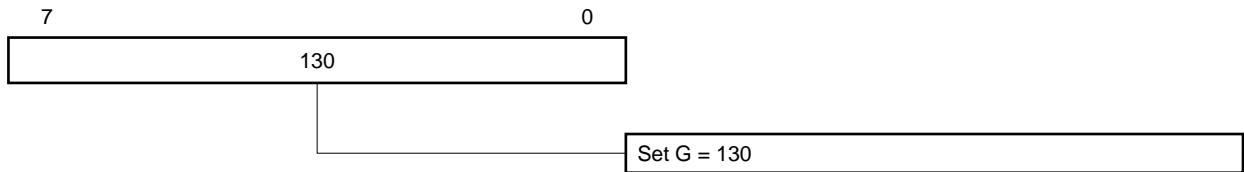
<1> SCM0 (serial mode register channel 0)



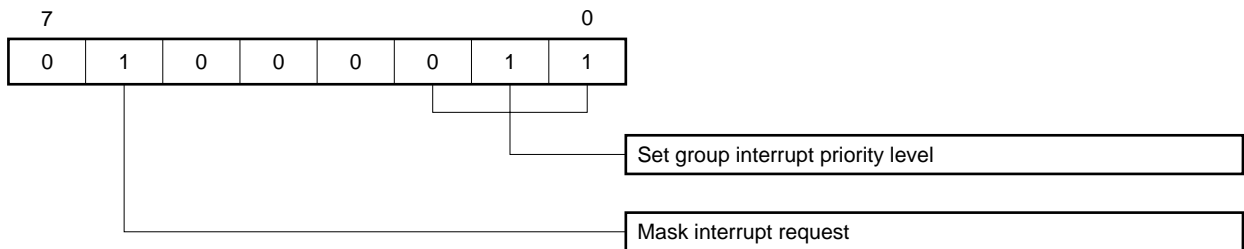
<2> SCC0 (serial control register 0)



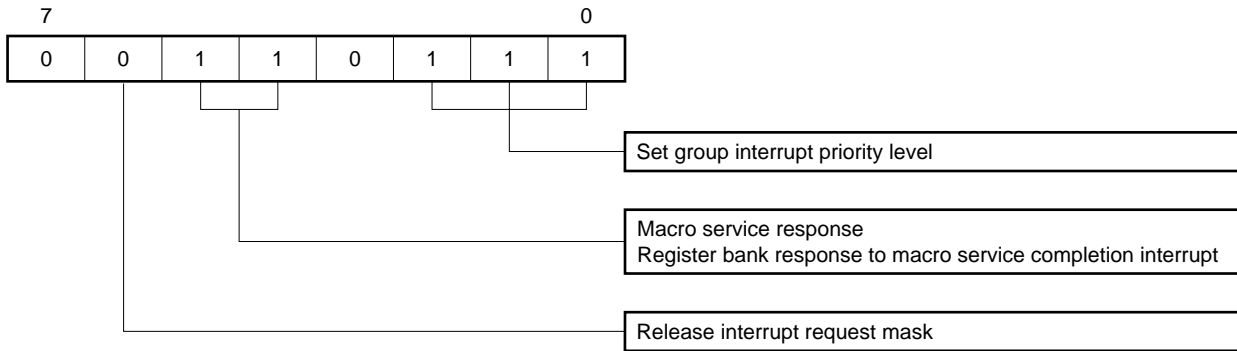
BRG0 (Baud rate generator register 0)



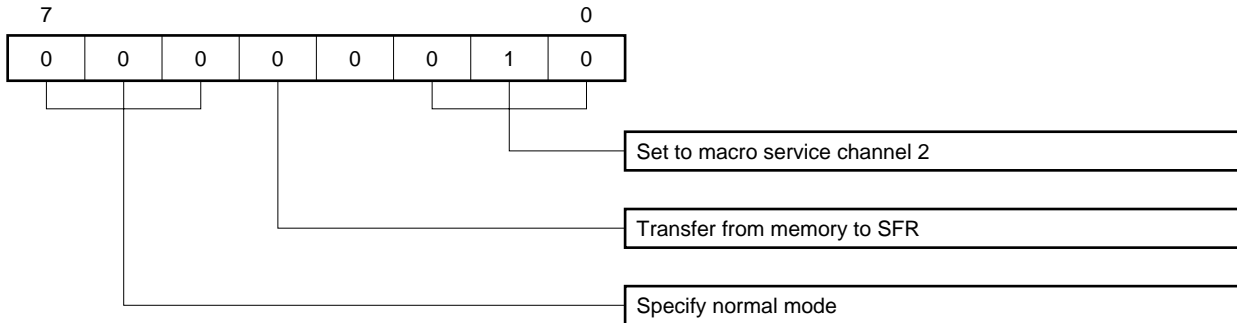
<3> SEIC0 (serial error interrupt request control register 0)



<4> STIC0 (serial transmission interrupt request control register 0)



<5> SRMS0 (macro service control register 0)



<6> Set the starting offset address of macro service channel 2 to the index register (IY).

<7> MSC [IY + 0]: Set number of transfers (15) using macro service.

<8> SFRP [IY + 1]: Set low-order byte of special function register (TxB0) address.

LOW: (RA70116-I) assembler's byte separator operand; return value of low-order byte in expression.

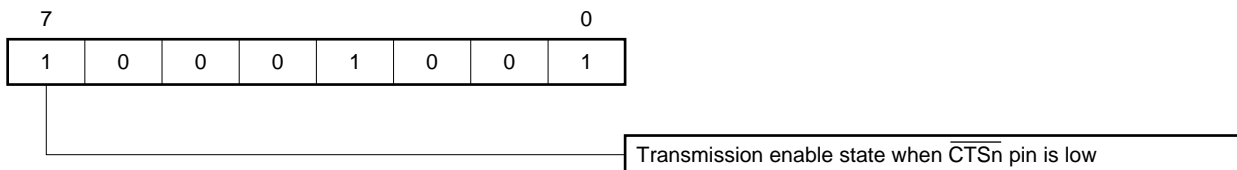
<9> MSP [IY + 4]: Set offset value of memory address to which data is sent by macro service.

<10> MSS [IY + 6]: Set segment value of memory address to which data is sent by macro service.

The memory address to which data is sent is $MSS \times 16 + MSP$.

The n value in brackets ([+ n]) indicates the offset from each macro service channel's start address.

<11> SCM0 (serial mode register channel 0)



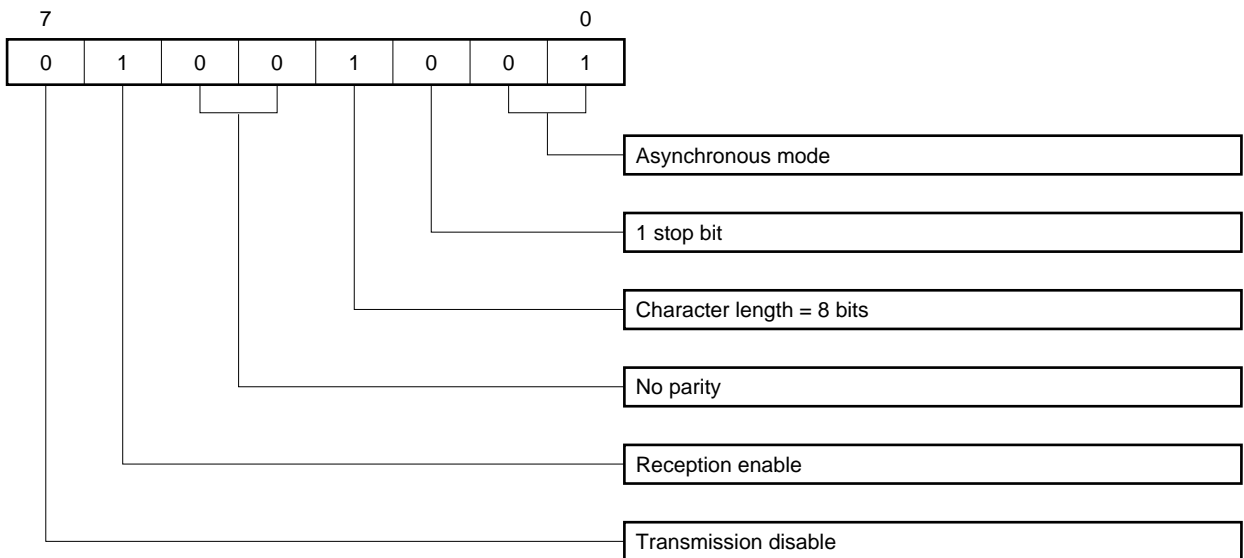
B.7.2 Character search mode (serial interface UART reception)

```

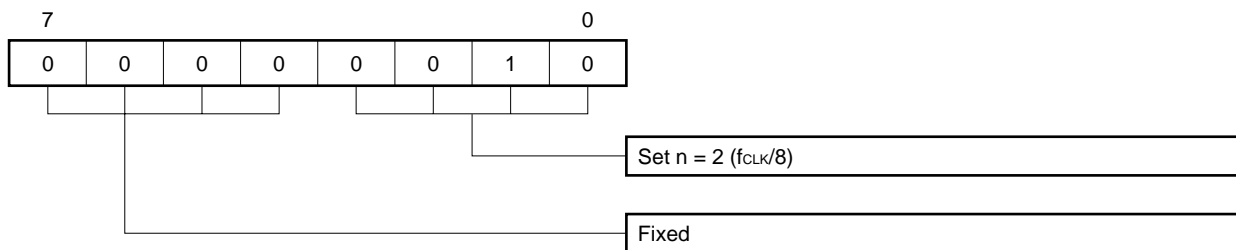
;
;
;+++++
;+++                               Special function register setting                               +++
;+++++
;
;
;      MOV      SCM0, 01001001B      ; <1>
;      MOV      SCC0, 00000010B      ; <2>
;      MOV      BRG0, 130             ;
;
;
;      MOV      SEIC0, 01000011B     ; <3>
;      MOV      SRIC0, 00110111B     ; <3>
;      MOV      STIC0, 01000111B     ; <3>
;
;
;*****
;***                               Initialization of macro service (channel 2) character search mode                               ***
;*****
;
;      MOV      SRMS0, 10000010B      ; <4>
;
;
;      MOV      IY, OFFSET REGBANK+8*2 ; <5>
;      MOV      BYTE PTR [IY], 15      ; <6>
;      MOV      BYTE PTR [IY+1], LOW RXB0 ; <7>
;      MOV      BYTE PTR [IY+2], 0AH    ; <8>
;      MOV      WORD PTR [IY+4], OFFSET BAR_CODE ; <9>
;      MOV      WORD PTR [IY+6], SEG BAR_CODE ; <10>

```

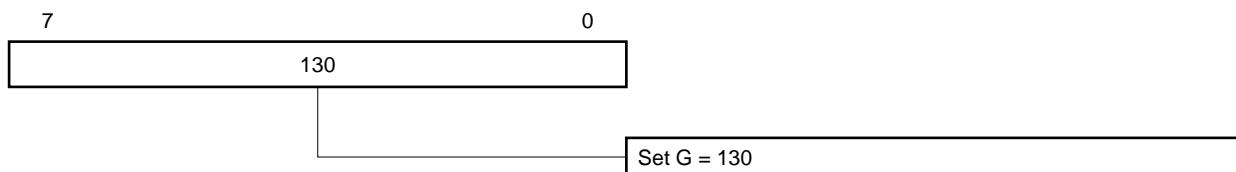
<1> SCM0 (serial mode register channel 0)



