

To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: <http://www.renesas.com>

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (<http://www.renesas.com>)

Send any inquiries to <http://www.renesas.com/inquiry>.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
 - “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.
 - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
 - “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

USER'S MANUAL

Phase-out/Discontinued

V40HL™ and V50HL™

16/8-BIT AND 16-BIT MICROPROCESSORS

HARDWARE

μPD70208H
μPD70216H

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.

Intertool™ is a trademark of Intermetrix Microsystems Software Corp.

V40HL™, V50HL™, V20HL™, V30HL™, V40™, V50™, and V series™ are trademarks of NEC Corporation.

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.

Major Revisions in this Edition

Page	Description
p. 24	3.2 (8) RESET Addition of description
p. 141	Table 5-17 Internal Clock and Maximum Baud Rate Addition of items
p. 178	5.9.7 Incomplete Interrupt Modification of Caution 2
p. 222	5.10.12 Cascade connection Addition of μ PD71037
p. 244	Table 6-1 Validity of Wait State in Each Bus Cycle Modification
p. 250, 251	6.3 Interrupt Function Addition of description
p. 252	Table 6-3 Interrupt Sources Addition of Note
p. 269	6.6.3 (2) HALT mode Addition of description

The mark * shows major revised points.

INTRODUCTION

Target users	This manual is intended for engineers who wish to understand the functions of the μ PD70208H and the μ PD70216H (also called the V40HL and V50HL, respectively) and use these products to design application systems.
Objectives	The objective of this manual is to provide the user with an understanding of the μ PD70208H and μ PD70216H hardware functions, which are indicated in the organization shown below.
Organization	The μ PD70208H and μ PD70216H user's manuals are divided into two versions, the hardware version (this manual) and the instruction version (16-Bit V Series™ User's Manual – Instruction).

Hardware Version

- General Description
- Pin Functions
- Architecture
- Internal Block Functions
- Internal/External Control Functions
- μ PD8080AF Emulation

Instruction Version

- General Description
- Description of Instructions
- Instruction Map
- Table of μ PD8086 and μ PD8088 Mnemonic Correspondence

How to Read this Manual Those who read this manual require a general understanding of electronics, logic circuits, and microprocessors. In this manual, descriptions of the part numbers μ PD70208H and μ PD70216H have been standardized to the V40HL and V50HL, respectively.

For users who already have experience with the μ PD70208 and μ PD70216 (also known as the V40™ and V50™, respectively), the V40HL and V50HL are high-speed, low-power versions of the V40 and V50. Chiefly read section 1.5, entitled "Differences Between the V40HL and V50HL and the V40 and V50," to ascertain the differences between these products.

To ascertain the details of a register when the register name is known, read **APPENDIX A, "REGISTER INDEX (IN ALPHABETICAL ORDER)."**

To ascertain the details of a function when the function name is known, read **APPENDIX B, "GENERAL INDEX."**

To comprehend the details of instruction functions, refer to the separate manual "16-Bit V Series User's Manual – Instruction."

To comprehend the electrical characteristics of the V40HL and V50HL, refer to the separate data sheets.

To gain a general understanding of the V40HL and V50HL, read this manual in the sequence indicated in the table of contents.

Conventions

In general, in this manual uses the following conventions:

- **Weighting of data notation** : The high-order data are written on the left, while the low-order data are written on the right.
- **Active low notation** : $\overline{\text{XXXX}}$ (over bar over pin and signal names).
- **Memory map addresses** : High-order addresses are on the top while low-order addresses are on the bottom.
- The term **"Note"** placed in the text indicates that an additional or supplementary information will be provided somewhere below.
- The term **"Caution"** indicates an explanation that the reader should pay particular attention to.
- The term **"Remark"** indicates that a supplemental explanation of the text is provided.
- **Numerical notation** : Binary ... xxxx or xxxxB
Decimal ... xxx
Hexadecimal ... xxxH
- **Easily confused characters** : 0 (zero) and the letter O.
1 (one), the upper-case letter I and the lower-case letter l.

Related Documents

Brochure: U10652J^{Note}

Data Sheet: IC-3659

User's Manual Hardware: This manual

16-bit V series User's Manual Instructions: U11301E

Application Note Hardware Design: U10037E

Application Note Software: U10911E

Questions and Answers: U11123E

Register Application Table: IEM-5575^{Note}

Note: This document number is that of Japanese-version document.

CONTENTS

CHAPTER 1 GENERAL DESCRIPTION	1
1.1 Features	2
1.2 Ordering Information	3
1.3 Pin Configurations	4
1.4 Internal Block Diagram	11
1.5 Differences Between the V40HL and V50HL and the V40 and V50	13
1.6 Functional Overview	14
CHAPTER 2 APPLICATIONS	17
CHAPTER 3 PIN FUNCTIONS	19
3.1 Pin Function List	19
3.2 Functional Description of Pins	22
3.3 Pin Input/Output Circuits and Recommended Handling of Unused Pins	31
CHAPTER 4 ARCHITECTURE	35
4.1 Memory and I/O Mapping	35
4.1.1 Memory map and access methods	35
4.1.2 I/O map and access methods	39
4.1.3 System I/O area	41
4.1.4 Memory I/O read and write timing	49
4.1.4 Memory I/O read and write timing	49
4.2 Logical Addresses and Physical Addresses	62
4.3 Address Generation	65
4.3.1 Instruction address	65
4.3.2 Data addressing	66
4.4 Speeding Up Instruction Execution	71
4.4.1 Dual data buses	71
4.4.2 Effective address generation circuit (EAG)	72
4.4.3 Temporary registers/shifters A/B (TA/TB)	72
4.4.4 Loop counter (LC)	73
4.4.5 Program counter (PC) and prefetch pointer (PFP)	73
CHAPTER 5 INTERNAL BLOCK FUNCTIONS	75
5.1 CPU	75
5.1.1 Features	75
5.1.2 Internal organization	76
5.2 Clock Generator (CG)	87
5.2.1 Features	87
5.2.2 Differences with the V40 and V50	87

5.2.3	Internal block	88
5.3	Bus Interface Unit (BIU)	89
5.4	Bus Arbitration Unit (BAU)	90
5.4.1	Priority	90
5.4.2	Bus wait operation	91
5.4.3	Timing of bus priority switching	92
5.4.4	DMA request and hold request acceptance waiting times	93
5.5	Wait Control Unit (WCU)	96
5.5.1	Features	96
5.5.2	Differences with the V40 and V50	96
5.6	Refresh Control Unit (REFU)	97
5.6.1	Features	97
5.6.2	Differences with the V40 and V50	97
5.7	Timer/Counter Unit (TCU)	98
5.7.1	Features	98
5.7.2	Internal block diagram	98
5.7.3	Addressing	99
5.7.4	Operating procedures	100
5.7.5	Registers and commands	101
5.7.6	Count mode	111
5.7.7	Precautions	124
5.8	Serial Control Unit (SCU)	125
5.8.1	Features	125
5.8.2	Differences with the V40 and V50	125
5.8.3	Internal block diagram	126
5.8.4	Addressing	127
5.8.5	Initialization	127
5.8.6	Serial data format	128
5.8.7	Operating procedures	130
5.8.8	Registers and commands	131
5.8.9	Transmission and reception baud rates	139
5.8.10	Serial transmission and reception	142
5.8.11	Precautions	146
5.9	Interrupt Control Unit (ICU)	148
5.9.1	Features	148
5.9.2	Internal block diagram	148
5.9.3	Addressing	149
5.9.4	Initialization	151
5.9.5	Registers and commands	152
5.9.6	Interrupt sequence	169
5.9.7	Incomplete interrupt	178
5.10	DMA Controller Unit (DMAU)	179
5.10.1	Features	179
5.10.2	Differences with the V40 and V50	179
5.10.3	Internal block diagram	180
5.10.4	Differences between μ PD71071 mode and μ PD71071	181
5.10.5	Differences between μ PD71037 mode and μ PD71037	181

5.10.6 Differences between the μ PD71071 and μ PD71037 modes	181
5.10.7 Basic operation of DMAU	183
5.10.8 μ PD71071 mode	188
5.10.9 μ PD71037 mode	204
5.10.10 Transfer mode	218
5.10.11 Auto initialization	220
5.10.12 Cascade connection	222
5.10.13 Status transition diagram	225
5.10.14 Precautions	229
CHAPTER 6 INTERNAL/EXTERNAL CONTROL FUNCTIONS	231
6.1 Wait Function	231
6.1.1 Wait function by using READY pin	231
6.1.2 Wait function by using WCU	233
6.1.3 Relationship between WCU and READY pin	243
6.2 Refresh Function	245
6.2.1 Refresh address	245
6.2.2 Refresh control register (RFC)	245
6.2.3 REFU bus control	247
6.2.4 Refresh timing	247
6.3 Interrupt Function	250
6.3.1 Operation of ICU Interrupt	255
6.3.2 BRK flag (single-step) interrupt	262
6.3.3 Timing when interrupts not acknowledged	262
6.3.4 Interrupt processing during execution of block processing instruction	263
6.4 Bus Hold Function	264
6.5 Variable Instruction Cycle Time Function	266
6.6 Standby Function	267
6.6.1 Features	267
6.6.2 Standby control register (SBCR)	268
6.6.3 HALT mode	269
6.6.4 STOP mode	272
6.7 Reset Function	275
6.7.1 CPU reset operation	275
6.7.2 Reset operations of system I/O area and internal peripheral units	275
6.7.3 Output pin status at reset	279
CHAPTER 7 μPD8080AF EMULATION	281
7.1 Changing from Native Mode to Emulation Mode	281
7.1.1 BRKEM imm8 instruction	281
7.1.2 RETI instruction	282
7.2 Changing Mode from Emulation Mode to Native Mode	283
7.2.1 RESET input	283
7.2.2 NMI and ICU request	283
7.2.3 CALLN imm8 instruction	283

7.2.4	RETEM instruction	284
7.2.5	Nesting of emulation	285
7.3	Emulation Mode	285
APPENDIX A REGISTER INDEX (IN ALPHABETICAL ORDER).....		289
APPENDIX B GENERAL INDEX		293

LIST OF FIGURES (1/5)

Figure No.	Title	Page
2-1	Handy Word-Processor Block Diagram	17
2-2	Facsimile Machine Block Diagram	18
3-1	Pin Input and Output Circuits	33
4-1	Memory Map	35
4-2	V40HL Interfacing Between CPU and Memory	36
4-3	V50HL Interfacing Between CPU and Memory	37
4-4	I/O Map	39
4-5	List of Registers in the System I/O Area	41
4-6	On-Chip Peripheral Connection Register (OPCN)	42
4-7	On-Chip Peripheral Selection Register (OPSEL)	43
4-8	System Control Register (SCTL)	44
4-9	Internal Peripheral Unit Mapping Example	46
4-10	On-Chip Peripheral High Address Register	47
4-11	TCU Low Address Register (TULA)	47
4-12	SCU Low Address Register (SULA)	47
4-13	ICU Low Address Register (IULA)	48
4-14	DMAU Low Address Register (DULA)	48
4-15	Memory and External I/O Read Timing (V40HL)	50
4-16	Memory and External I/O Write Timing (V40HL)	52
4-17	Internal I/O Read Timing (V40HL)	54
4-18	Internal I/O Write Timing (V40HL)	55
4-19	Memory and External I/O Read Timing (V50HL)	56
4-20	Memory and External I/O Write Timing (V50HL)	58
4-21	Internal I/O Read Timing (V50HL)	60
4-22	Internal I/O Write Timing (V50HL)	61
5-1	CPU Internal Block Diagram	76
5-2	PSW Bit Fields	81
5-3	Effective Address Generation	85
5-4	Synchronizing the RESET and READY Signals	89
5-5	Internal Bus Cycles	91
5-6	Bus Waiting Operation	91

LIST OF FIGURES (2/5)