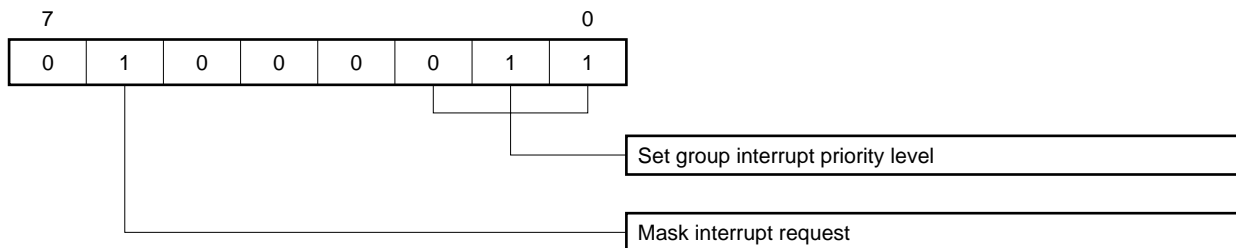
<2> SCC0 (serial control register 0)



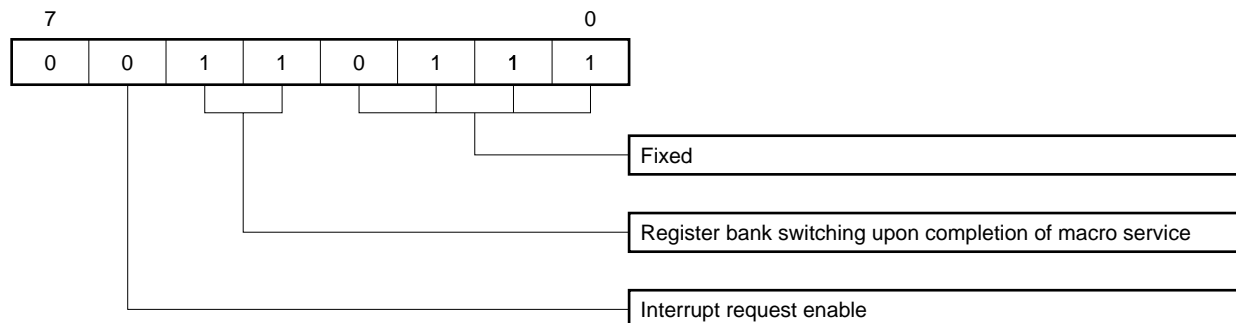
BRG0 (Baud rate generator register 0)



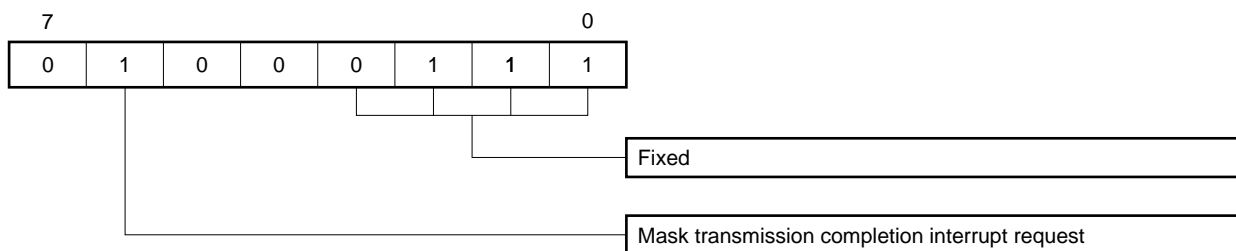
<3> SEIC0 (serial error interrupt request control register 0)



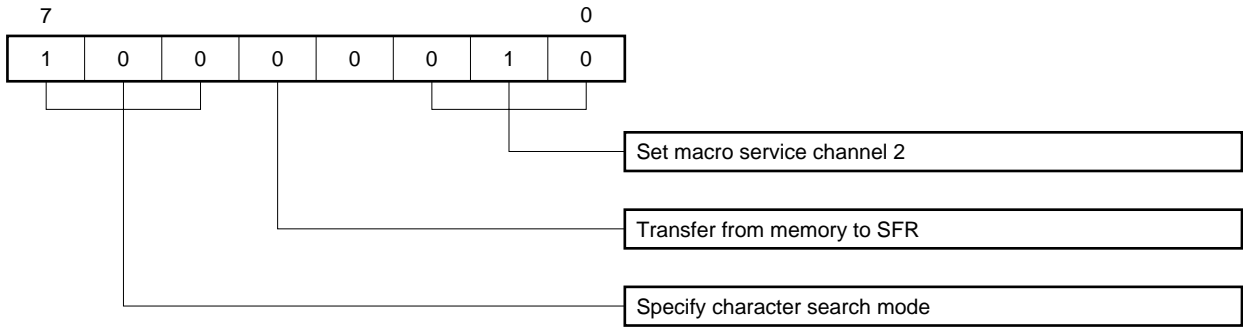
SRIC0 (serial reception interrupt request control register 0)



STIC0 (serial transmission interrupt request control register 0)



<4> SRMS0 (macro service control register 0)



<5> Set the starting of macro service channel 2 to the index register (IY).

<6> MSC [IY + 0]: Set number of transfers (15) using macro service.

<7> SFRP [IY + 1]: Set low-order byte of special function register (RxB0) address.

LOW: (RA70116-l) assembler's byte separator operand; return value of low-order byte in expression.

<8> SCHR [IY + 2]: Set 8-bit data (0AH) to be compared during character search mode.

<9> MSP [IY + 4]: Set offset value of memory address to which data is sent by macro service.

<10> MSS [IY + 6]: Set segment value of memory address to which data is sent by macro service.

The memory address to which data is sent is $MSS \times 16 + MSP$.

The n value in brackets ([+ n]) indicates the offset from each macro service channel's start address.

B.8 DMA Controller

Shown below is an example in which (MSG_TBL_SAR) is the starting address of the memory space where the source data to be transferred is stored and (MSG_TBL_DAR) is the starting address of the transfer destination (data storage memory).

B.8.1 Demand release mode

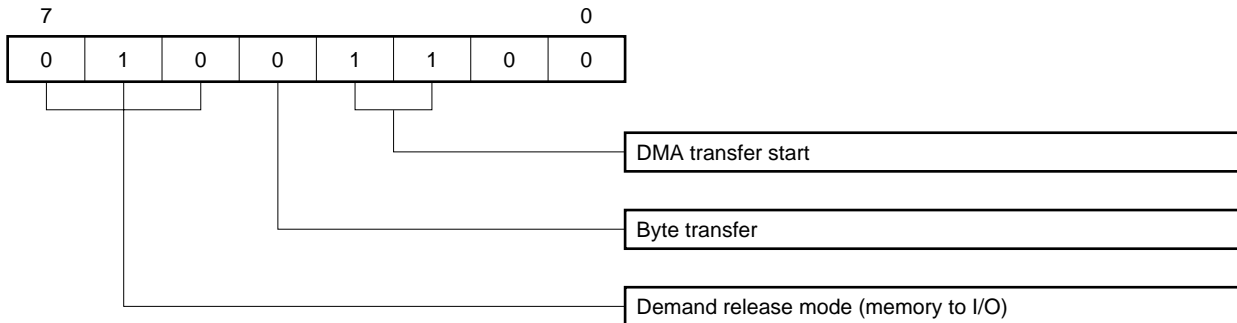
```

;
;
;*****
;***                               Initialization of DMA controller (channel 0)                               ***
;*****
;
;
;      MOV      DMAM0, 01001100B      ; <1>
;      MOV      DMAC0, 00000001B     ; <2>
;
;
;      MOV      IY, OFFSET REGBANK+8*0 ; <3>
;      MOV      WORD PTR [IY], OFFSET MSG_TBL_ADR ; <4>
;      MOV      AW, SEG MSG_TBL_ADR    ; <5>
;      MOV      BYTE PTR [IY+5], AH    ; <5>
;      MOV      WORD PTR [IY+6], 3600  ; <6>
;
;
;      :
;

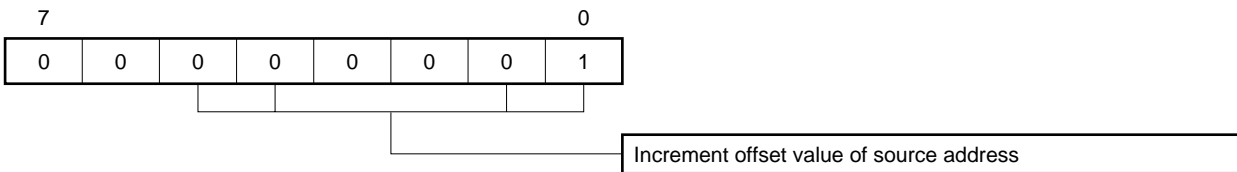
```

Remark Set the low-order eight bits of the segment address to 0.

<1> DMAM0 (DMA mode register 0)



<2> DMAC0 (DMA control register 0)



- <3> Set the starting address of DMA service channel 0 to index register (IY).
- <4> SAR [IY]: Set offset (low-order 16 bits) of source address for DMA transfer.
- <5> BARH [IY + 5]: Set segment (high-order eight bits) of source address for DMA transfer.
However, the low-order eight bits are fixed to 0.
- <6> TC [IY + 6]: Set DMA transfer count (3,600 times)
Because this is byte transfer, 3,600 bytes of data are required.

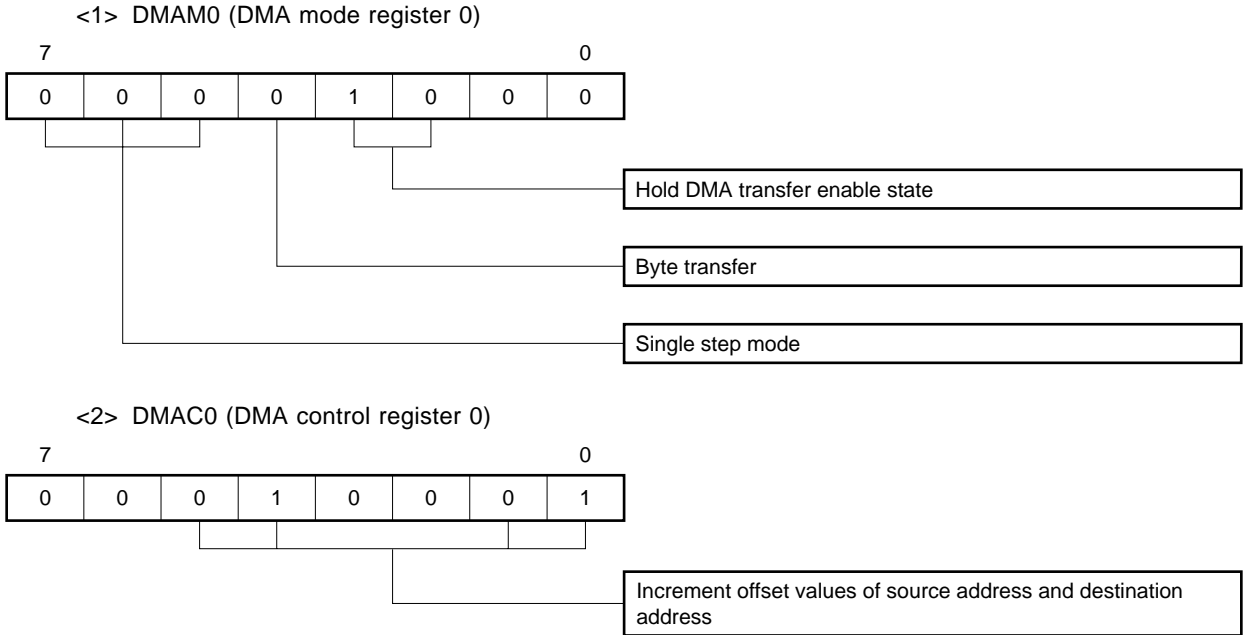
B.8.2 Single step mode

```

;
;*****
;***                Initialization of DMA controller (channel 0)                ***
;*****
;
;
;   MOV    DMAM0, 00001000B                ; <1>
;   MOV    DMAC0, 00010001B                ; <2>
;
;
;   MOV    IY, OFFSET REGBANK+8*0         ; <3>
;   MOV    WORD PTR [IY],  OFFSET MSG_TBL_SAR ; <4>
;   MOV    WORD PTR [IY+2], OFFSET MSG_TBL_DAR ; <5>
;   MOV    AW, SEG MSG_TBL_DAR            ; <6>
;   MOV    BYTE  PTR [IY+4], AH           ; <7>
;   MOV    AW, SEG MSG_TBL_SAR            ; <7>
;   MOV    BYTE  PTR [IY+5], AH           ; <7>
;   MOV    WORD PTR [IY+6], 3600          ; <8>
;
;
;   SET1   DMAM0, 2                        ;DMA transfer start
;
;

```

Remark Set the low-order eight bits of the segment address to 0.



- <3> Set the starting address of DMA service channel 0 to index register (IY).
- <4> SAR [IY]: Set offset (low-order 16 bits) of source address for DMA transfer.
- <5> DAR [IY + 2]: Set offset (low-order 16 bits) of destination address for DMA transfer.
- <6> DARH [IY + 4]: Set segment (high-order eight bits) of destination address for DMA transfer. However, the low-order eight bits are fixed to 0.
- <7> SARH [IY + 5]: Set segment (high-order eight bits) of source address for DMA transfer. However, the low-order eight bits are fixed to 0.
- <8> TC [IY + 6]: Set DMA transfer count (3,600 times) Because this is byte transfer, 3,600 bytes of data are required.

B.8.3 Burst mode

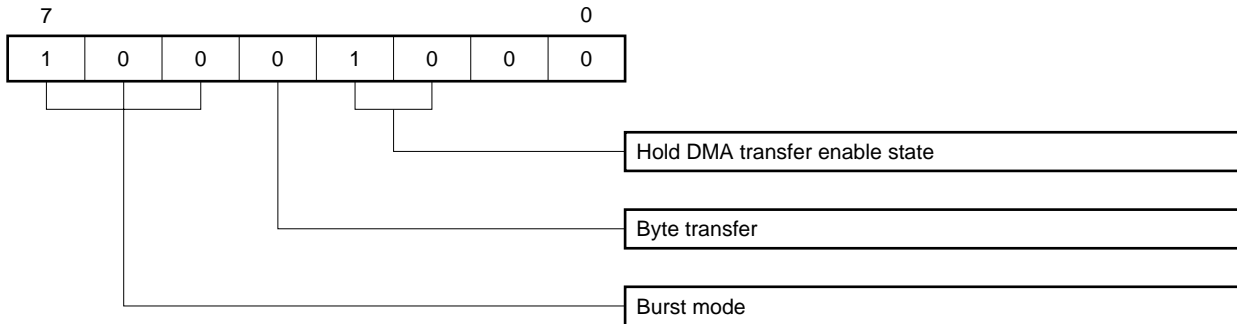
```

;
;*****
;***          Initialization of DMA controller (channel 0)          ***
;*****
;
;
MOV    DMAM0, 10001000B          ; <1>
MOV    DMAC0, 00010001B        ; <2>
;
;
MOV    IY, OFFSET REGBANK+8*0   ; <3>
MOV    WORD PTR [IY],  OFFSET MSG_TBL_SAR ; <4>
MOV    WORD PTR [IY+2], OFFSET MSG_TBL_DAR ; <5>
MOV    AW, SEG MSG_TBL_DAR      ; <6>
MOV    BYTE PTR [IY+4], AH      ; <7>
MOV    AW, SEG MSG_TBL_SAR      ; <7>
MOV    BYTE PTR [IY+5], AH      ; <7>
MOV    WORD PTR [IY+6], 3600    ; <8>
;
;
SET1   DMAM0, 2                  ;DMA transfer start
;
;

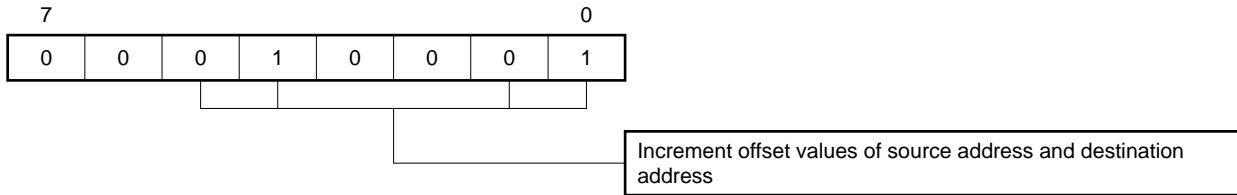
```

Remark Set the low-order eight bits of the segment address to 0.

<1> DMAM0 (DMA mode register 0)



<2> DMAC0 (DMA control register 0)



- <3> Set the starting address of DMA service channel 0 to index register (IY).
- <4> SAR [IY]: Set offset (low-order 16 bits) of source address for DMA transfer.
- <5> DAR [IY + 2]: Set offset (low-order 16 bits) of destination address for DMA transfer.
- <6> DARH [IY + 4]: Set segment (high-order eight bits) of destination address for DMA transfer.
However, the low-order eight bits are fixed to 0.
- <7> SARH [IY + 5]: Set segment (high-order eight bits) of source address for DMA transfer.
However, the low-order eight bits are fixed to 0.
- <8> TC [IY + 6]: Set DMA transfer count (3,600 times)
Because this is byte transfer, 3,600 bytes of data are required.

APPENDIX C Q & A

C.1 Internal CPU Function ... 304

Q.1.1 In memory space

- (1) If a word access is made to the offset address FFFFH in a segment, what does the real address become? ... 304
- (2) When the V25/V35 family is reset, the internal data area is located at FFE00H to FFFFFH. Does it overlap the special function register area and bootstrap address of the internal data area? ... 304

Q.1.2 What is the concept of the memory access time when the V25 family is interfaced with memory? ... 304

Q.1.3 In I/O space

- (1) Addresses FF00H to FFFFH are a reserved area. When is this area used? ... 305
- (2) If an I/O space address is not decoded with respect to the high-order eight bits (A8 to A15), what problem arises for its use? ... 305

Q.1.4 When using memory-mapped I/O

- (1) When the CPU internal memory is accessed, is the IDB register rewritten by the CPU rather than by the user? ... 305
- (2) When using a register bank for hardware interrupts, is a write made directly into the register bank area? ... 305

C.2 Interrupt Function ... 306

Q.2.1 Can multiple interrupts be made within the same group? ... 306

Q.2.2 For priority levels of external interrupts

- (1) If an external interrupt request of INTP0 or INTP1 input is made while an INT interrupt is being executed, is the external interrupt acknowledged? ... 307
- (2) If interrupt requests are input to INTP0 and INTP1 at the same time that an INT interrupt is being executed, how is the operation performed? ... 307
- (3) If INT is input while an INTP0 or INTP1 interrupt is being executed, is the INT interrupt acknowledged? ... 307

Q.2.3 Register bank switching function

- (1) Is the same register bank used for multiple interrupts within the same group? If so, can multiple interrupt servicing be performed? ... 309
- (2) What types of interrupts can the register bank switching function not be used for? ... 309
- (3) Does the CPU automatically switch register banks? If so, can the register bank contents before the register banks are switched be saved and restored? ... 309
- (4) Interrupt request registers PR0 to PR2 specify a new register bank. Within the same group, the value is fixed to 7 (PR0 to PR2 = 1, 1, 1) except for the highest priority. Is it fixed to register bank 7 except for the highest priority? ... 309

Q.2.4 The user's manual says "FINT instruction is executed just before termination of interrupt servicing except for NMI, INT, or software interrupts". However:

- (1) Is the FINT instruction required for input/output instruction interrupts and FPO instruction interrupts? ... 310
- (2) May the FINT instruction be executed in any case other than just before termination of interrupt servicing? ... 310

Q.2.5 When bit 7 (IF) of the interrupt request control register is set to 1 by software, how does the V25/V35 family operate (EI state)? ... 310

Q.2.6 The macro service function transfers data between the special function register area and memory space according to an interrupt request. However:

- (1) Can block transfer be executed by the macro service function? ... 311
- (2) Can data be transferred between I/O and memory without using any special function register? ... 311
- (3) When data is transferred between a special function register and memory, are the address and bus control signals externally output? ... 311
- (4) What are application examples of the macro service function? ... 311
- (5) What is the efficient use of macro service channels when a number of tasks are processed? ... 311
- (6) When the MSC value (transfer count by macro service) is 0, what vector does an interrupt occurrence go to? ... 311

Q.2.7 Can wait states be inserted into an $\overline{\text{INTAK}}$ cycle? ... 312**Q.2.8 To read or write one word in the V25 family, the word is divided into the low-order byte and high-order byte. When an interrupt request occurs between reading the low-order byte and reading the high-order byte, is the interrupt acknowledged? ... 312****C.3 Bus Control ... 313****Q.3.1 Memory bank configuration of V35 family**

- (1) Why are the $\overline{\text{UBE}}$ and $\overline{\text{MSTB}}$ pins used to select a chip in memory bank selection? Why is the $\overline{\text{UBE}}$ pin required for the $\overline{\text{MSTB}}$ pin to read the low-order address? ... 313
- (2) Are the A0 signal and $\overline{\text{UBE}}$ signal required in a read cycle? ... 313
- (3) Why is the physical address of the A18 bit output in the second bus cycle of address time division output in the memory cycle? ... 313
- (4) Which of the high-order and low-order bits of an address are output first? ... 313
- (5) Can a program be fetched in byte units? ... 313

Q.3.2 When the on-chip RAM area is accessed, can wait states be inserted into a bus cycle by setting the wait control register (WTC)? ... 313**Q.3.3 READY signal input**

- (1) When the number of wait states is fixed to 0, 1, or 2, if the READY signal goes low, does the CPU malfunction? ... 314
- (2) With the V25, when two TAW states are inserted, the READY pin is set low, and one TW state is inserted, what is the timing specification? ... 314
- (3) If READY signal input is made asynchronous (when the t_{SCRY} , t_{HCRY} condition is not satisfied), does the CPU malfunction? ... 314
- (4) Is it possible to first deactivate the READY signal (low) and activate (high) only when a bus cycle is escaped? ... 314