Figure No.	Title	Page
5-7	Timing of Bus Priority Switching	92
5-8	Timing of Bus Master Switch Transition from Non-DMAU	93
5-9	Timing of Bus Master Switch Transition from DMAU	93
5-10	Minimum Wait Time	94
5-11	Maximum Wait Time	95
5-12	TCU Basic Operational Procedures	100
5-13	TMD (For Mode Word)	102
5-14	Timer Clock Selection Register (TCKS)	104
5-15	Supply a Clock to the TCU	105
5-16	TMD (When there is a count latch command)	107
5-17	TMD (When there is a multiple latch command)	108
5-18	TSTn	109
5-19	Example of NC Flag Changes	109
5-20	Multiple Latch Command Execution Example	110
5-21	Mode 0 Operation Example	113
5-22	Mode 1 Operation Example	115
5-23	Mode 2 Operation Example	117
5-24	Mode 3 Operation Example	119
5-25	Mode 4 Operation Example	121
5-26	Mode 5 Operation Example	123
5-27	Serial Data Format	128
5-28	SCU Operating Procedures	130
5-29	Serial Mode Register (SMD)	132
5-30	Serial Command Register (SCM)	134
5-31	$\overline{\text{SRDY}}$ Signal	135
5-32	Serial Status Register (SST)	136
5-33	Break Input and Detection	137
5-34	Serial Interrupt Masking Register (SIMK)	138
5-35	Relationship Between the TM Bit, RM bit and INTL1	138
5-36	Baud Rate Generator (BRC)	139
5-37	Baud Rate Clock Generation Diagram	140
5-38	Serial Transmit Operation	143
5-39	Serial Receive Operation	145
5-40	Initialization Sequence	151

LIST OF FIGURES (3/5)

Figure No.	Title	Page
5-41	Interrupt Initialization Word 1 Register (IIW1)	152
5-42	Interrupt Request Input Circuit	153
5-43	Interrupt Initialization Word 2 Register (IIW2)	155
5-44	Interrupt Vector Number Generation	155
5-45	Interrupt Initialization Word 3 Register (IIW3)	156
5-46	Interrupt Initialization Word 4 Register (IIW4)	157
5-47	Normal Nesting Mode	158
5-48	Interrupt Mask Word Register (IMKW)	160
5-49	Interrupt Priority, Finish Word Register (IPFW)	161
5-50	INTL Request Priorities (Rotating Priority)	162
5-51	Interrupt Mode Word Register (IMDW)	165
5-52	Examples of the Exception Nesting Mode	166
5-53	Interrupt Polling Register (IPOL)	167
5-54	Interrupt Request Register (IRQ)	167
5-55	Interrupt In-Service Register (IIS)	168
5-56	Interrupt Sequence	170
5-57	Interrupt Acknowledge Timing Example (single mode)	171
5-58	Example of Connecting Slave in Expansion Mode	174
5-59	Interrupt Sequence in Expansion Mode	175
5-60	Interrupt Acknowledge Timing Example (Expansion Mode)	176
5-61	DMAU Operation Flow	183
5-62	DMA Timing	185
5-63	Occurrence of Terminal Count (TC)	187
5-64	DMA Initialize Command Register (DICM)	191
5-65	DMA Channel Register Read Command	192
5-66	DMA Channel Register Write Command	193
5-67	DMA Count Register Read/Write Command	194
5-68	DMA Address Register Read/Write Command	195
5-69	DMA Device Control Register Read/Write Command	196
5-70	Expanded Write Timing	197
5-71	DMA Priority	197
5-72	Difference in Bus Control Depending on Bus Mode	198
5-73	DMA Mode Control Register Read/Write Command	199
5-74	DMA Status Register Read Command	202

LIST OF FIGURES (4/5)

Figure No.	Title	Page
5-75	DMA Mask Register Read/Write Command	203
5-76	Read Status Register (DRST)	208
5-77	Write Command Register (DWC)	209
5-78	Expanded Write Timing	209
5-79	Write Request Register (DWRQ)	211
5-80	Write Single Mask Register (DWSM)	212
5-81	Write All Mask Register (DWAM)	213
5-82	Write Mode Register (DWM)	214
5-83	Bank Address Register (BADR)	216
5-84	Bank Select Register (BSEL)	217
5-85	Bank Registers (BNKR0 to BNKR3)	217
5-86	Auto Initialization Application Example 1	220
5-87	Auto Initialization Application Example 2	221
5-88	Example of Cascade Connection	222
5-89	Example of Cascade Timing	223
5-90	Idle Cycle	225
5-91	DMA Cycle	225
5-92	Terminal Count Generation	229
6-1	Ready Timing (no-wait status)	231
6-2	Ready Timing (with 1 wait state inserted)	232
6-3	Programmable Wait Memory Area Setting Register (WMB)	234
6-4	Dividing Memory Space into Three Blocks	235
6-5	Extended Wait Block Select Register (EXWB)	236
6-6	Dividing Memory Space into Five Blocks	237
6-7	Dividing I/O Space into Three Blocks	237
6-8	Wait Submemory Block Setting Register (WSMB)	238
6-9	Wait I/O Block Setting Register (WIOB)	239
6-10	Programmable Wait Cycle Setting Register 1 (WCY1)	240
6-11	Programmable Wait Cycle Setting Register 3 (WCY3)	241
6-12	Programmable Wait Cycle Setting Register 2 (WCY2)	242
6-13	Relationship between WCU and READY Pin	243
6-14	Example of Bus Timing When 4 Clocks of TWs Are Inserted	243

LIST OF FIGURES (5/5)

Figure No.	Title	Page
6-15	Refresh Control Register (RFC)	245
6-16	Invalid Refresh Request	247
6-17	Refresh Timing	248
6-18	ICU Inputs	250
6-19	Interrupt Vector Table	253
6-20	Example of Interrupt Acknowledge Timing (single mode)	258
6-21	Example of Interrupt Acknowledge Timing (expanded mode)	260
6-22	Bus Hold Timing	264
6-23	Standby Control Register (SBCR)	268
7-1	Changing Mode by BRKEM Instruction	281
7-2	Changing Mode by RETI Instruction	282
7-3	Changing Mode by NMI Input, ICU Interrupt Request, and CALLN Instruction	283
7-4	Changing Mode by RETEM Instruction	284
7-5	Trouble due to Nesting of Emulation	285
7-6	Correspondence of Register Set	286
7-7	Correspondence of PSW and FLAG	286
7-8	CPU Mode Transition	287

[MEMO]

LIST OF TABLES (1/2)

Table No.	Title	Page
1-1	Differences Between the V40HL and V50HL and the V40 and V50	13
3-1	Pin Functions	19
3-2	Input/Output Circuits for Each Pin and Recommended Handling When Not Used	31
4-1	Memory Element Addresses and Data Organization	36
4-2	V50HL Data Access	38
5-1	Offset and Segment Register Combinations	78
5-2	Bus Masters	90
5-3	TCU Internal Registers and Command Addresses	99
5-4	TCU Registers and Commands Addresses	101
5-5	Writing to the Count Register	106
5-6	Reading the Counter	106
5-7	NC Flag Changes	109
5-8	Mode 0 Operation	112
5-9	Mode 1 Operation	114
5-10	Mode 2 Operation	116
5-11	Mode 3 Operation	118
5-12	Mode 4 Operation	120
5-13	Mode 5 Operation	122
5-14	SCU Internal Register and Command Addresses	127
5-15	SCU Register and Command Addresses	131
5-16	Baud Rate Settings	140
5-17	Internal Clock and Maximum Baud Rate	141
5-18	ICU Internal Register and Command Addresses	149
5-19	ICU Register and Command Addresses	150
5-20	General Description of the ICU Registers	152
5-21	Differences between μ PD71071 Mode and μ PD71071	181
5-22	Differences between μ PD71037 Mode and μ PD71037	181
5-23	Registers of DMAU (μ PD71071 mode)	188
5-24	Addresses of DMAU Internal Registers (μ PD71071 mode)	189
5-25	DMAU Command Address (μ PD71071 mode)	190
5-26	Initializing Registers by Reset	191

LIST OF TABLES (2/2)

Table No.	Title	Page
5-27	Updating Current Registers	201
5-28	A0 and \overline{UBE} Signals during DMA Operation	201
5-29	Addresses of DMAU's Internal Registers (μ PD71037 mode)	205
5-30	Commands in μ PD71037 Mode	207
5-31	Transfer Modes and DMA Service End Conditions	218
5-32	DMA Operation for Each Combination of Transfer Mode and Bus Mode	219
5-33	Pin Status During Cascade Connection	224
6-1	Validity of Wait State in Each Bus Cycle	244
6-2	REFU Bus Control	247
6-3	Interrupt Sources	252
6-4	Pin Status in Hold Status	265
6-5	Relationship between Division Ratio and Internal Clock	266
6-6	Pin Status in HALT Mode (when peripheral functions do not operate)	270
6-7	Pin Status in STOP Mode (when peripheral functions do not operate)	273
6-8	Resetting CPU	275
6-9	Resetting System I/O Area	276
6-10	Resetting Internal Peripheral Unit	278
6-11	Pin Status at Reset	279

CHAPTER 1 GENERAL DESCRIPTION

The V40HL is a 16/8-bit microprocessor that has a 16-bit architecture, an 8-bit data bus, and internal general-purpose peripheral functions.

The V50HL is a 16-bit microprocessor that has a 16-bit architecture, a 16-bit data bus, and internal general-purpose peripheral functions.

These products are the high-speed, low-power-consumption versions of the V40 and V50 microprocessors, and incorporate the following additional functions and enhanced functions:

- 20-MHz maximum operating frequency
- Available 3-V power supply voltage
- STOP mode function
- Available system clock input stop
- A variable instruction cycle function
- Addition of the μ PD71037 mode to the DMA controller unit (DMAU)

The V40HL and V50HL are software-compatible with the V40 and V50, allowing the use of current programs without modifications. The V40HL and V50HL are also pin-compatible with the V40 and V50, which allows them to replace the V40 and V50.

The V40HL and V50HL are optimal for the control and data processing of a variety of compact, high-speed, low-voltage-operation and low-power-consumption products, and have applications in the following areas:

- Personal computers
- Dedicated word processors
- Handy terminals
- Printers
- Facsimile machines
- Robots and numerical machine control
- Communications control

1.1 Features

The V40HL and V50HL feature the following:

- High-speed and low-power-consumption versions of the V40 and V50
- Operating supply voltage of 3 V or 5 V
- Maximum operating frequencies of 10, 12.5, 16, and 20 MHz (when externally-supplied frequency is 20, 25, 32, and 40 MHz)
- Use of high-performance 16/8-bit and 16-bit CPUs (equivalent to the V20HL™ and V30HL™)
 - Minimum instruction execution times : 100 ns (at 20-MHz, 5-V operation)
200 ns (at 10-MHz, 3-V operation)
 - Multiply/divide instruction execution times : 0.95 to 2.8 μ s (at 20-MHz, 5-V operation)
1.9 to 5.6 μ s (at 10-MHz, 3-V operation)
- Standby functions:
 - HALT mode: I_{DD} (max.) = 2.2 mA/MHz (at 5-V operation) and 1.5 mA/MHz (at 3-V operation)
 - STOP mode: I_{DD} (max.) = 50 μ A (at 5-V operation) and 30 μ A (at 3-V operation)
- External system clock input stop
 - When stopping external clock input: I_{DD} (max.) = 50 μ A (at 5-V operation) and 30 μ A (at 3-V operation)
 - External clock frequency can be varied from the DC level to the maximum operating frequency.
- Variable instruction cycle function
- General-purpose peripheral functions:
 - Clock generator (CG)
 - Bus interface unit (BIU)
 - Bus arbitration unit (BAU)
 - Wait control unit (WCU)
 - Refresh control unit (REFU)
 - Timer/counter unit (TCU)
 - Serial control unit (SCU)
 - Interrupt control unit (ICU)
 - DMA control unit (DMAU)

1.2 Ordering Information

(1) V40HL

Part Number	Package Type	Maximum operating frequency (MHz)
μ PD70208HGF-10-3B9	80-pin plastic QFP (14 x 20 mm) (2.7-mm resin thickness)	10
μ PD70208HGF-12-3B9	80-pin plastic QFP (14 x 20 mm) (2.7-mm resin thickness)	12.5
μ PD70208HGF-16-3B9	80-pin plastic QFP (14 x 20 mm) (2.7-mm resin thickness)	16
μ PD70208HGF-20-3B9	80-pin plastic QFP (14 x 20 mm) (2.7-mm resin thickness)	20
μ PD70208HGK-10-9EU	80-pin plastic TQFP (fine pitch) (12 x 12 mm) (1-mm resin thickness)	10
μ PD70208HGK-12-9EU	80-pin plastic TQFP (fine pitch) (12 x 12 mm) (1-mm resin thickness)	12.5
μ PD70208HGK-16-9EU	80-pin plastic TQFP (fine pitch) (12 x 12 mm) (1-mm resin thickness)	16
μ PD70208HGK-20-9EU	80-pin plastic TQFP (fine pitch) (12 x 12 mm) (1-mm resin thickness)	20
μ PD70208HLP-10	68-pin plastic QFJ (950 x 950 mils)	10
μ PD70208HLP-12	68-pin plastic QFJ (950 x 950 mils)	12.5
μ PD70208HLP-16	68-pin plastic QFJ (950 x 950 mils)	16
μ PD70208HLP-20	68-pin plastic QFJ (950 x 950 mils)	20

Remark Plastic QFJ is the new name for the PLCC.