Q.3.4 In memory cycles

- (1) What are the differences between the $\overline{\text{MREQ}}$ signal and $\overline{\text{MSTB}}$ signal in the V25 family? ... 315
- (2) When the CPU accesses internal memory, are the $\overline{\text{MREQ}}$ and $\overline{\text{MSTB}}$ signals deactivated? ... 315
- (3) In the V35 family, the $\overline{\text{MSTB}}$ signal is prolonged by wait insertion at the memory read timing. Why isn't the $\overline{\text{MSTB}}$ signal prolonged by wait insertion in a memory write cycle? ... 315

Q.3.5 When moving from a DMA cycle to a CPU bus cycle (fetch or data access), is an idle cycle entered? ... 315

- Q.3.6** In the V25 family user's manual, data is read or written on the CLKOUT signal's falling edge in the I/O read/write cycle drawing. When designing, is it necessary to make the same drawing as in the user's manual? ... 316
- Q.3.7** In the V25 family memory read timing
- (1) Is it necessary to satisfy t_{DADR} , t_{DMRD} , and t_{DMSD} ? ... 316
 - (2) Is speed within the data delay time t_{DMRD} required from the \overline{MREQ} signal's falling edge? ... 316
 - (3) Is data read on the \overline{MREQ} signal's rising edge (t_{WMRL})? ... 316
- Q.3.8** Are t_{DAIS} and t_{DAMR} prolonged by wait insertion at the V25 family memory and I/O read/write cycle timing? ... 317

C.4 DMA Controller ... 318

- Q.4.1** What is the MIN. value of the required time from DMARQ signal input to \overline{DMAAK} signal output at the V25/V25S DMA transfer timing? ... 318
- Q.4.2** DMA mode register
- (1) IF the EDMA bit is 0 when the DMARQ signal is input, is a DMA request acknowledged? ... 319
 - (2) Is there a specification for DMARQ signal input for register setting? ... 319
 - (3) Can temporary stop of DMA transfer be controlled by setting the EDMA bit? ... 319
- Q.4.3** In the V25 family, how is an access to I/O determined in DMA transfer from memory to I/O? ... 319
- Q.4.4** Differences between V25/V35 and V25+/V35+
- (1) Do they differ in their DMA transfer addressing method? ... 320
 - (2) Is there a direct source of transfer rate improvement in DMA transfer on the V25+/V35+? ... 320
- Q.4.5** Can the number of transfer bytes of one DMA be specified? ... 320
- Q.4.6** When a DMA request is canceled, how does the CPU operate at the following timing (except in the demand release mode)? ... 321
- Q.4.7** On the V25, how much time is taken from the DMA transfer termination to a DMA interrupt occurrence? ... 322
- Q.4.8** In DMA transfer in the burst mode, if a refresh request is made during V25+/V35+ DMA transfer, what does the DMA timing become? ... 322
- Q.4.9** In the single step mode and the burst mode, can asynchronous DMA transfer processing be performed such that with the DMARQ signal always set high, DMA processing is controlled by setting the EDMA and TDMA bits of the DMA mode register? ... 323
- Q.4.10** In the one transfer mode
- (1) When a memory address is output to the address bus, where is an I/O address specified? ... 324
 - (2) If the DMARQ signal is active while the EDMA bit of the DMA mode register is changed from 0 to 1, is a DMA request acknowledged? ... 324
 - (3) In the user's manual, transfer from memory to I/O is the only type of transfer described in the one transfer mode. Does transfer from I/O to memory differ from transfer from memory to I/O only in the $\overline{R/W}$ signal level? For transfer from I/O to memory, is it necessary to address the DMA service channels? ... 324
 - (4) What is the MIN. value of the DMARQ signal pulse width to acknowledge a DMA request with the V25 family? ... 324
- Q.4.11** V25+/V35+ demand release mode
- (1) When consecutive transfer is executed one byte at a time under the DMARQ signal control, does the CPU bus cycle operate in any period other than DMA transfer cycle? ... 325
 - (2) What control is performed for DMA transfer by using DMARQ signal control? ... 325

C.5 Clock Generator ... 326**Q.5.1 Processor control register (PRC)**

- (1) If the oscillation frequency dividing ratio is changed by setting the PCK0 and PCK1 bits, is CPU operation affected? ... 326
- (2) If an internal RAM access is disabled by setting the PRC RAMEN bit, can the external memory of the same address be accessed? At that time, can the register bank be used? ... 326

C.6 Timer Unit ... 327**Q.6.1 When the timer unit is set to the interval timer mode**

- (1) Are timers 0 and 1 fixed to the 16-bit full count mode? ... 327
- (2) Is it possible to make the period set in the TM0 register of timer 0 a square wave and to output the square wave to the TOUT pin? ... 327

Q.6.2 If the system clock frequency (f_{CLK}) is 8 MHz in the interval timer mode on the V25, when TCLKn bit = 0 of the timer control register is used, what value is set in the MDn register to set the 20-ms interval timer value? ... 327**C.7 Serial Interface ... 328****Q.7.1 Does insertion of an idle time of one clock cycle or more between data units in data transmission/reception at a baud rate of 750 kbps or more mean insertion of one clock cycle between bits? ... 328****Q.7.2 To make a serial interface of the V25/V35 family with another microcomputer, data shifts one bit and is not restored due to mixing noise in the serial line, etc. What is a countermeasure for this symptom? ... 328****Q.7.3 When using the asynchronous mode for transmission on the V25 family serial interface**

- (1) How long does it take for a start bit to be actually output after a macro service is started by a program? ... 329
- (2) Assuming a baud rate of 4800 bps, how long does it take for a start bit to be actually output after data is set in a transmit buffer by a program? ... 329

Q.7.4 For transmission in the I/O interface mode, can a transmit clock be input from the outside? ... 330**Q.7.5 In the V35 family, what is the macro service response time (maximum transfer rate) in the I/O interface mode? ... 330****Q.7.6 In serial interface interrupt requests**

- (1) What is the condition for clearing (to 0) bit 7 of the interrupt request control register (SEFn, SRFn, STFn)? ... 331
- (2) What is the timing at which the SRFn bit is set to ON? ...331
- (3) Is register bank switching response enabled for serial interface interrupts? ...331
- (4) If a reception error interrupt occurs during reception completion interrupt servicing by a macro service, is the interrupt servicing executed so far then held pending? ...331

C.8 Standby Function ... 332

- Q.8.1** How long is the oscillation stabilization time if the STOP mode is released by NMI? ... 332
- Q.8.2** How much is the V_{DD} consumption current until the CPU operates, if the STOP mode is released by NMI? ... 332

C.9 Reset Function ... 333

- Q.9.1** How much is the V_{DD} consumption current if system reset is applied? ... 333
- Q.9.2** In the V35 family, when the low-to-high transition of the $\overline{\text{RESET}}$ signal is made to release system reset
 - (1) How long does it take until the CPU operates? ... 334
 - (2) How long does it take until the $\overline{\text{REFRQ}}$ signal is output? ... 334

C.10 Other ... 335

- Q.10.1** Can a program distinguish between the V25 and V25+ and between the V35 and V35+? ... 335
- Q.10.2** Why is the IC pin fixed high with an external pull-up resistor? ... 335
- Q.10.3** What do EA and T of “EA + 6 + T” represent in the instruction execution time (number of clock cycles)? ... 335
- Q.10.4** For transfer instruction MOV, why are no wait states inserted with the number of clock cycles for on-chip RAM access disable being “EA + 2” for “MOV mem reg”? ...336

C.1 Internal CPU Function

Q.1.1

In memory space

- (1) If a word access is made to the offset address FFFFH in a segment, what does the real address become?
- (2) When the V25/V35 family is reset, the internal data area is located at FFE00H to FFFFFH. Does it overlap the special function register area and bootstrap address of the internal data area?

A.1.1

- (1) If a word access is made across the segment boundary (offset address = address next to FFFFH) in the V25/V35 family, one byte in the segment (offset address = FFFFH) and one byte outside the segment (starting address of the next segment) are accessed. On the V20, V30, V40, and V50, address 0000H in the same segment is accessed, unlike in the V25/V35 family.
- ★ (2) The addresses overlap each other, but program fetch is not executed for the internal data area. When memory is accessed to fetch a program, bus control signals, such as \overline{MREQ} , $\overline{R/W}$, and \overline{MSTB} are always output to the external data area, thus the external memory is accessed.
Only a data access is made to the special function register area.

Q.1.2

What is the concept of the memory access time when the V25 family is interfaced with memory?

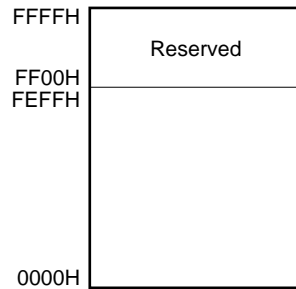
A.1.2

To interface the V25 family with memory, relate V25 family specification t_{DADR} to the address access time of the memory and t_{DMRD} to the \overline{OE} access time of the memory.

Q.1.3

In I/O space

- (1) Addresses FF00H to FFFFH are a reserved area. When is this area used?



- (2) If an I/O space address is not decoded with respect to the high-order eight bits (A8 to A15), what problem arises for its use?

A.1.3

- (1) The reserved area is an area which can be used for future product expansion, and is not used at present.
 (2) No problems arise for its use.

Q.1.4

When using memory-mapped I/O

- (1) When the CPU internal memory is accessed, is the IDB register rewritten by the CPU rather than by the user?
 (2) When using a register bank for hardware interrupts, is a write made directly into the register bank area?

A.1.4

- (1) The user needs to set the IDB register. A special function register access is programmed as a normal memory access is programmed.
 (2) The register bank is relocated to memory according to the IDB register. Write directly into the register bank area.

C.2 Interrupt Function

Q.2.1

Can multiple interrupts be made within the same group?

A.2.1

Multiple interrupts cannot be made within the same group. If interrupt flags are set within the same group, the interrupt having the highest priority level is acknowledged, and other interrupts within the same group are not acknowledged until a new FINT instruction is executed.

Therefore, for example, even if an INTP0 interrupt occurs while an external interrupt INTP1 is being executed, multiple interrupt servicing of INTP0 and INTP1 cannot be performed.

Q.2.2

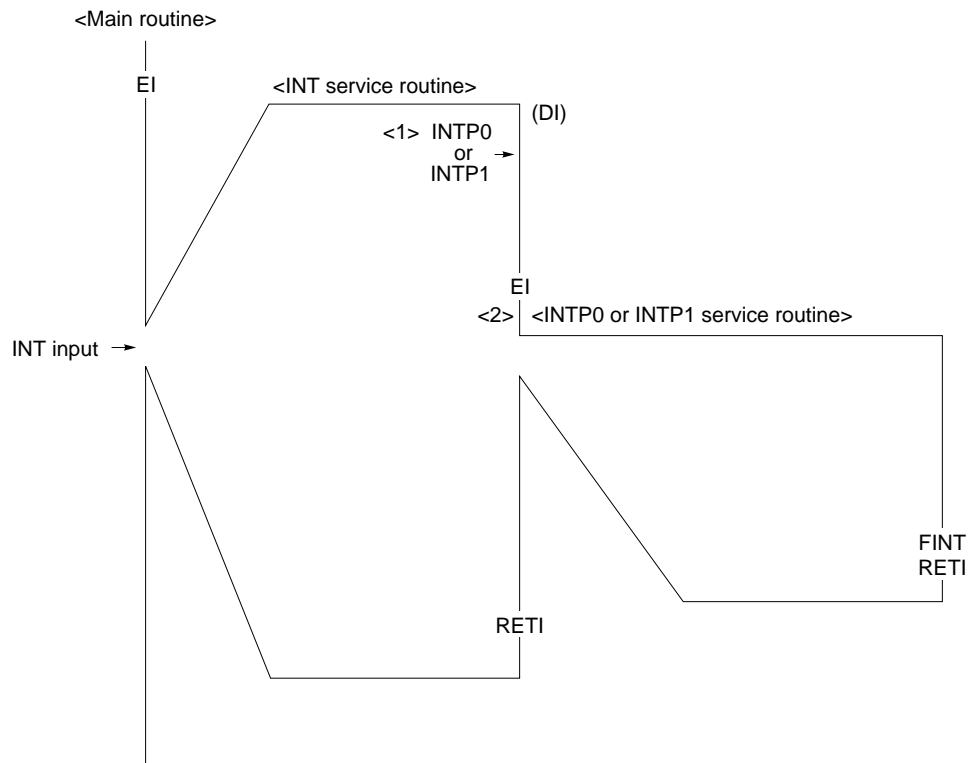
For priority levels of external interrupts

- (1) If an external interrupt request of INTP0 or INTP1 input is made while an INT interrupt is being executed, is the external interrupt acknowledged?
- (2) If interrupt requests are input to INTP0 and INTP1 at the same time that an INT interrupt is being executed, how is the operation performed?
- (3) If INT is input while an INTP0 or INTP1 interrupt is being executed, is the INT interrupt acknowledged?

A.2.2

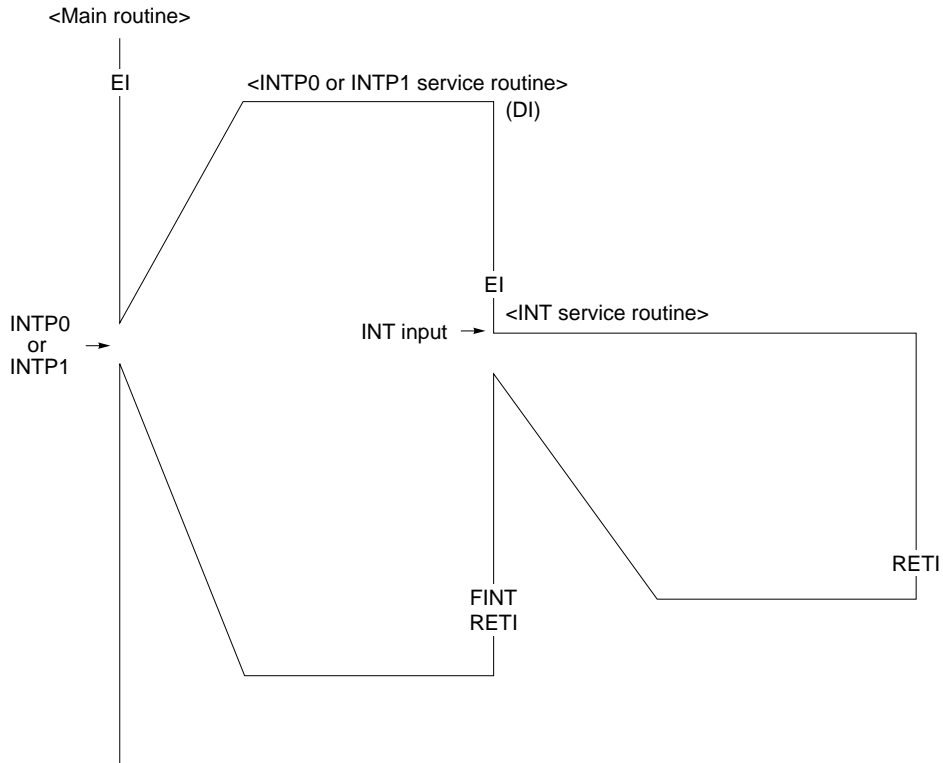
- (1) If INTP0 or INTP1 is input during execution of the INT interrupt service routine, it is acknowledged if the interrupt flag is set to the EI state.

An example of the INT and INTP0 or INTP1 service routine is given below.



- <1> Because INT is acknowledged and the interrupt disable state (DI) is set, even if an INTP0 or INTP1 interrupt request occurs, it is not acknowledged.
- <2> When the interrupt flag is set to EI (enable interrupt), execution of the service routine for the pending INTP0 or INTP1 interrupt is started.

- (2) If INTP0 and INTP1 are input at the same time during execution of the INT interrupt service routine, INTP0 is acknowledged and INTP1 is held pending because multiple interrupts within the same group cannot be executed.
- (3) The INT interrupt is not subject to multiple interrupt servicing control. Therefore, the INT interrupt is always acknowledged if the interrupt flag is set to the EI state. Shown below is an example of an INTP0 or INTP1 and INT interrupt service routine.



Q.2.3

Register bank switching function

- (1) Is the same register bank used for multiple interrupts within the same group? If so, can multiple interrupt servicing be performed?
- (2) What types of interrupts can the register bank switching function not be used for?
- (3) Does the CPU automatically switch register banks? If so, can the register bank contents before the register banks are switched be saved and restored?
- (4) Interrupt request registers PR0 to PR2 specify a new register bank. Within the same group, the value is fixed to 7 (PR0 to PR2 = 1, 1, 1) except for the highest priority. Is it fixed to register bank 7 except for the highest priority?

A.2.3

- (1) If a response by register bank switching is set for multiple interrupts within the same group, the interrupts respond by using the same register bank.
However, the V25/V35 cannot examine by occurrence of what interrupt a branch is taken to the service routine using the register bank, and cannot make two or more interrupts within the same group respond by register bank switching, but can perform the same interrupt servicing.
On the other hand, the V25+/V35+ can examine what interrupt is acknowledged by using the interrupt source register (IRQS) after register bank switching response.
- (2) The register bank switching function cannot be used for NMI, INT, INTTB, or software interrupts (except for the BRKCS instruction).
- (3) The CPU automatically switches register banks. At that time, PC and PSW are saved for the bank after switching, but other registers remain unchanged and are retained in the register bank before switching. Only PC and PSW are restored.
- (4) It is not fixed to bank 7.
The bank number at register bank switching of interrupt sources making up the same group is determined by the interrupt request control register (PR0 to PR2) of the interrupt source having the highest priority level within the same priority level group.

Q.2.4

The user's manual says "FINT instruction is executed just before termination of interrupt servicing except for NMI, INT, or software interrupts". However:

- (1) Is the FINT instruction required for input/output instruction interrupts and FPO instruction interrupts?
- (2) May the FINT instruction be executed in any case other than just before termination of interrupt servicing?

A.2.4

(1) The FINT instruction is not required. The FINT instruction is an instruction used as a signal of the interrupt servicing termination for the internal interrupt controller, and therefore it needs to be executed in returning from interrupt servicing subject to priority level control from the interrupt controller.

(2) Be sure to execute the FINT instruction just before the specified RETI (or RETRBI) instruction.

The V25/V35 family contains a special function register for managing the interrupt priority levels and the FINT instruction resets the least significant bit (highest priority level) of the set bits of the interrupt priority register (ISPR) in the interrupt controller. Therefore, when the FINT instruction is executed, an interrupt having the same or lower level as the interrupt currently being serviced is acknowledged. (However, no interrupt is acknowledged between the FINT instruction and the next instruction.)

If the FINT instruction is not executed, priority level control cannot be performed accurately.

Q.2.5

When bit 7 (IF) of the interrupt request control register is set to 1 by software, how does the V25/V35 family operate (EI state)?

A.2.5

An interrupt (or macro service) occurs as an interrupt request is generated by the hardware.

Q.2.6

The macro service function transfers data between the special function register area and memory space according to an interrupt request. However:

- (1) Can block transfer be executed by the macro service function?
- (2) Can data be transferred between I/O and memory without using any special function register?
- (3) When data is transferred between a special function register and memory, are the address and bus control signals externally output?
- (4) What are application examples of the macro service function?
- (5) What is the efficient use of macro service channels when a number of tasks are processed?
- (6) When the MSC value (transfer count by macro service) is 0, what vector does an interrupt occurrence go to?

A.2.6

- (1) When one interrupt request occurs, one data transfer (containing operation processing) is executed by the macro service function and block transfer cannot be executed by the function. However, block data processing can be performed by making a number of requests.
- (2) DMA is used for transfer between I/O and memory. Macro service is used for data transfer between a special function register and memory.
- (3) The address and bus control signals for memory are output when external memory is accessed. However, address and bus control signals are not output for special function registers.
- (4) Application examples are given below.
 - At given time intervals, data from a port is read and a search is made for the data.
 - At given time intervals, data is output to a port.
 - The interval time is changed according to an external request.
- (5) Because macro service need not be held pending by priority level control, it is generally efficient to start using the macro service channels at 0 and the register banks at 7.
- (6) If MSC is 0, the $\overline{\text{MS/INT}}$ bit of the interrupt request control register is set to 0. If the bit is 0, a vectored interrupt or an interrupt due to register bank switching occurs, depending on how the ENCS bit is set. When the ENCS bit is 0, a vectored interrupt is used, and when set to 1 the register bank switching function is used.

Q.2.7

Can wait states be inserted into an $\overline{\text{INTAK}}$ cycle?

A.2.7

Wait states cannot be inserted into the $\overline{\text{INTAK}}$ cycle.

In the V25/V35 family, the external interrupt controller is positioned as an additional function (for expansion) and the on-chip wait controller does not contain a wait state insertion function for interrupt acknowledge cycles.

Q.2.8

To read or write one word in the V25 family, the word is divided into the low-order byte and high-order byte. When an interrupt request occurs between reading the low-order byte and reading the high-order byte, is the interrupt acknowledged?

A.2.8

The interrupt request is not acknowledged until execution of one instruction terminates.

C.3 Bus Control

Q.3.1

Memory bank configuration of V35 family

- (1) Why are the \overline{UBE} and \overline{MSTB} pins used to select a chip in memory bank selection? Why is the \overline{UBE} pin required for the \overline{MSTB} pin to read the low-order address?
- (2) Are the A0 signal and \overline{UBE} signal required in a read cycle?
- (3) Why is the physical address of the A18 bit output in the second bus cycle of address time division output in the memory cycle?
- (4) Which of the high-order and low-order bits of an address are output first?
- (5) Can a program be fetched in byte units?

A.3.1

- (1) The \overline{UBE} pin is multiplexed with the A18 pin, and is activated at the timing when the \overline{MSTB} pin is activated. Therefore, for high-order memory bank chip selection, the \overline{UBE} pin and \overline{MSTB} pin are ANDed with each other.
The \overline{UBE} pin is required to access only memory's high-order bank or memory's low-order bank during write operation into odd addresses in word units (read operation in byte units is performed twice) and write operation in byte units.
- (2) The A0 signal and \overline{UBE} signal are not required in read cycles.
- (3) The A18 bit is output in order to output the address as early as possible in limitation of the number of bits.
- (4) The A9 to A19 bits and A0 bit are output first.
- (5) A program is always fetched in word units by signals. When an odd address is accessed, only the high-order byte of the address is read (the low-order byte becomes invalid).