(2) V50HL

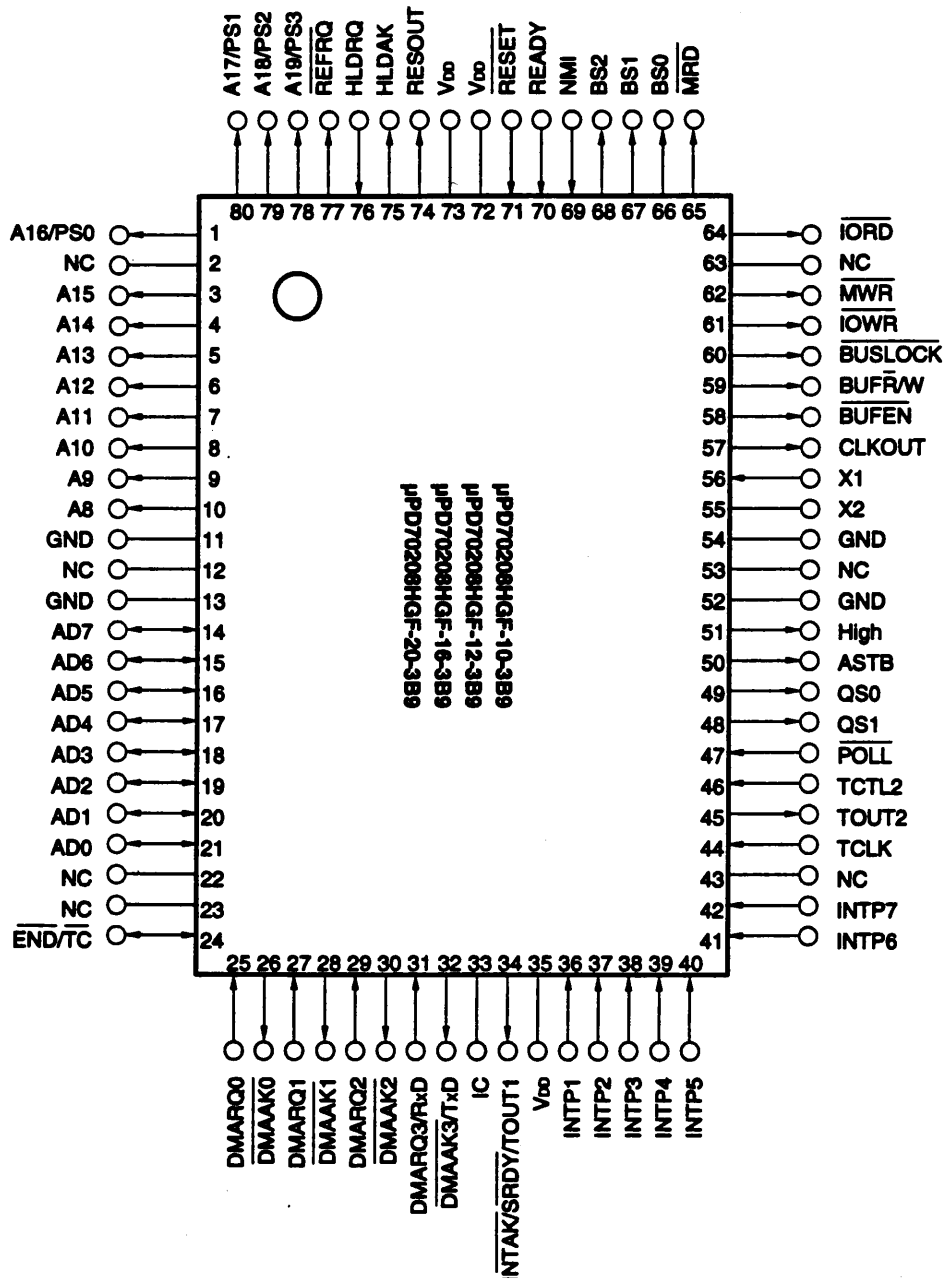
Part Number	Package Type	Maximum operating frequency (MHz)
μ PD70216HGF-10-3B9	80-pin plastic QFP (14 x 20 mm) (2.7-mm resin thickness)	10
μ PD70216HGF-12-3B9	80-pin plastic QFP (14 x 20 mm) (2.7-mm resin thickness)	12.5
μ PD70216HGF-16-3B9	80-pin plastic QFP (14 x 20 mm) (2.7-mm resin thickness)	16
μ PD70216HGF-20-3B9	80-pin plastic QFP (14 x 20 mm) (2.7-mm resin thickness)	20
μ PD70216HGK-10-9EU	80-pin plastic TQFP (fine pitch) (12 x 12 mm) (1-mm resin thickness)	10
μ PD70216HGK-12-9EU	80-pin plastic TQFP (fine pitch) (12 x 12 mm) (1-mm resin thickness)	12.5
μ PD70216HGK-16-9EU	80-pin plastic TQFP (fine pitch) (12 x 12 mm) (1-mm resin thickness)	16
μ PD70216HGK-20-9EU	80-pin plastic TQFP (fine pitch) (12 x 12 mm) (1-mm resin thickness)	20
μ PD70216HLP-10	68-pin plastic QFJ (950 x 950 mils)	10
μ PD70216HLP-12	68-pin plastic QFJ (950 x 950 mils)	12.5
μ PD70216HLP-16	68-pin plastic QFJ (950 x 950 mils)	16
μ PD70216HLP-20	68-pin plastic QFJ (950 x 950 mils)	20

Remark Plastic QFJ is the new name for the PLCC.

1.3 Pin Configurations

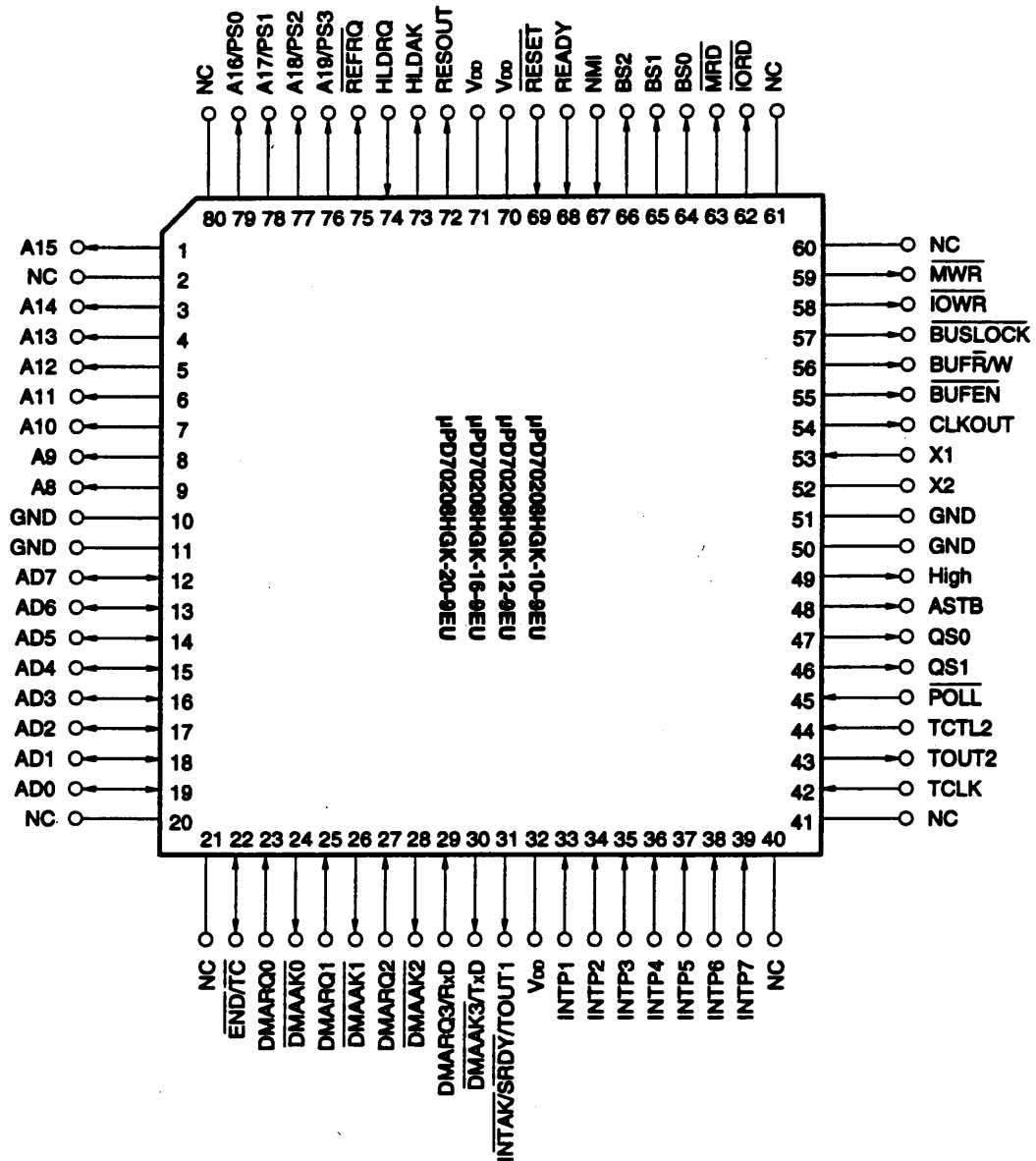
(1) V40HL

(a) 80-pin plastic QFP (Top View)