★

Q.3.2

When the on-chip RAM area is accessed, can wait states be inserted into a bus cycle by setting the wait control register (WTC)?

★

A.3.2

The on-chip RAM area is always accessed with no wait, independently of how programmable wait is set.

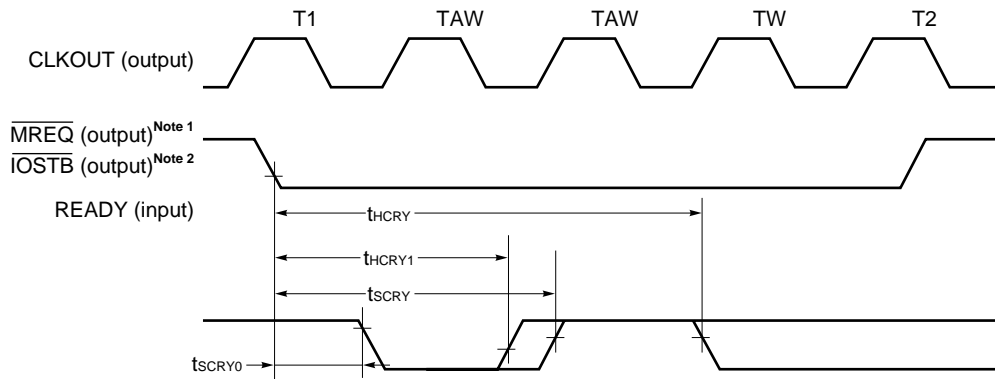
Q.3.3

READY signal input

- (1) When the number of wait states is fixed to 0, 1, or 2, if the READY signal goes low, does the CPU malfunction?
- (2) With the V25, when two TAW states are inserted, the READY pin is set low, and one TW state is inserted, what is the timing specification?
- (3) If READY signal input is made asynchronous (when the t_{SCRY} , t_{HCRY} condition is not satisfied), does the CPU malfunction?
- (4) Is it possible to first deactivate the READY signal (low) and activate (high) only when a bus cycle is escaped?

A.3.3

- (1) When the number of wait states is fixed to 0, 1, or 2, READY signal input is ignored. Therefore, the CPU operates normally.
- (2) For V25 READY signal input, satisfy the t_{SCRY} and t_{HCRY} specifications.



Notes 1. In memory cycle
2. In I/O cycle

- (3) If READY signal input is made asynchronous, the state in which a wait state is entered or the state in which no wait is entered is set, and the CPU does not malfunction.
- (4) It is possible. Even if the READY signal starts as inactive (low) and is active (high) only when a bus cycle is skipped, no problems will arise.

Q.3.4

In memory cycles

- (1) What are the differences between the $\overline{\text{MREQ}}$ signal and $\overline{\text{MSTB}}$ signal in the V25 family?
- (2) When the CPU accesses internal memory, are the $\overline{\text{MREQ}}$ and $\overline{\text{MSTB}}$ signals deactivated?
- (3) In the V35 family, the $\overline{\text{MSTB}}$ signal is prolonged by wait insertion at the memory read timing. Why isn't the $\overline{\text{MSTB}}$ signal prolonged by wait insertion in a memory write cycle?

A.3.4

- (1) To interface the V25 family with memory, an address is decoded to generate a memory chip select signal. At that time, the $\overline{\text{MREQ}}$ signal controls memory $\overline{\text{OE}}$.
The $\overline{\text{MSTB}}$ signal is used for an interface with DRAM. The $\overline{\text{MSTB}}$ signal controls the DRAM $\overline{\text{CAS}}$ signal. It is not necessary to use the $\overline{\text{MSTB}}$ signal for other than the DRAM interface.
- (2) When the CPU accesses internal memory, the $\overline{\text{MREQ}}$ signal and $\overline{\text{MSTB}}$ signal are deactivated (high).
- (3) The width of the $\overline{\text{MSTB}}$ signal is fixed regardless of whether or not a wait is inserted in the write cycle. With the V35 family, to interface DRAM, the $\overline{\text{CAS}}$ signal (memory input signal) is generated by the $\overline{\text{MSTB}}$ signal. To execute DMA transfer from I/O (slow output) to memory, the $\overline{\text{MSTB}}$ signal is delayed to write data into DRAM on the $\overline{\text{CAS}}$ signal's falling edge ($\overline{\text{MSTB}}$ signal's falling edge).

Q.3.5

When moving from a DMA cycle to a CPU bus cycle (fetch or data access), is an idle cycle entered?

A.3.5

No idle cycle is entered.

If a program is already prefetched at the DMA cycle termination, the next bus cycle (prefetch) is executed without entering an idle cycle.

Q.3.6

In the V25 family user's manual, data is read or written on the CLKOUT signal's falling edge in the I/O read/write cycle drawing. When designing, is it necessary to make the same drawing as in the user's manual?

A.3.6

For the I/O read/write cycle timing, refer to the AC characteristics described on the Data Sheet. When designing, the CLKOUT signal need not be used. The I/O read/write cycle timing is generated by the $\overline{\text{IOSTB}}$ signal.

Q.3.7

In the V25 family memory read timing

- (1) Is it necessary to satisfy t_{DADR} , t_{DMRD} , and t_{DMSD} ?
- (2) Is speed within the data delay time t_{DMRD} required from the $\overline{\text{MREQ}}$ signal's falling edge?
- (3) Is data read on the $\overline{\text{MREQ}}$ signal's rising edge (t_{WMRL})?

A.3.7

- (1) t_{DADR} , t_{DMRD} , and t_{DMSD} must be satisfied. (See section **Q.1.2**)
- (2) Speed within t_{DMRD} is required.
- (3) Data is read on the $\overline{\text{MREQ}}$ signal's rising edge.

Q.3.8

Are t_{DAIS} and t_{DAMR} prolonged by wait insertion at the V25 family memory and I/O read/write cycle timing?

A.3.8

t_{DAIS} and t_{DAMR} are not prolonged, even if a wait is inserted.

Remark If the AC characteristics change by wait insertion, the specification is represented by a function of the number (n) of wait states.

C.4 DMA Controller

Q.4.1

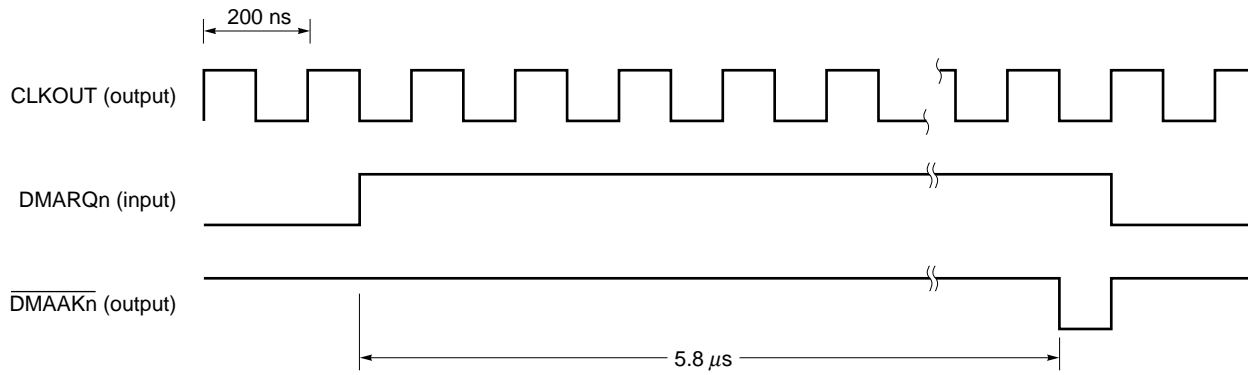
What is the MIN. value of the required time from DMARQ signal input to $\overline{\text{DMAAK}}$ signal output at the V25/V25S DMA transfer timing?

A.4.1

The MIN. value of the required time from DMARQ signal input to $\overline{\text{DMAAK}}$ signal output on the V25/V25S is: N denotes the number of required clock cycles for the instruction being executed.

- One transfer mode: 29 + N clock cycles
- Other modes: 27 + N clock cycles

At the following timing, $29 \times 200 \text{ ns} = 5.8 \mu\text{s}$ is required until the $\overline{\text{DMAAK}}$ signal is output when one clock cycle is 200 ns in the one transfer mode (with no wait).



(when bus hold function, refresh function is not used)

Q.4.2

DMA mode register

- (1) If the EDMA bit is 0 when the DMARQ signal is input, is a DMA request acknowledged?
- (2) Is there a specification for DMARQ signal input for register setting?
- (3) Can temporary stop of DMA transfer be controlled by setting the EDMA bit?

A.4.2

- (1) If the EDMA bit of the DMA mode register is 0, DMA requests (DMARQ signal input) are ignored.
- (2) The input delay time of the DMARQ signal for register setting is not specified.
- (3) It can be controlled. If the EDMA bit is set to 0 to temporarily stop DMA transfer and then is set to 1, DMA transfer can be restarted.

Q.4.3

In the V25 family, how is an access to I/O determined in DMA transfer from memory to I/O?

A.4.3

I/O is accessed by the $\overline{\text{DMAAK}}$ signal.

In DMA transfer between I/O and memory, the I/O which becomes the destination or source must be fixed by the hardware. In response to the $\overline{\text{DMAAK}}$ signal, the I/O recognizes that a DMA response follows.

Q.4.4

Differences between V25/V35 and V25+/V35+

- (1) Do they differ in their DMA transfer addressing method?
- (2) Is there a direct source of transfer rate improvement in DMA transfer on the V25+/V35+?

A.4.4

- (1) The V25/V35 uses the segment specification method; the V25+/V35+ uses the linear addressing method.
- (2) A direct source of transfer rate improvement is as to whether or not a dummy cycle exists on the timing chart.

The V25/V35 starts a DMA transfer cycle by the microprogram (basic software in the CPU) and performs request acknowledgment and transfer address control by software. Therefore, a dummy cycle is inserted before and after the DMA bus cycle.

On the other hand, the V25+/V35+ performs transfer processing by dedicated hardware, and therefore performs processing instantaneously as compared with the V25/V35; for this reason, no dummy cycle is inserted.

Because of these differences, the V25+/V35+ enables a drastic improvement in DMA response and transfer rates.

Q.4.5

Can the number of transfer bytes of one DMA be specified?

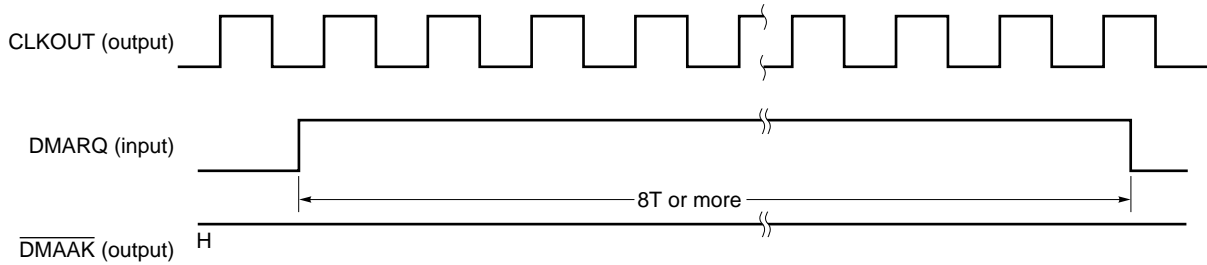
A.4.5

The number of transfer bytes and words for DMA transfer can be specified by setting the terminal counter (TC). In the byte transfer mode, 64 Kbytes can be transferred consecutively; in the word transfer mode, 128 Kbytes can be transferred consecutively.