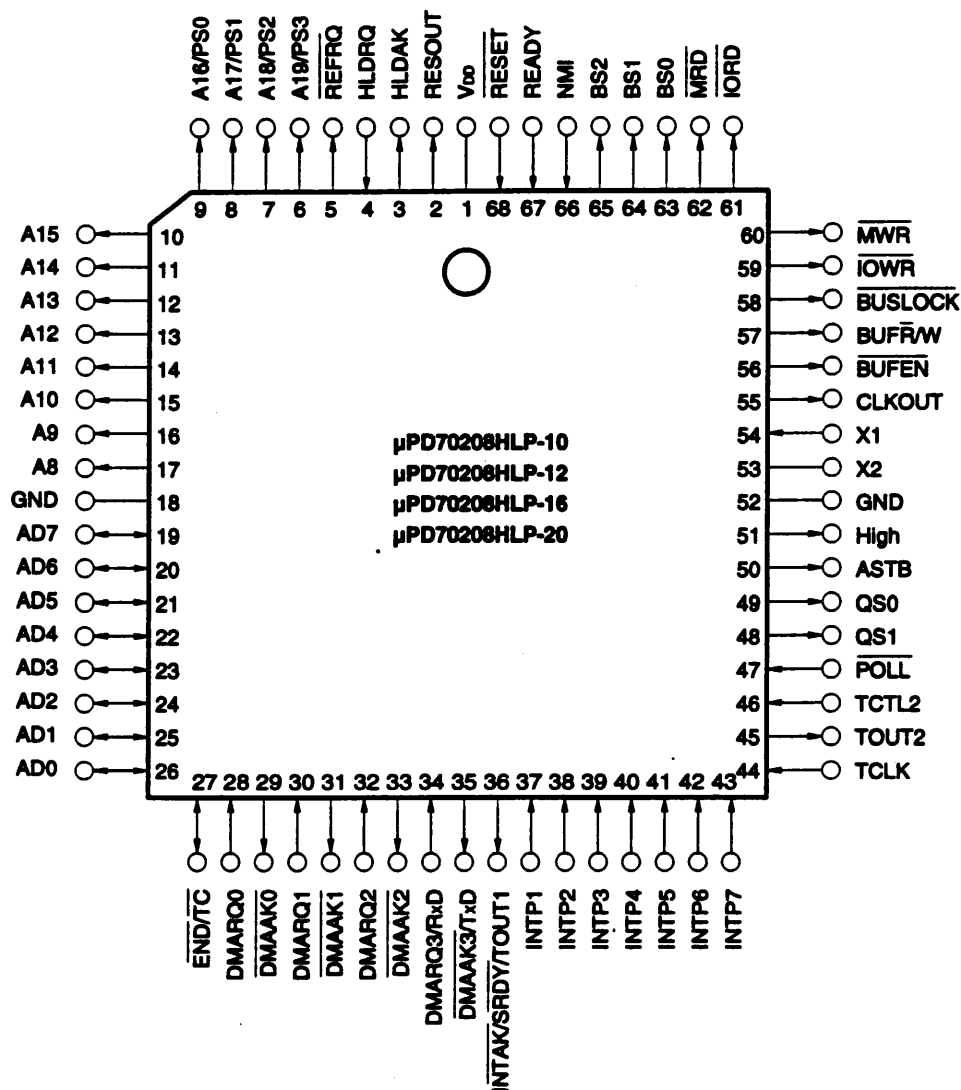


Caution Connect nothing to the IC pin.

(b) 80-pin plastic TQFP (Top View)

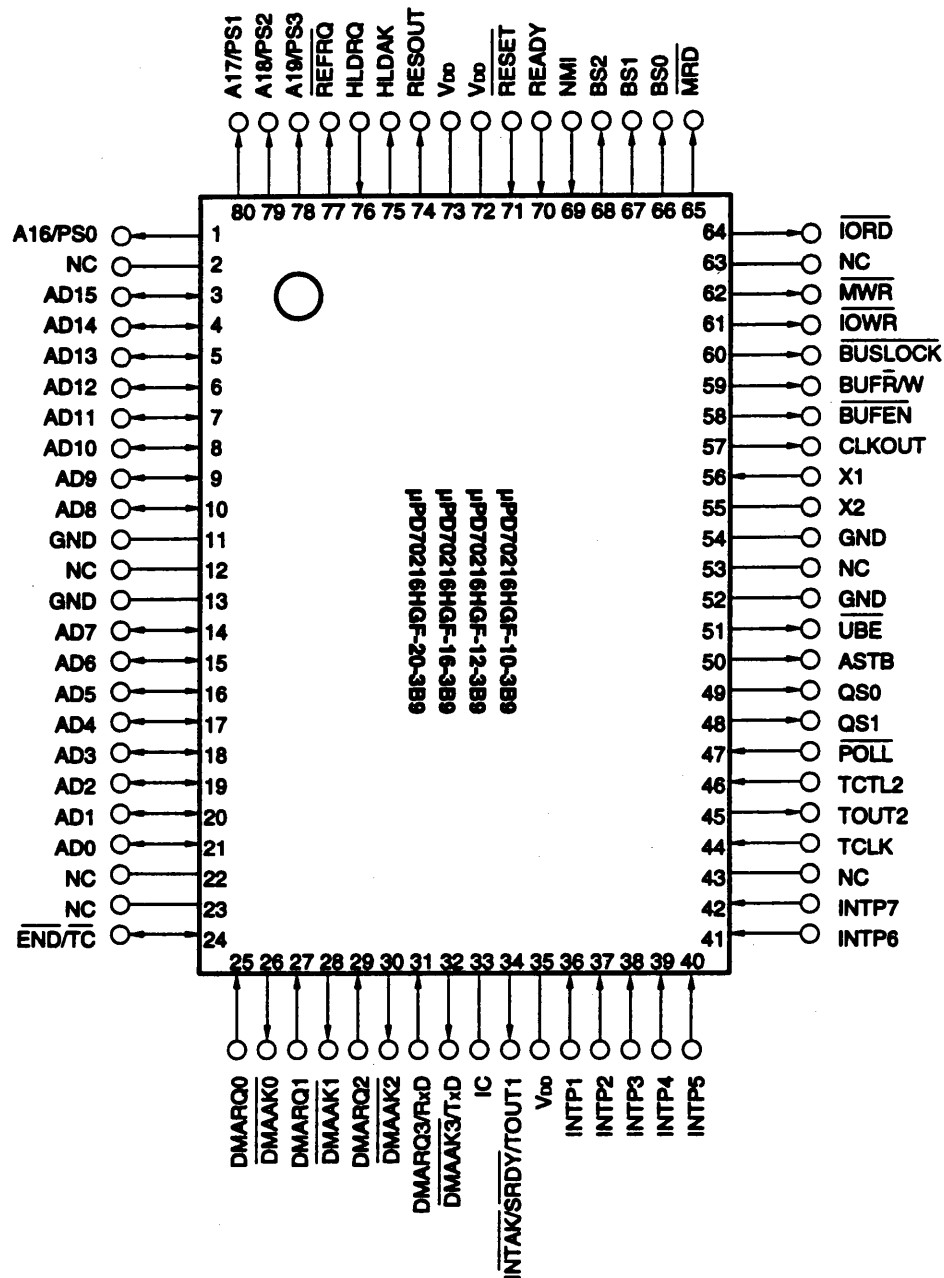


(c) 68-pin plastic QFJ (Top View)



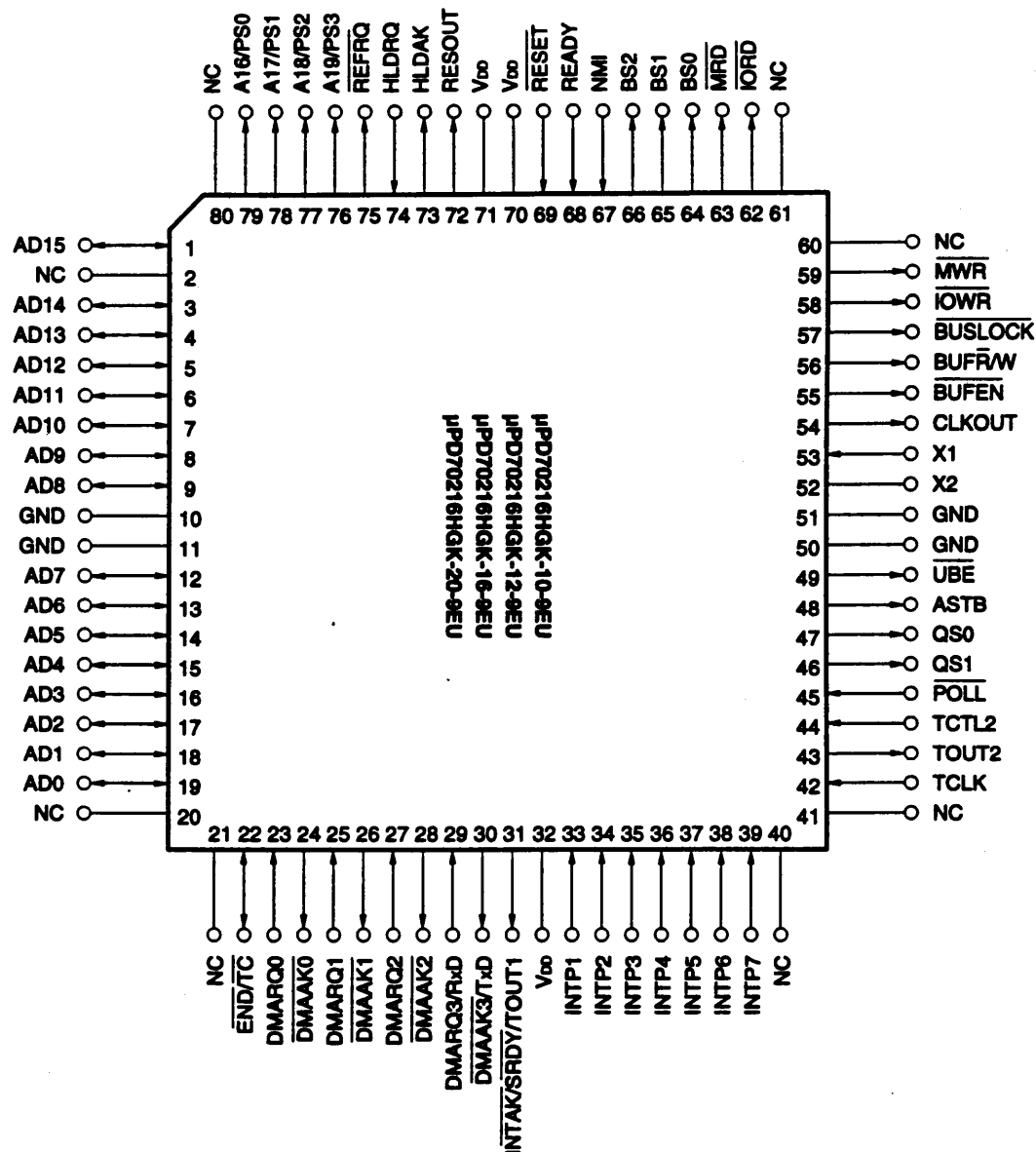
(2) V50HL

(a) 80-pin plastic QFP (Top View)

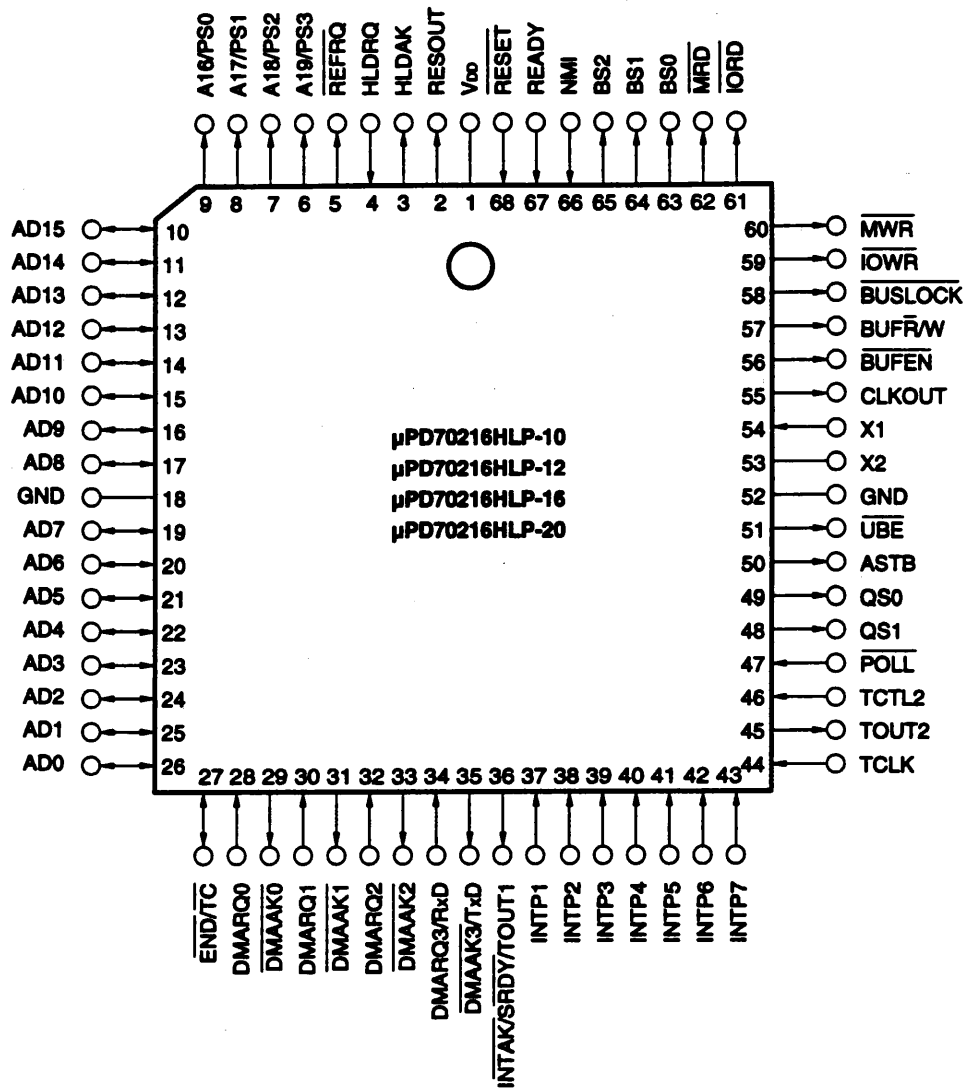


Caution Connect nothing to the IC pin.

(b) 80-pin plastic TQFP (Top View)



(c) 68-pin plastic QFJ (Top View)

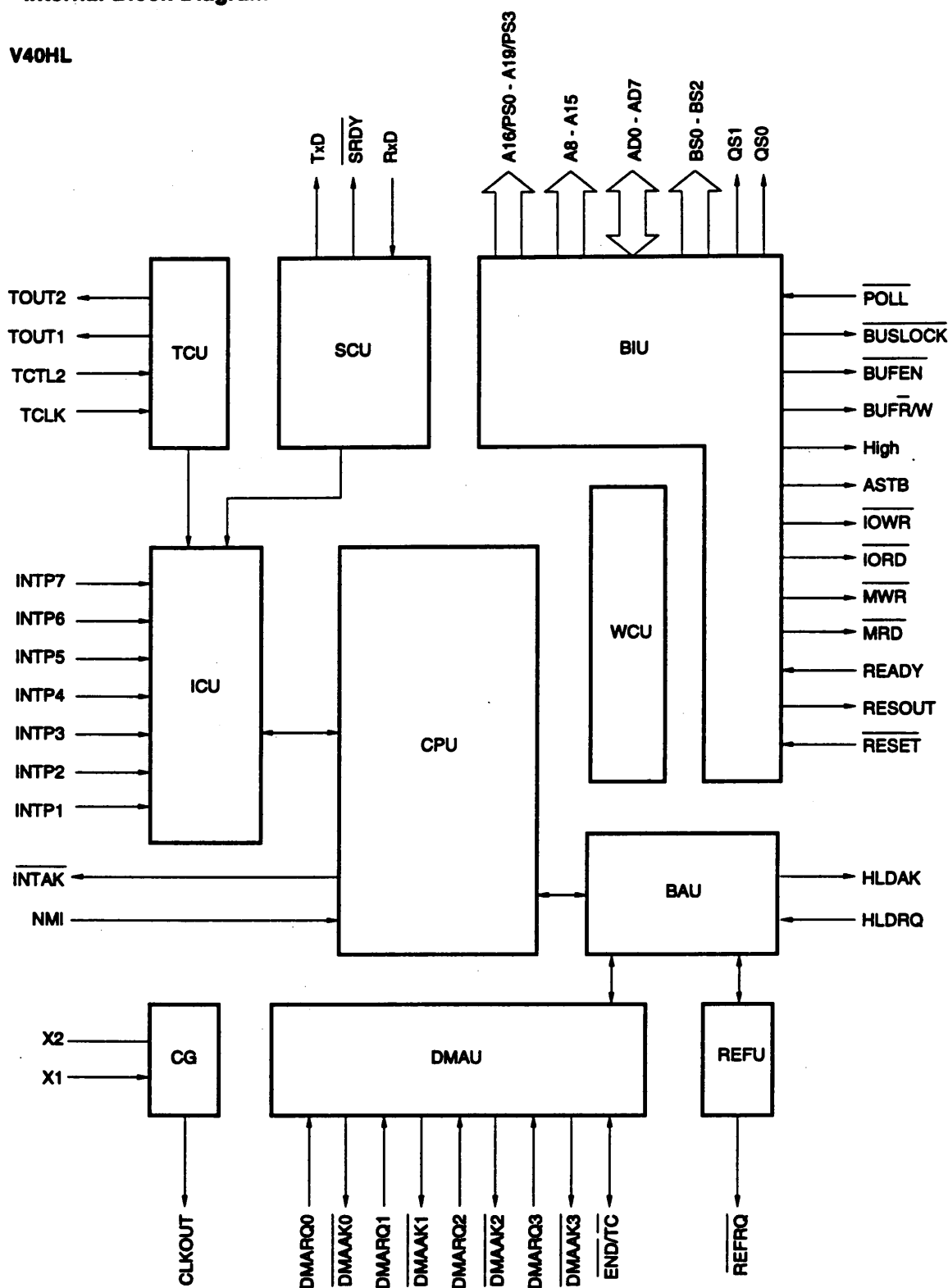


Pin Identification

AD0 to AD15:	Address Bus/Data Bus
A8 to A15:	Address Bus
A16/PS0 to A19/PS3:	Address/Processor Status
REFRQ:	Refresh Request
HLDRQ:	Hold Request
HLDAK:	Hold Acknowledge
RESET:	Reset
RESOUT:	Reset Output
READY:	Ready
NMI:	Non-Maskable Interrupt Request
MRD:	Memory Read
MWR:	Memory Write
IOR:	I/O Read
IOW:	I/O Write
ASTB:	Address Strobe
UBE:	Upper Byte Enable
High:	High Level Output
BUSLOCK:	Bus Lock
POLL:	Poll
BUFR/W:	Buffer Read/Write
BUFEN:	Buffer Enable
X1, X2:	Crystal
CLKOUT:	Clock Output
BS0 to BS2:	Bus Status
QS0, QS1:	Queue Status
TOUT2:	Timer Output 2
TCTL2:	Timer Control 2
TCLK:	Timer Clock
INTP1 to INTP7:	Interrupt Request from Peripherals
INTAK/SRDY/TOUT1:	Interrupt Acknowledge/Serial Ready/Timer Output 1
DMAAK3/TxD:	DMA Acknowledge/Transmit Data
DMARQ3/RxD:	DMA Request/Receive Data
DMAAK0 to DMAAK2:	DMA Acknowledge
DMARQ0 to DMARQ2:	DMA Request
END/TC:	End/Terminal Count
V _{DD} :	Power Supply
GND:	Ground
IC:	Internally Connected

1.4 Internal Block Diagram

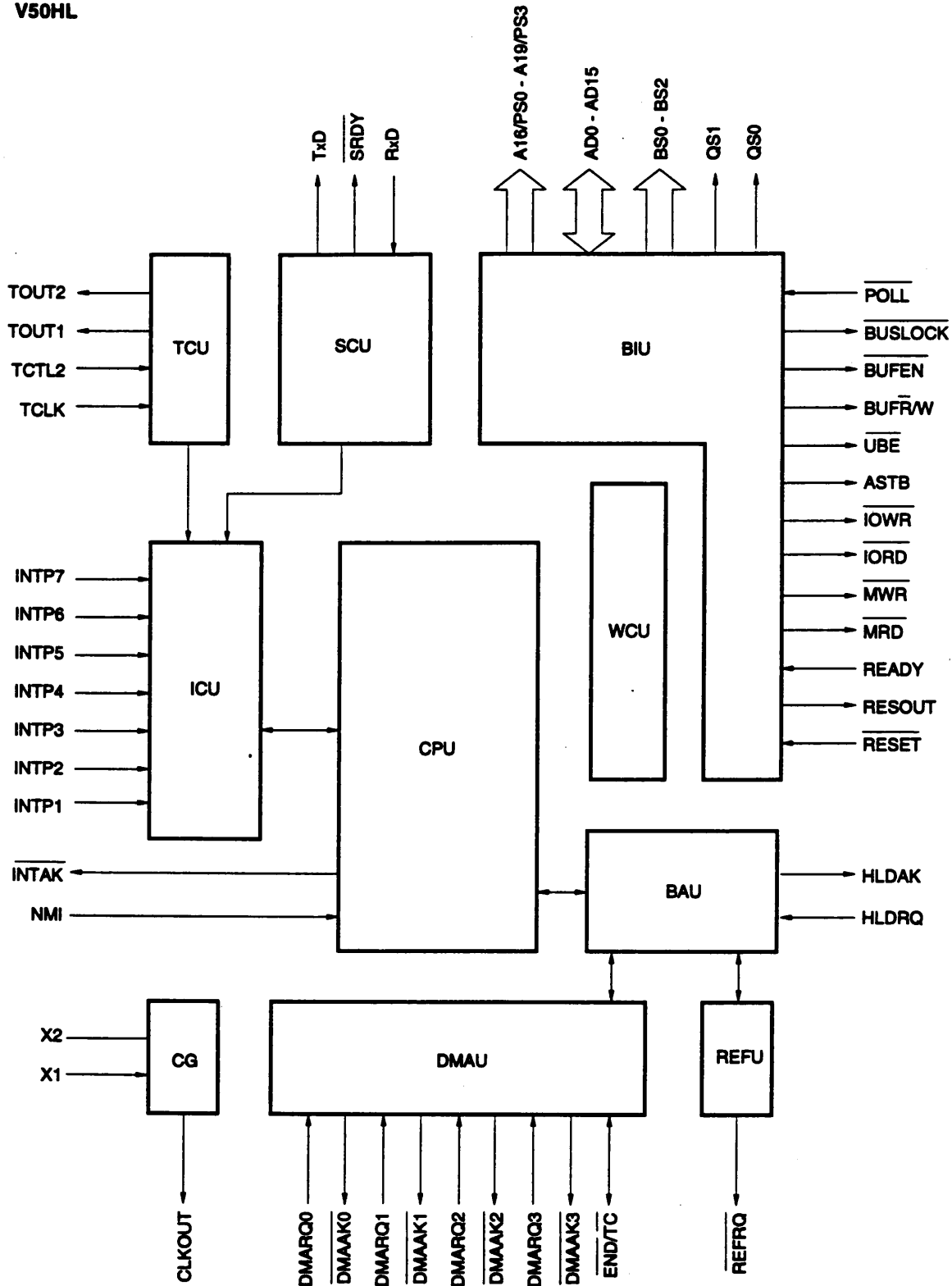
(1) V40HL



CPU : Central processing unit
 CG : Clock generator
 BIU : Bus interface unit
 BAU : Bus arbitration unit
 WCU : Wait control unit

REFU : Refresh control unit
 TCU : Timer/counter unit
 SCU : Serial control unit
 ICU : Interrupt control unit
 DMAU : DMA control unit

(2) V50HL



CPU : Central processing unit
CG : Clock generator
BIU : Bus interface unit
BAU : Bus arbitration unit
WCU : Wait control unit

REFU : Refresh control unit
TCU : Timer/counter unit
SCU : Serial control unit
ICU : Interrupt control unit
DMAU : DMA control unit

1.5 Differences Between the V40HL and V50HL and the V40 and V50

The major differences between the V40HL and V50HL and the V40 and V50 are shown below in Table 1-1.