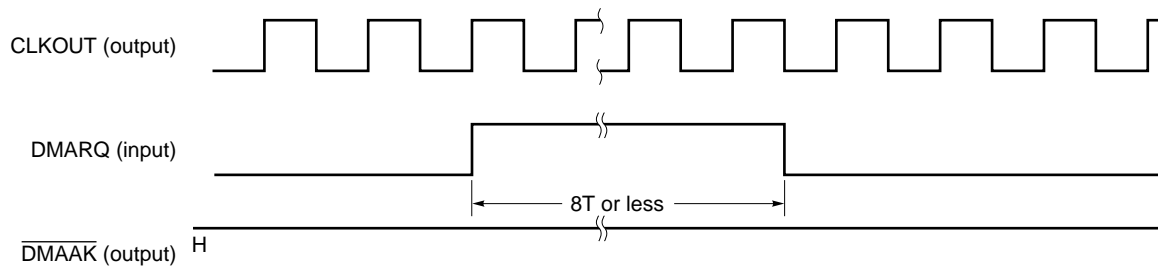
Q.4.6

When a DMA request is canceled, how does the CPU operate at the following timing (except in the demand release mode)?

<1> When the DMARQ signal is active for 8T or more and the DMA request is canceled before the $\overline{\text{DMAAK}}$ signal is output



<2> When the DMARQ signal is active within 8T (containing a short pulse) and the DMA request is canceled before the $\overline{\text{DMAAK}}$ signal is output

**A.4.6**

<1> Because the DMARQ signal is received on the DMARQ rising edge (when in single step, burst, or one transfer mode), the DMA request is acknowledged.

<2> The CPU does not operate abnormally, but may not acknowledge the DMA request.

Q.4.7

On the V25, how much time is taken from the DMA transfer termination to a DMA interrupt occurrence?

A.4.7

After setting DMA interrupt request control register DF0, DF1, the V25 executes the first instruction of the interrupt service routine in 76 clock cycles (MIN.) for a vectored interrupt or in 45 clock cycles (MIN.) for a register bank switching interrupt.

However, depending on the bus state at the last DMA transfer with TC = 0, program instruction fetch may be inserted before the interrupt acknowledge cycle starts just after the DMA transfer. At that time, interrupt servicing is delayed as long as the instruction fetch clock cycles.

Q.4.8

In DMA transfer in the burst mode, if a refresh request is made during V25+/V35+ DMA transfer, what does the DMA timing become?

A.4.8

If a refresh request is made during DMA transfer in the burst mode, a refresh cycle is inserted. A refresh cycle is inserted between 1-byte transfer cycles. At that time, no idle cycle is entered before or after it.

Q.4.9

In the single step mode and the burst mode, can asynchronous DMA transfer processing be performed such that with the DMARQ signal always set high, DMA processing is controlled by setting the EDMA and TDMA bits of the DMA mode register?

A.4.9

In the single step mode, it can be controlled (stopped) by setting the EDMA bit.

In the burst mode, once DMA transfer starts, instruction processing containing manipulation of the EDMA and TDMA bits is not performed, and therefore the DMA transfer cannot be stopped.

These modes are started on the rising edge of DMARQ signal input or by manipulating the TDMA bit. At that time, the EDMA bit must be set to the enable state.

For asynchronous operation, DMA transfer is also started by fixing the DMARQ signal high and manipulating the PMC2 register from the port mode to the control mode. However, TDMA bit control is recommended.

Q.4.10

In the one transfer mode

- (1) When a memory address is output to the address bus, where is an I/O address specified?
- (2) If the DMARQ signal is active while the EDMA bit of the DMA mode register is changed from 0 to 1, is a DMA request acknowledged?
- (3) In the user's manual, transfer from memory to I/O is the only type of transfer described in the one transfer mode. Does transfer from I/O to memory differ from transfer from memory to I/O only in the R/\overline{W} signal level? For transfer from I/O to memory, is it necessary to address the DMA service channels?
- (4) What is the MIN. value of the DMARQ signal pulse width to acknowledge a DMA request with the V25 family?

A.4.10

- (1) In transfer from memory to I/O in the one transfer mode, no I/O address is output. When DMA transfer is executed, the I/O requesting the DMA transfer is determined by the hardware. Therefore, if the \overline{DMAAK} signal is used as an I/O chip select signal, I/O can be selected.
- (2) The DMA request is not acknowledged. In the one transfer mode, when the EDMA bit is 0, DMA requests are ignored.
- (3) In the one transfer mode, transfer from I/O to memory differs from transfer from memory to I/O only in the R/\overline{W} signal output level. In transfer between I/O and memory, I/O cannot be addressed by software, therefore any DMA service channel setting is insignificant.
- (4) It is specified by AC characteristic t_{WIQH} .

Q.4.11

V25+/V35+ demand release mode

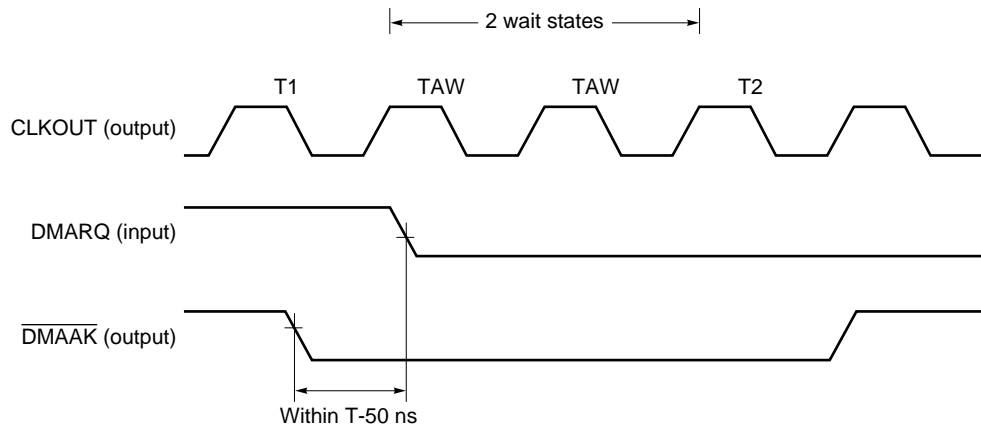
- (1) When consecutive transfer is executed one byte at a time under the DMARQ signal control, does the CPU bus cycle operate in any period other than DMA transfer cycle?
- (2) What control is performed for DMA transfer by using DMARQ signal control?

A.4.11

- (1) The V25+/V35+ processes DMA transfer using dedicated hardware, and therefore the CPU bus cycle operates in any period other than the DMA transfer cycle.
- (2) To execute DMA transfer in the demand release mode, stop control must be performed by the DMARQ signal. In the stop control, the number of DMA transfer cycle wait states must be two or more and the DMARQ signal must be turned off within (T-50) ns (t_{SDADQ}) after the $\overline{\text{DMAAK}}$ signal's falling edge in the last cycle before the transfer stops. (See below.)

Therefore, to ensure that the transfer is stopped in the last DMA transfer cycle, the DMARQ signal exceeding (T-50) ns is masked after the $\overline{\text{DMAAK}}$ signal's falling edge.

To ensure that DMA is started once or more, hold the DMARQ signal high until the $\overline{\text{DMAAK}}$ signal's falling edge.



C.5 Clock Generator

Q.5.1

Processor control register (PRC)

- (1) If the oscillation frequency dividing ratio is changed by setting the PCK0 and PCK1 bits, is CPU operation affected?
- (2) If an internal RAM access is disabled by setting the PRC RAMEN bit, can the external memory of the same address be accessed? At that time, can the register bank be used?

A.5.1

- (1) Even if the oscillation frequency dividing ratio is changed by setting the PCK0 and PCK1 bits, no problems arise in CPU operation. The frequency can be changed by software.
- (2) If the RAMEN bit is set to 0 to disable accessing the internal RAM, external memory is accessed regardless of the addresses.
The register bank can be used. A data access cannot be made as memory, but the register bank can be referenced as registers.

C.6 Timer Unit

Q.6.1

When the timer unit is set to the interval timer mode

- (1) Are timers 0 and 1 fixed to the 16-bit full count mode?
- (2) Is it possible to make the period set in the TM0 register of timer 0 a square wave and output the square wave to the TOUT pin?

A.6.1

- (1) Timers 0 and 1 are not fixed to the 16-bit full count mode. Any desired count operation can be performed by setting count values in the MD0 and MD1 registers.
- (2) In timer 0 in the interval timer mode, the TM0 register serves as a timer (counter) and a period is set in the MD0 bit. The square wave inverted in the period can be output to the TOUT pin. Square waves that are not 50% duty square waves cannot be output.

Q.6.2

If the system clock frequency (f_{CLK}) is 8 MHz in the interval timer mode on the V25, when TCLKn bit = 0 of the timer control register is used, what value is set in the MDn register to set the 20-ms interval timer value?

A.6.2

To set the 20-ms interval timer value when $f_{CLK} = 8$ MHz and TCLKn bit = 0, set the MDn register value to 682BH (26667).

Calculation example

TCLKn = 0: Count clock $f_{CLK}/6$

For use with 8 MHz, $8 \text{ MHz}/6 \approx 1.33 \text{ MHz}$ (clock period: $0.75 \mu\text{s}$)

The 20-ms interval timer counts as follows.

$$\frac{20 \text{ ms}}{0.75 \mu\text{s}} \approx 26667$$

Set MDn = 682BH (26667).

C.7 Serial Interface

Q.7.1

Does insertion of an idle time of one clock cycle or more between data units in data transmission/reception at a baud rate of 750 kbps or more mean insertion of one clock cycle between bits?

A.7.1

Insertion of the idle time of one clock cycle or more between data units means that the idle time of one clock cycle or more (shift clock) is required for each one-byte data transfer. In the asynchronous mode, an idle cycle with high level is required for one clock cycle or more. In the asynchronous mode, set one stop bit for V25/V35 family reception and two stop bits for the serial controller at the transmitting party for interfacing.

Q.7.2

To make a serial interface of the V25/V35 family with another microcomputer, data shifts one bit and is not restored due to mixing noise in the serial line, etc. What is a countermeasure for this symptom?

A.7.2

If data bit shift occurs due to noise, the serial register must be cleared and transfer must be executed again from the beginning.

To clear the serial register, perform either of the following.

<1> Write into the SCC0 register

<2> Change the MD0 bit of the SCM0 register (I/O interface mode → asynchronous mode → I/O interface mode)

Q.7.3

When using the asynchronous mode for transmission on the V25 family serial interface

- (1) How long does it take for a start bit to be actually output after a macro service is started by a program?
- (2) Assuming a baud rate of 4800 bps, how long does it take for a start bit to be actually output after data is set in a transmit buffer by a program?