Table 1-1. Differences Between the V40HL and V50HL and the V40 and V50

Item		V40HL, V50HL	V40, V50
Operating supply voltages		3 V, 5 V	5 V
Operating frequencies	V _{DD} = 5 V	MAX.: 10, 12.5, 16, 20 MHz MIN. : DC	MAX.: 8, 10 MHz MIN. : 2 MHz
	V _{DD} = 3 V	MAX.: 5, 6.25, 8, 10 MHz MIN. : DC	No operation
Clock generators (CG)		Variable frequency division ratio	Fixed frequency division ratio
		Variable instruction cycle times	Fixed instruction cycle time
		40-MHz maximum input frequency	20-MHz maximum input frequency
Internal I/O remapping function		I/O boundary can be remapped to an 8-bit or 16-bit boundary.	The V40 can be remapped to an 8-bit boundary. The V50 can be remapped to a 16-bit boundary.
Wait control unit (WCU)		Memory area: 5 segments ^{Note 1}	Memory area: 3 segments
		I/O area: 3 segments ^{Note 2}	I/O area: No segments
Refresh control unit (REFU)		16-bit refresh address	9-bit refresh address
		Supports $\overline{\text{REFRQ}}$ expansion timing.	No $\overline{\text{REFRQ}}$ expansion timing
Serial control unit (SCU)		On-chip dedicated baud rate generator	No on-chip dedicated baud rate generator
DMA control unit (DMAU)		$\mu\text{PD71071}$ or $\mu\text{PD71037}$ subset (either function is available)	$\mu\text{PD71071}$ subset
Standby functions		HALT and STOP modes	HALT mode only

Notes 1. Divides into three partitions after reset.

2. No division into partitions.

1.6 Functional Overview

Part Number		V40HL	V50HL
Item			
CPU		V20HL equivalent	V30HL equivalent
Internal data bus width		16 bits	
External data bus width		8 bits	16 bits
Operating ambient temperature		-40 to +85 °C	
Operating supply voltages		$V_{DD} = 5\text{ V} \pm 10\%$ $V_{DD} = 3\text{ V} \pm 10\%$	
Maximum operating frequencies	$V_{DD} = 5\text{ V}$	10, 12.5, 16, 20 MHz (when externally-supplied frequency is 20, 25, 32, and 40 MHz)	
	$V_{DD} = 3\text{ V}$	5, 6.25, 8, 10 MHz (when externally supplied frequency is 10, 12.5, 16, and 20 MHz)	
Minimum instruction execution times	$V_{DD} = 5\text{ V}$	200, 160, 125, 100 ns (when maximum operating frequencies are at 10, 12.5, 16, and 20 MHz)	
	$V_{DD} = 3\text{ V}$	400, 320, 250, 200 ns (when maximum operating frequencies are 5, 6.25, 8, and 10 MHz)	
Multiply/divide instruction execution times	$V_{DD} = 5\text{ V}$	1.9 to 5.6, 1.5 to 4.5, 1.2 to 3.5, 0.95 to 2.8 μs (when maximum operating frequencies are 10, 12.5, 16, and 20 MHz)	
	$V_{DD} = 3\text{ V}$	3.8 to 11.2, 3.0 to 9.0, 2.4 to 7.0, 1.9 to 5.6 μs (when maximum operating frequencies are 5, 6.25, 8, and 10 MHz)	
Memory space		1 MB (512 Kwords)	
I/O area		64 KB (32 Kwords)	
Number of instructions		101	
Number of registers		12	
Clock generator (CG)		<ul style="list-style-type: none"> Variable instruction cycle time function (Variable frequency division ratios of oscillator: 1/2, 1/4, 1/8, or 1/16-divide) 40-MHz Maximum input frequency 	
Wait control unit (WCU)		<ul style="list-style-type: none"> Memory space can be divided into five partitions. I/O space can be divided into three partitions. From zero to three waits can be inserted to the following cycles: CPU cycle, DMA cycle, and refresh cycle 	
Refresh control unit (REFU)		<ul style="list-style-type: none"> Lowest priority refresh/highest priority refresh 7 refresh queues 16-bit refresh address Supports REFRQ expansion timing ($\overline{\text{REFRQ}}$ becomes active from the T1 state) 	
Timer/counter unit (TCU)		<ul style="list-style-type: none"> $\mu\text{PD71054}$ equivalent 	
Serial control unit (SCU)		<ul style="list-style-type: none"> $\mu\text{PD71051}$ (asynchronous) equivalent On-chip dedicated baud rate generator 	

(continued)

Item \ Part Number		V40HL	V50HL
Interrupt control unit (ICU)		<ul style="list-style-type: none"> • μPD71059 equivalent (vector mode only) • μPD71059 can be cascade-connected. 	
DMA control unit (DMAU)		<ul style="list-style-type: none"> • Two operating modes: μPD71071 and μPD71037 modes • 20-bit address bus • Four DMA channels 	
Interrupt functions		7 external interrupts; 3 internal interrupts	
Standby functions	$V_{DD} = 5\text{ V}$	<ul style="list-style-type: none"> • HALT mode: $I_{DD}(\text{MAX.}) = 2.2\text{ mA/MHz}$ • STOP mode: $I_{DD}(\text{MAX.}) = 50\text{ }\mu\text{A}$ 	
	$V_{DD} = 3\text{ V}$	<ul style="list-style-type: none"> • HALT mode: $I_{DD}(\text{MAX.}) = 1.5\text{ mA/MHz}$ • STOP mode: $I_{DD}(\text{MAX.}) = 30\text{ }\mu\text{A}$ 	
Packages		<ul style="list-style-type: none"> • 80-pin plastic QFP (14 x 20-mm body; 2.7-mm resin thickness) • 80-pin plastic TQFP (12 x 12-mm body; 1-mm resin thickness) • 68-pin plastic QFJ 	
Miscellaneous		<ul style="list-style-type: none"> • Internal bus interface unit (BIU) • Internal bus arbitration unit (BAU) 	

[MEMO]

CHAPTER 2 APPLICATIONS

Office equipment, as represented by machines such as dedicated word-processing computers, is becoming more compact today as it shifts from desk-top models to portable models.

The V40HL and V50HL are low-power consumption microprocessors that especially satisfy the demand for compactness in this type of equipment. In addition, because they are able to operate at low voltage, they also accommodate battery-driven systems.

Figure 2-1 shows a block diagram of a typical handy word-processing computer, while Figure 2-2 shows that of a facsimile machine.

Figure 2-1. Handy Word-Processor Block Diagram

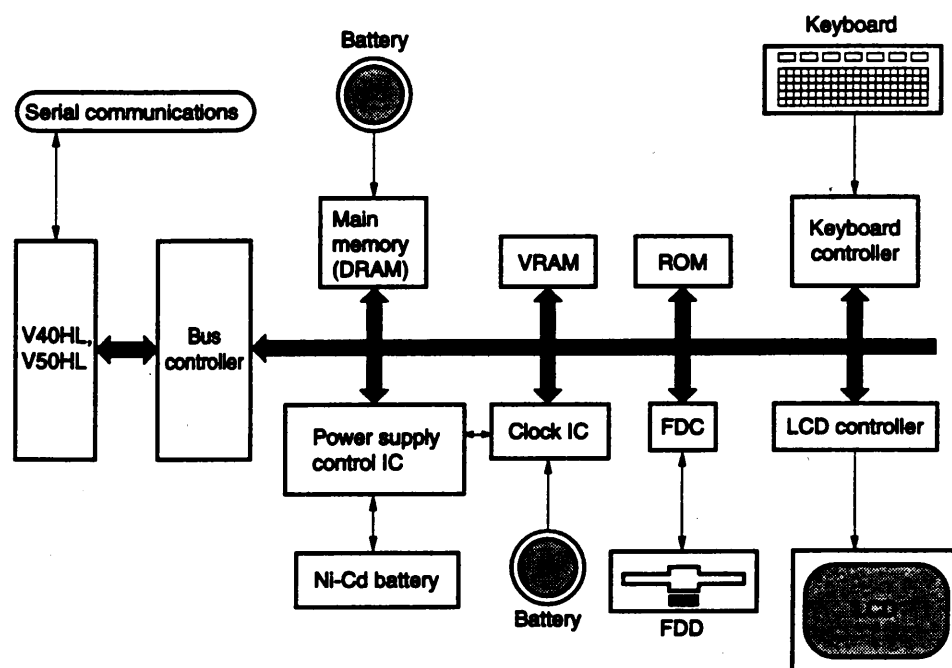
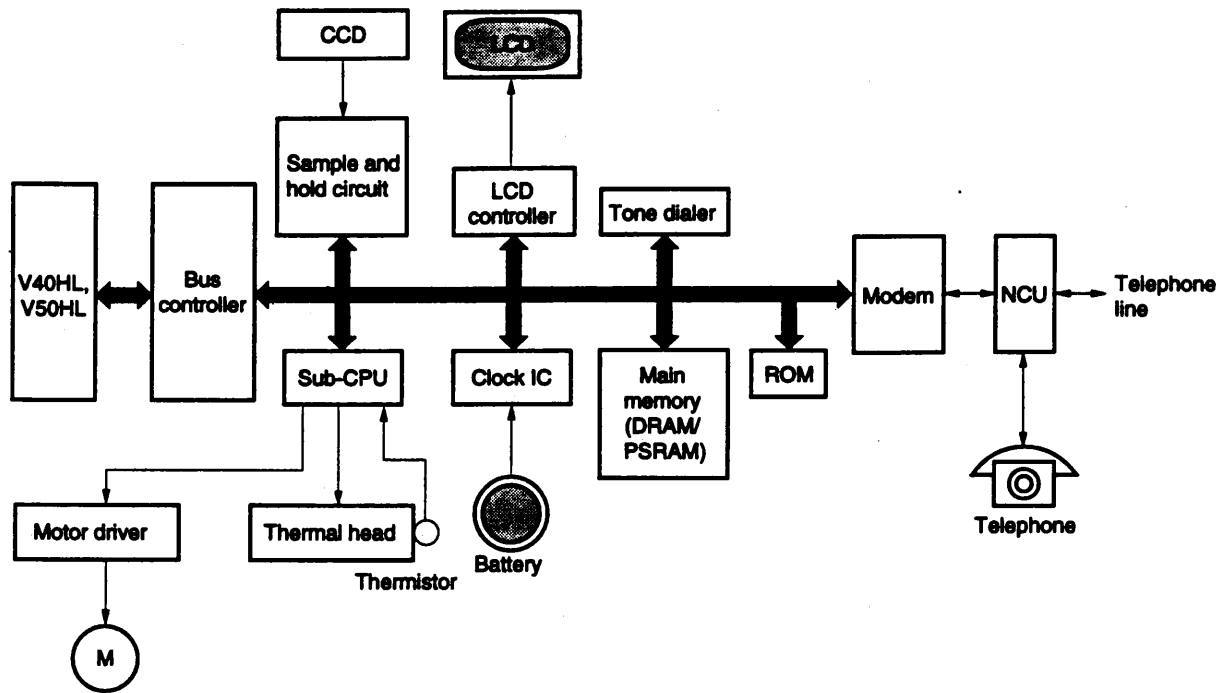


Figure 2-2. Facsimile Machine Block Diagram



CHAPTER 3 PIN FUNCTIONS

3.1 Pin Function List

The functions of each pin are shown below in Table 3-1.

Table 3-1. Pin Functions (1 of 2)

Symbol	Input/Output	Function
AD0 to AD15 ^{Notes 1, 3}	3-state input/output	Time division address/data bus
AD0 to AD7 ^{Notes 2, 3}	3-state input/output	Time division address/data bus
A8 to A15 ^{Notes 2, 3}	3-state output	Address bus
A16/PS0 to A19/PS3 ^{Note 3}	3-state output	Time division address/processor status
REFRQ	Output	Refresh request
HLDRQ	Input	Bus hold request
HLDAK	Output	Bus hold acknowledge
RESET	Input	Reset
RESOUT	Output	System reset output
READY	Input	Bus cycle end
NMI	Input	Nonmaskable interrupt
MRD ^{Note 3}	3-state output	Memory read strobe
MWR ^{Note 3}	3-state output	Memory write strobe
IOR ^{Note 3}	3-state output	I/O read strobe
IOWR ^{Note 3}	3-state output	I/O write strobe
ASTB	Output	Address strobe
UBE ^{Notes 1, 3}	3-state output	Data bus high-order byte enable
High ^{Note 2}	3-state output	High level output
BUSLOCK ^{Note 3}	3-state output	Bus lock
POLL	Input	Polling of floating point operation coprocessor
BUF \overline{R} W ^{Note 3}	3-state output	Buffer read/write
BUFEN ^{Note 3}	3-state output	Buffer enable
X1	Input	Crystal/external clock
X2	—	
CLKOUT	Output	Clock output
BS0 to BS2 ^{Note 3}	3-state output	Bus status
QS0, QS1	Output	Queue status

Table 3-1. Pin Functions (2 of 2)

Symbol	Input/Output	Function
TOUT2	Output	Timer 2 output
TCTL2	Input	Timer 2 control
TCLK	Input	Timer clock
INTP1 to INTP7	Input	Maskable interrupt
INTAK/SRDY/TOUT1	Output	Interrupt acknowledge/serial receive enable/timer 1 output
DMAAK3/TxD	Output	DMA acknowledge 3/serial communications data
DMARQ3/RxD	Input	DMA request 3/serial receive data
DMAAK0 to DMAAK2	Output	DMA acknowledge
DMARQ0 to DMARQ2	Input	DMA request
END/TC	Input/output	Forced DMA service end/DMA service complete
V _{DD}	—	Positive power voltage supply pin
GND	—	Ground potential pin
IC	—	Internally-connected pin (cannot be connected externally)

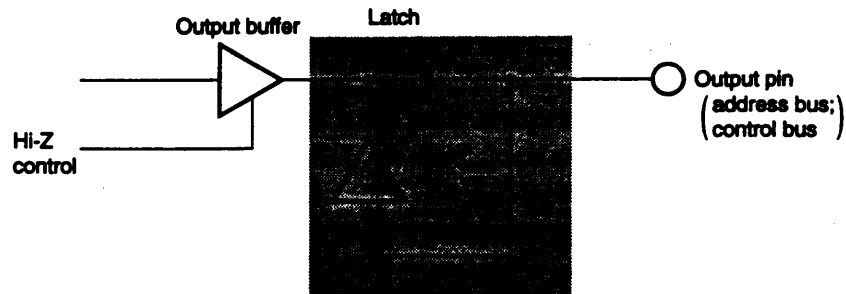
Notes 1. V50HL only

2. V40HL only

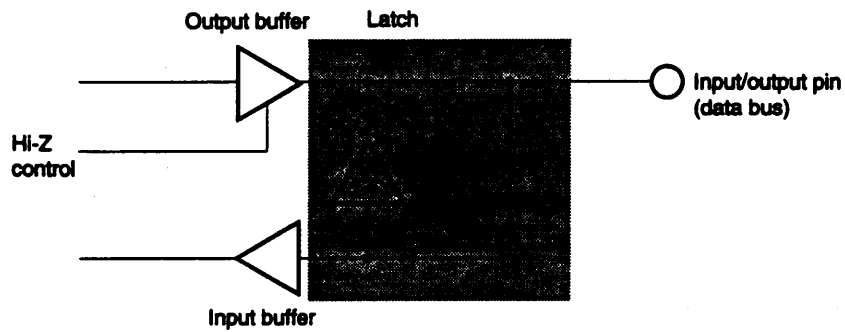
3. Has an internal latch. For this reason, at high impedance, the state prior to becoming high impedance is maintained until driven externally. Thus, pulling the bus up or down is not necessary. In addition, in the high impedance state, to invert the pin level externally, drive capacity at or above latch inversion current (I_{LH} , I_{LL}) is required.

Remark The latch has the circuit composition shown below. When inverting a pin which has a latch, a drive capacity at or above latch inversion current is required.

(1) Output pins



(2) Input/output pins



3.2 Functional Description of Pins

The functions of the V40HL and V50HL pins will be described. The V40HL and V50HL differ on two points only, the number of data bus pins and the \overline{UBE} pin. They have everything else in common.

(1) AD0 to AD15 (Address/Data Bus): Three-State Inputs/Outputs (V50HL)

The following explanation applies only to the V50HL.

The pins indicated above are used for the combined address and data bus. The output of the low-order 16 bits and the input/output of the byte/word data of this 20-bit address data take place in time divisions. During bus cycles, they function as an address bus in the T1 state and as a data bus in the T2, T3, TW and T4 states. The V50HL interfaces by dividing the byte data banks into one that accesses memory or I/O at even-number addresses ($AD0 = 0$) and another that accesses them at odd-number addresses ($AD0 = 1$). The selection between these two byte data banks is controlled by two signals, $AD0$ and \overline{UBE} . The output levels of $AD0$ and \overline{UBE} when accessing an even-number address and an odd-number address are as shown in the following table. For details on this table, refer to section 4.1.1 Memory map and access methods.

Operand	\overline{UBE}	$AD0$	Number of bus cycles
Even-number address word	0	0	1
Odd-number address word	0	1 Note 1	2
	1	0 Note 2	
Even-number address byte	1	0	1
Odd-number address word	0	1	1

- Notes** 1. First bus cycle
2. Second bus cycle

The access of an odd-number address word operand takes place over two bus cycles, dividing the banks into an even-number address and an odd-number address. At this time, $AD0 = 1$, which indicates an odd-number address bank, will be output in the first bus cycle. In the second bus cycle, $AD0 = 0$ will be output automatically to indicate contiguous even-number addresses. In addition, in this case, in the second bus cycle, the address will be output to a segment at a value that has incremented by 1.

During hold acknowledge, these pins are at high impedance.

(2) AD0 to AD7 (Address/Data Bus): Three-State Outputs (V40HL)

The following explanation applies only to the V40HL.

The pins indicated above are used for the combined address and data bus. The output of the low-order 8 bits and input and output of 8-bit data of this 20-bit address data take place in time divisions.

During bus cycles, they function as an address bus in the T1 state and as a data bus in T2, T3, TW and T4 states.

The input and output of 16-bit data takes place over two bus cycles. The first cycle is for the low-order bytes, the second for the high-order bytes.

During hold acknowledge, these pins are at high impedance.

(3) A8 to A15 (Address Bus): Three-State Outputs (V40HL)

This explanation applies only to the V40HL.

These pins output the middle 8 bits of 20-bit address data.

These pins are at high impedance during a hold acknowledge.

(4) A16/PS0 to A19/PS3 (Address Bus/Processor Status): Three-State Outputs

These pins are combined output pins for the address and processor status signals. The contents of each are output in time divisions. During bus cycles, they function as an address bus in the T1 state, and they output the processor statuses in the T2, T3, TW and T4 states.

As an address bus, the high-order 4 bits of this 20-bit address data are output. During the I/O access, zeros are output to all bits.

The processor status signal is output for the access of both memory and I/O. PS3 outputs a zero when in the native mode but not in the refresh cycle and outputs a one at other times. The content of the interrupt enable flag (IE) is output to PS2. PS0 and PS1 show which segment the bus cycle is currently using.

A17/PS1	A16/PS0	Segment
0	0	Data segment 1 (DS1)
0	1	Stack segment (SS)
1	0	Program segment (PS)
1	1	Data segment 0 (DS0)

These pins go to high impedance during a hold acknowledge.

(5) REFRQ (Refresh Request): Output

This signal becomes active-low in the T2, T3 and TW states.

In the case of REFRQ expansion timing, it becomes active-low in the T1, T2, T3 and TW states.

(6) HLDRQ (Hold Request): Input

This is an active-high signal input pin that requests that the V40HL and V50HL release the address bus, address/data bus and the control bus. This signal is sampled every time the CLKOUT signal falls.

The bus use priority due to a requester from HLDRQ is as follows:

REFU (highest priority) > DMAU > HLDRQ > CPU > REFU (lowest priority)

(7) HLDAK (Hold Acknowledge): Output

This signal indicates that the V40HL and V50HL have accepted a hold request signal (HLDRQ) and placed the bus in a high impedance state. During the time this signal is active-high, the address bus, address/data bus and three-state output control signals are at high impedance.

During a hold acknowledge, if a refresh request or DMA request is generated, and both are a higher priority than HLDRQ, HLDAK will become inactive. Then the V40HL and V50HL will request the return of bus control (HLDRQ becomes inactive). In this manner, the high-level width of the HLDAK signal when external and internal bus requests overlap is one clock width at a minimum.

(8) RESET (Reset): Input

A reset is an active-low input, and it has priority over all other operations. In addition to having an impact on the CPU, a reset also has an impact on the internal peripheral units. After a reset has been canceled, the CPU will start a program from address FFFF0H.

★ To reset normally, input a low-level signal of four or more clocks.

However, a $\overline{\text{RESET}}$ when there is a STOP mode cancellation has to be made active during the oscillation stabilization period.

$\overline{\text{RESET}}$ input is also used for the release of the CPU standby mode.

(9) RESOUT (Reset Output): Output

This pin synchronizes an asynchronous signal that is to be input to the $\overline{\text{RESET}}$ pin to the internal clock and then outputs it again in the active-high state. This signal may also be used to reset the system.

(10) READY (Ready): Input

The basic bus cycle of the V40HL and V50HL is that of the clock. When this input signal is made inactive-low, a wait state (TW) will be inserted between the T3 state and the T4 state, lengthening the bus cycle. This function is used for memory or I/O with a slow access time.

This signal is supplied internally to all blocks after synchronization with the clock and is checked in the T2, T3 and TW states. Also, in addition to this signal, the TW state is inserted by WCU. (Refer to section 6.1 Wait Function.)

Note that during internal I/O access, the READY input is invalid.

(11) NMI (Nonmaskable Interrupt Request): Input

This is an interrupt request input in which software masking is not possible. The input is active at the rising edge. It is sampled at each clock. By inputting the high level for five or more clocks (CLKOUT), the NMI will be accepted. When the NMI is accepted, an interrupt for vector number 2 will be generated at the end of an instruction that is being executed.

This NMI input can also be used to cancel the CPU standby mode.

(12) $\overline{\text{MRD}}$ (Memory Read): Three-State Output

This signal becomes active-low when data is read from memory. It also becomes active when memory is refreshed by REFU and when data is transferred from memory to I/O by DMAU. The periods in which $\overline{\text{MRD}}$ becomes active are in the T2, T3 and TW states during bus cycles.

This pin is at high impedance during a hold acknowledge.

(13) $\overline{\text{MWR}}$ (Memory Write): Three-State Output

This signal becomes active-low when data is written to memory. It also becomes active when DMAU transfers data from the I/O to memory. The periods in which $\overline{\text{MWR}}$ becomes active are, if due to the CPU, in the T2, T3 and TW states during bus cycles. If due to DMAU, the normal write timing is in the T3 and TW states, while the expansion write timing is in the T2, T3 and TW states.

This pin is at high impedance during a hold acknowledge.

(14) $\overline{\text{IORD}}$ (I/O Read): Three-State Output

This signal becomes active-low when data is read from the I/O. However, when the I/O to be accessed is an internal peripheral unit, it does not become active. It becomes active when data is being transferred from I/O to memory by DMAU. $\overline{\text{IORD}}$ becomes active during the T2, T3 and TW states in bus cycles.

This pin is at high impedance during a hold acknowledge.