A.7.3

- (1) Described below is how to find the approximate time when on-chip RAM is enabled and byte transfer is executed.
 - <1> First, the IF flag is set at the execution start timing of the next instruction to the IF flag setting instruction.
 - <2> To execute a macro service (in normal mode)
The macro service is executed 11 clock cycles (MIN.) after the IF flag is set. Furthermore, it takes $(24 + W)$ clock cycles to execute data transfer from memory to a transmit buffer (TxBn). (W is the number of wait states.)
 - <3> To execute a macro service (in character search mode)
The macro service is executed 11 clock cycles after the IF flag is set. Furthermore, it takes $(27 + W)$ clock cycles to execute data transfer from memory to a transmit buffer (TxBn), as in <2> above.
 - <4> When the macro service terminates and execution of the next instruction starts, data transfer to the transmit buffer (TxBn) executed in <2> or <3> is actually executed.
 - <5> A start bit is output on the rising edge of the next shift clock cycle after step <4> terminates. The approximate time is found from the time of $\text{<2>} + \text{<5>}$ or $\text{<3>} + \text{<5>}$ if <1> and <4> can be ignored.
- (2) When a write instruction into a transmit buffer (TxBn) is executed, data is transferred to the transmit buffer at the start timing of the next instruction. The serial interface operates independently of the instruction execution (system clock) and a start bit is output $1/f_{\text{CLK}}$ (seconds) to $1/4800$ (seconds) after.

Q.7.4

For transmission in the I/O interface mode, can a transmit clock be input from the outside?

A.7.4

External transmit clocks cannot be input.

Q.7.5

In the V35 family, what is the macro service response time (maximum transfer rate) in the I/O interface mode?

A.7.5

The macro service response time used for the serial interface (request occurrence → transfer execution start) is as follows. (N is the remaining number of clock cycles of the instruction being executed when a request is acknowledged.)

For transmission and on-chip RAM enable

- Normal mode: $36 + N$ to $52 + N$ clock cycles
- Character search mode: $39 + N$ to $55 + N$ clock cycles

Therefore, the data exchange speed with a transmit buffer is limited by the response time.

This also applies to reception (but the number of clock cycles differs).

However, if bus hold, refresh cycle, etc., are contained, it is beyond the range.

Q.7.6

In serial interface interrupt requests

- (1) What is the condition for clearing (to 0) bit 7 of the interrupt request control register (SEFn, SRFn, STFn)?
- (2) What is the timing at which the SRFn bit is set to ON?
- (3) Is register bank switching response enabled for serial interface interrupts?
- (4) If a reception error interrupt occurs during reception completion interrupt servicing by a macro service, is the interrupt servicing executed so far then held pending?

A.7.6

- (1) The SEFn, SRFn, and STFn bits are automatically cleared to 0 in their respective interrupt acknowledge cycles.
- (2) The SRFn bit is set to ON when data is set in a receive buffer (RxBn) from the shift register.
- (3) Register bank switching response is enabled for serial interface interrupts.
- (4) The reception error interrupt has a higher priority level than the reception completion interrupt or transmission completion interrupt. The reception completion interrupt is held pending while the reception error interrupt is being serviced.

Therefore, for reception interrupt servicing by a macro service, the pending state is released by executing a FINT instruction, after which macro service is executed and data in the reception bank is transferred.

C.8 Standby Function

Q.8.1

How long is the oscillation stabilization time if the STOP mode is released by NMI?

A.8.1

The oscillation stabilization time is 30 ms.

Q.8.2

How much is the V_{DD} consumption current until the CPU operates, if the STOP mode is released by NMI?

A.8.2

The V_{DD} consumption current becomes the same as that in the HALT mode.

C.9 Reset Function**Q.9.1**

How much is the V_{DD} consumption current if system reset is applied?

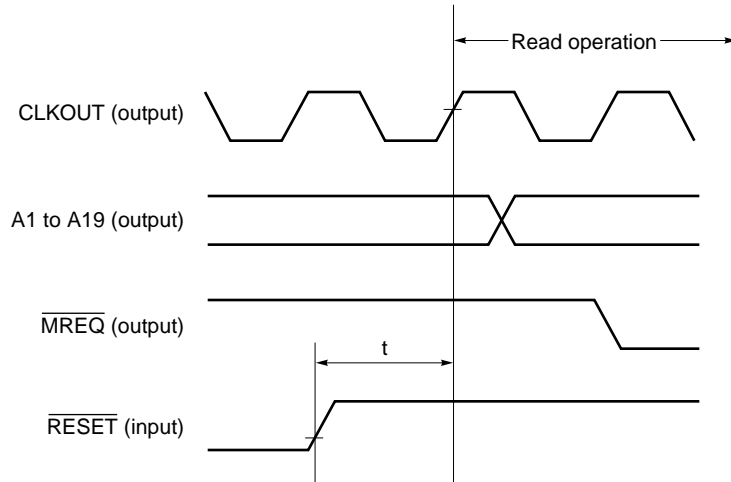
A.9.1

The V_{DD} consumption current when reset is the same as that in the normal operation mode.

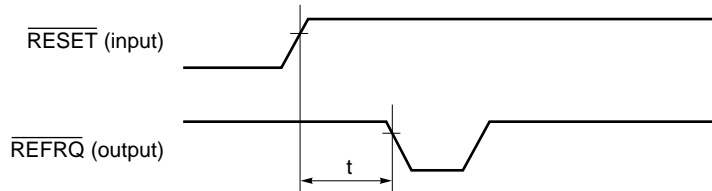
Q.9.2

In the V35 family, when the low-to-high transition of the $\overline{\text{RESET}}$ signal is made to release system reset

(1) How long does it take until the CPU operates?



(2) How long does it take until the $\overline{\text{REFRQ}}$ signal is output?



A.9.2

- (1) A branch is taken to the start address (FFFF0H) 21 clock cycles after the $\overline{\text{RESET}}$ signal goes high.
- (2) When the refresh mode register (RFM) is in the state in which the $\overline{\text{RESET}}$ signal is input consecutively, the first $\overline{\text{REFRQ}}$ signal is output $24/f_{\text{CLK}}$ (seconds) after the system reset is released.

★ **C.10 Other****Q.10.1**

Can a program distinguish between the V25 and V25+ and between the V35 and V35+?

A.10.1

They can be distinguished from each other by reading the contents of address $\times\times F6BH$ or $\times\times F7BH$ of the special function register area just after reset, as listed below.

	F6BH	F7BH
V25, V35	00H (SCE0)	00H (SCE1)
V25+, V35+	60H (SCS0)	60H (SCS1)

However, because for start bit (bit 7), the RxD0 and RxD1 pins' state is monitored intact, the actually read value depends on the state of the RxD0 and RxD1 pins.

For functional differences between the V25 and V25+ and between the V35 and V35+, see the V25+/V35+ user's manual.

Q.10.2

Why is the IC pin fixed high with an external pull-up resistor?

A.10.2

The IC pin is connected internally. It is recommended to connect a pull-up resistor for protection of the pin against external input such as a surge.

Q.10.3

What do EA and T of "EA + 6 + T" represent in the instruction execution time (number of clock cycles)?

A.10.3

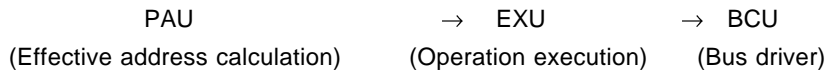
EA denotes the number of clock cycles which varies depending on the memory addressing mode. T denotes the number of wait states.

Q.10.4

For transfer instruction MOV, why are no wait states inserted with the number of clock cycles for on-chip RAM access disable being "EA + 2" for "MOV mem reg"?

A.10.4

The V25/V35 family performs three-stage pipeline processing as follows.



The number of "MOV mem reg" instruction execution clock cycles is only the number of clock cycles required for the PAU and EXU, and when operation processing terminates, data is sent to the BCU for execution of the next instruction.

APPENDIX D REGISTER INDEX (IN ALPHABETICAL ORDER)

Symbol	Name	Page(s)
BRG0,1	Baud rate generator register 0 and 1	78, 79, 233
DAR	Destination address register	181
DARH	Destination address register high	181
DIC0, 1	DMA interrupt request control registers 0 and 1	79, 180
DMAC0, 1	DMA control registers 0 and 1	79, 179
DMAM0, 1	DMA mode registers 0 and 1	79, 177
EMS0 to 2	External interrupt macro service control registers 0 to 2	78, 114
EXIC0 to 2	External interrupt request control registers 0 to 2	78, 114
FLAG	User flag register	71, 80
IDB	Internal data area base register	77, 80
INTM	External interrupt mode register	78, 113
ISPR	Interrupt priority register	78, 111
MD0, 1	Modulo/timer registers 0 and 1	79, 207
MSC	Macro service counter	105
MSP	Macro service pointer	105
MSS	Macro service segment	105
P0 to 2	Ports 0 to 2	78, 191
PC	Program counter	67
PM0 to 2	Port 0 to 2 mode registers	78, 191
PMC0 to 2	Port 0 to 2 mode control registers	78, 191
PMT	Port T mode register	78, 199
PRC	Processor control register	80, 206
PSW	Program status word	67
PT	Port T	78, 199
RFM	Refresh mode register	80, 161
RxB0, 1	Receive buffer registers 0 and 1	78, 79, 221
SAR	Source address register	181
SARH	Source address register high	181
SCC0, 1	Serial control registers 0 and 1	78, 79, 235
SCE0, 1	Serial error registers 0 and 1	78, 79, 236
SCHR	Search character	105
SCM0, 1	Serial mode registers 0 and 1	78, 79, 229
SEIC0, 1	Serial error interrupt request control registers 0 and 1	78, 79, 239
SFRP	Special function register pointer	105

APPENDIX D REGISTER INDEX (IN ALPHABETICAL ORDER)

Symbol	Name	Page(s)
SRIC0, 1	Serial reception interrupt request control registers 0 and 1	78, 79, 239
SRMS0, 1	Serial reception macro service control registers 0 and 1	78, 79, 240
STBC	Standby control register	80, 241
STIC0, 1	Serial transmission interrupt request control registers 0 and 1	78, 79, 239
STMS0, 1	Serial transmission macro service control registers 0 and 1	78, 79, 240
TBIC	Time base interrupt request control register	80, 219
TC	Terminal counter	181
TM0, 1	Timer registers 0 and 1	79, 207
TMC0, 1	Timer control registers 0 and 1	79, 209
TMIC0 to 2	Timer unit interrupt request control registers 0 to 2	79, 216
TMMS0 to 2	Timer unit macro service control registers 0 to 2	79, 216
TxB0, 1	Transmit buffer registers 0 and 1	78, 79, 221
WTC	Wait control register	80, 139

Facsimile Message

From:

Name

Company

Tel.

FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America

NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Hong Kong, Philippines, Oceania

NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Asian Nations except Philippines

NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Europe

NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

Korea

NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Japan

NEC Corporation
Semiconductor Solution Engineering Division
Technical Information Support Dept.
Fax: 044-548-7900

South America

NEC do Brasil S.A.
Fax: +55-11-6465-6829

Taiwan

NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Technical Accuracy	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>