(15) $\overline{\text{IOWR}}$ (I/O Write): Three-State Output

This signal becomes active-low when data is written to the I/O. However, when the I/O to be accessed is an internal peripheral unit, it does not become active. It becomes active when data is being transferred from memory to I/O by DMAU. During CPU bus cycles, $\overline{\text{IOWR}}$ becomes active during the T2, T3 and TW states. If due to DMAU, for normal write cycles, this signal becomes active during the T3 and TW states. For extended write cycles, it becomes active during the T2, T3 and TW states.

This pin is at high impedance during a hold acknowledge.

(16) ASTB (Address Strobe): Output

This is an active-high strobe signal for latching address data to an external latch. When the clock (CLKOUT) of the T1 state is at the low level during bus cycles, it becomes active.

This signal is active low during a hold acknowledge.

(17) \overline{UBE} (Upper Byte Enable): Three-State Output (V50HL)

This signal is valid only with the V50HL, which has a data bus width of 16 bits. In the case of the V40HL, it becomes a high-impedance signal during a hold acknowledge. Otherwise, it outputs the high level. The following explanation applies to the V50HL.

This signal indicates that the high-order eight bits (AD8 to AD15) of AD0 to AD15 will be used in the T2 to T4 states of the bus cycle. This signal is active-low and is output during the T1 to T4 states of the bus cycle. This signal becomes active in the following bus cycles:

- A bus cycle due to a byte access of an odd-number address or the first byte access for word data of an odd-number address.
- A bus cycle due to a word data access of an even-number address.

These bus cycles can be found through a combination of address data and \overline{UBE} signals that are output to the AD0 pin in the T1 state of the bus cycle.

Operand	\overline{UBE}	AD0	Number of bus cycles
Even-number address word	0	0	1
Odd-number address word	0	1 <small>Note 1</small>	2
	1	0 <small>Note 2</small>	
Even-number address byte	1	0	1
Odd-number address byte	0	1	1

- Notes**
1. First bus cycle
 2. Second bus cycle

The \overline{UBE} signal will continue to be at the low level during interrupt acknowledge (even-number address word access required because of vectored read).

This pin is at high impedance during a hold acknowledge.

(18) $\overline{BUSLOCK}$ (Bus Lock): Three-State Output

This signal requests that other master CPUs in a multiprocessor system not use the system bus during the execution of an instruction that follows the $\overline{BUSLOCK}$ prefix and during an interrupt acknowledge cycle.

Hold requests and DMA requests are ignored and refresh requests are held during the bus lock period in which this signal is active-low.

The pin is at high impedance during hold acknowledge.

(19) $\overline{\text{POLL}}$ (Poll): Input

$\overline{\text{POLL}}$ input is checked by the $\overline{\text{POLL}}$ instruction. If it is at the low level, a shift is made to the next instruction. If it is the high level, the $\overline{\text{POLL}}$ input is checked every five clocks until it moves to the low level. This function is used to synchronize the operation of the CPU program and external devices.

(20) $\overline{\text{BUF}\overline{\text{R}}/\text{W}}$ (Buffer Read/Write): Three-State Output

The purpose for outputting signal is to determine the data transfer direction of an external bi-directional buffer. When it is high, data transfer will take place from the V40HL and V50HL to external device. When it is low, data transfer will take place from the external device to the V40HL and V50HL. However, during the CPU data read cycle, it will be at the low level. At other cycles, it will be at the high level. Therefore, it is at the high level during the DMA and refresh cycles.

This pin is at high impedance during a hold acknowledge.

(21) $\overline{\text{BUFEN}}$ (Buffer Enable): Three-State Output

This signal is used as the output enable signal for external bi-directional buffers. In the read cycle and the interrupt acknowledge cycles, it is active-low in the T2 to T4 states. In the write cycle, it is active in the T1 to T4 states. However, $\overline{\text{BUFEN}}$ is not active when accessing I/O within the chip.

This pin is at high impedance during a hold acknowledge.

(22) X1 and X2 (Crystals)

When using an internal clock generator, the crystal resonator is connected to the X1 and X2 pins. The oscillating frequency of the crystal resonator that is connected to the pins is twice that of the operating frequency.

When using an external clock generator, a square wave with a frequency that is twice that of the operating frequency is input to X1. At this time, either input the complement of the X1 pin to X2 or leave it unconnected. A clock that is input externally can be stopped.

(23) CLKOUT (Clock Output): Output

The frequency of the signals on pins X1 and X2 are divided in half and the resulting square wave clock is output. The duty of this output clock is about 50 percent (excluding internal circuit delays).

(24) BS0 to BS2 (Bus Status): Three-State Outputs

These are status signals that inform external bus controllers of what the current bus cycle is.

These signals become active in the T1 and T2 states. They are encoded as indicated in the table below. External bus controllers can decode these signals and generate the control signals for accessing memory and I/O. Only in the HALT mode will the output of BS0 to BS2 be delayed by one clock compared to normal (synchronizes with the rise of T1). Moreover, when a wait state (TW) is inserted, BS0 to BS2 will be active until the CLKOUT fall preceding the last TW state.

BS0	BS1	BS2	Bus Cycle
0	0	0	Interrupt acknowledge
1	0	0	I/O read (CPU internal and external I/O access)
0	1	0	I/O write (CPU internal and external I/O access)
1	1	0	HALT and STOP ^{Note 1}
0	0	1	Program prefetch
1	0	1	Memory read ^{Note 2}
0	1	1	Memory write ^{Note 3}
1	1	1	Passive state ^{Note 4}

Notes 1. HALT mode: Two clocks. Pins BS0 to BS2 output this value and shift to the passive state.

STOP mode: During the STOP mode and after canceling the STOP mode, this value will continue to be output until the oscillation stabilization time has elapsed.

2. In addition to the CPU memory read cycle, the memory read/I/O write DMA cycles, the DMA verify cycle and the refresh cycle are included.
3. In addition to the CPU memory write cycle, the I/O read/memory write DMA cycles are included.
4. In addition to the CPU, DMAU and REFU passive states, the cascade mode DMA cycle is included.

The passive state indicates the state of ending one bus cycle and waiting to move to the next bus cycle.

These pins are at high impedance during a hold acknowledge.

(25) QS0 and QS1 (Queue Status): Outputs

This signal informs the external devices (floating point operation chips) of the status of the instruction queue inside the CPU.

Here, the status of the instruction queue refers to the status when the execution unit (EXU) within the CPU accesses the instruction queue. The data output to the QS0 and QS1 pins is valid only for one clock cycle immediately after a queue access.

QS0	QS1	Instruction Queue Status
0	0	No operation (no change in queue)
1	0	Fetch the first byte of an instruction
0	1	Empty queue
1	1	Fetch from the second byte onward of instruction

This status signal is provided so that a floating point operation chip can monitor the CPU program execution status and synchronize with the CPU to implement processing when control passes to the chip (using a floating point operation (FPO) instruction).

(26) TOUT2 (Timer Output): Output

This is the TCU output pin. Of the three counters, the results of TCT #2 will be output.

(27) TCTL2 (Timer Control): Input

This is the TCU control input pin. Of the three counters, TCT #2 will be controlled.

(28) TCLK (Timer Clock): Input

This is the TCU clock input pin. Whether the clock that is actually input to each counter is the clock from this pin or the frequency-divided V40HL or V50HL operating clock is determined by selection through the software.

(29) INTP1 to INTP7 (Interrupt Request from Peripheral): Inputs

These seven pins are asynchronous interrupt requests from ICU. Either a rising edge trigger or high level trigger can be selected for the input signals. Also, these pin are pulled up internally.

These interrupt request inputs may also be used to cancel the CPU standby mode.

(30) $\overline{\text{INTAK}}$ / $\overline{\text{SRDY}}$ /TOUT1 (Interrupt Acknowledge/Serial Ready/Timer Output): Output

This pin is used for the output of the interrupt acknowledge signal, serial ready signal and the timer output signal (TCT #1).

The interrupt acknowledge signal ($\overline{\text{INTAK}}$) becomes active-low in the T2, T3 and TW states of the interrupt acknowledge cycle.

As for $\overline{\text{SRDY}}$, it outputs the low level when the receiver is enabled due to an output signal from SCU.

TOUT1 is the output of TCU. Of the three counters, the results of TCT #1 are output.

The functions of this pin can be selected through the software using the OPCN register located in the system I/O area. (See section 4.1.3 System I/O area.)

(31) $\overline{\text{DMAAK3}}$ /TxD (DMA Acknowledge/Transmit Data): Output

This pin is used both for the output of the DMAU channel 3 acknowledge signal and the SCU serial data. $\overline{\text{DMAAK3}}$ is active-low.

When functioning as [transmission] TxD, DMAAK3 shifts to the high level (marking status) when there is no data to transmit. When there is data to transmit, it is set in place and the start bit (low) is output automatically.

This data is then output serially. A parity bit and stop bit (high) are added to the end of all data. Adding or not adding the parity bit is programmable.

The functions of this pin can be selected through the software by using the OPCN register located in the system I/O area. (See section 4.1.3 System I/O area)

(32) DMARQ3/RxD (DMA Request/Receive Data): Input

This pin is used both for the DMAU channel 3 request signal and the SCU serial data input. DMARQ3 is active at the high level. DMARQ3 must maintain the high level until $\overline{\text{DMAAK3}}$ becomes active. Also, the withdrawal of DMARQ3 must take place while $\overline{\text{DMAAK3}}$ is active.

In the case of RxD, the high level is input when data is not being transmitted. When it detects the start bit (low), it begins receiving serial data.

In the case of the three pins described above, numbers (30) to (32), the following four selections are possible by using the software to manipulate the OPCN register located in the system I/O area.

Selection \ Pin	$\overline{\text{DMAAK3/TxD}}$	DMARQ3/RxD	$\overline{\text{INTAK/SRDY/TOUT1}}$
1	$\overline{\text{DMAAK3}}$	DMARQ3	$\overline{\text{INTAK}}$
2	$\overline{\text{DMAAK3}}$	DMARQ3	TOUT1
3	TxD	RxD	$\overline{\text{INTAK}}$
4	TxD	RxD	$\overline{\text{SRDY}}$

(33) $\overline{\text{DMAAK0}}$ to $\overline{\text{DMAAK2}}$ (DMA Acknowledge): Outputs

These are DMA acknowledge output pins for DMAU channels 0 to 2.

The signals on these pins are active-low.

(34) DMARQ0 to DMARQ2 (DMA Request): Inputs

These are DMA request output pins for DMAU channels 0 to 2.

The signals on these pins are active-high.

DMARQ must remain at the high level until the corresponding $\overline{\text{DMAAK}}$ becomes active. Also, the withdrawal of DMARQ is required while the corresponding $\overline{\text{DMAAK}}$ is active.

(35) $\overline{\text{END/TC}}$ (End/Terminal Count): Input and Output

This is an active-low input and output pin involved with the end of DMA transfer by DMAU. When a low level pulse ($\overline{\text{END}}$) is input to this pin during DMA transfer, DMAU will end the DMA service in process at that time, even if the service has not been completed. In addition, when the number of DMA transfers set for each channel has ended, it will output a low-level pulse ($\overline{\text{TC}}$).

Because this pin is an open drain, it needs an external pull-up resistor.

(36) V_{DD} (Power)

This is the positive power supply pin. Connect this pin to all common power supplies.

(37) GND (Ground)

This is the grounding pin potential (0 V). Connect this pin to all common grounds.

(38) IC (Internally Connected)

Connect nothing to this pin (in the case of the 80-pin plastic QFP only)

3.3 Pin Input/Output Circuits and Recommended Handling of Unused Pins

Table 3-2 shows the type of input and output circuit for each pin and the recommended handling (recommended connections) of them when they are not used. Figure 3-1 shows a diagram for each type of circuit.

When connecting pins to V_{DD} or to ground through resistors, the recommendation is for using a 1 to 10 k Ω resistor.

Table 3-2. Input/Output Circuits for Each Pin and Recommended Handling When Not Used (1 of 2)

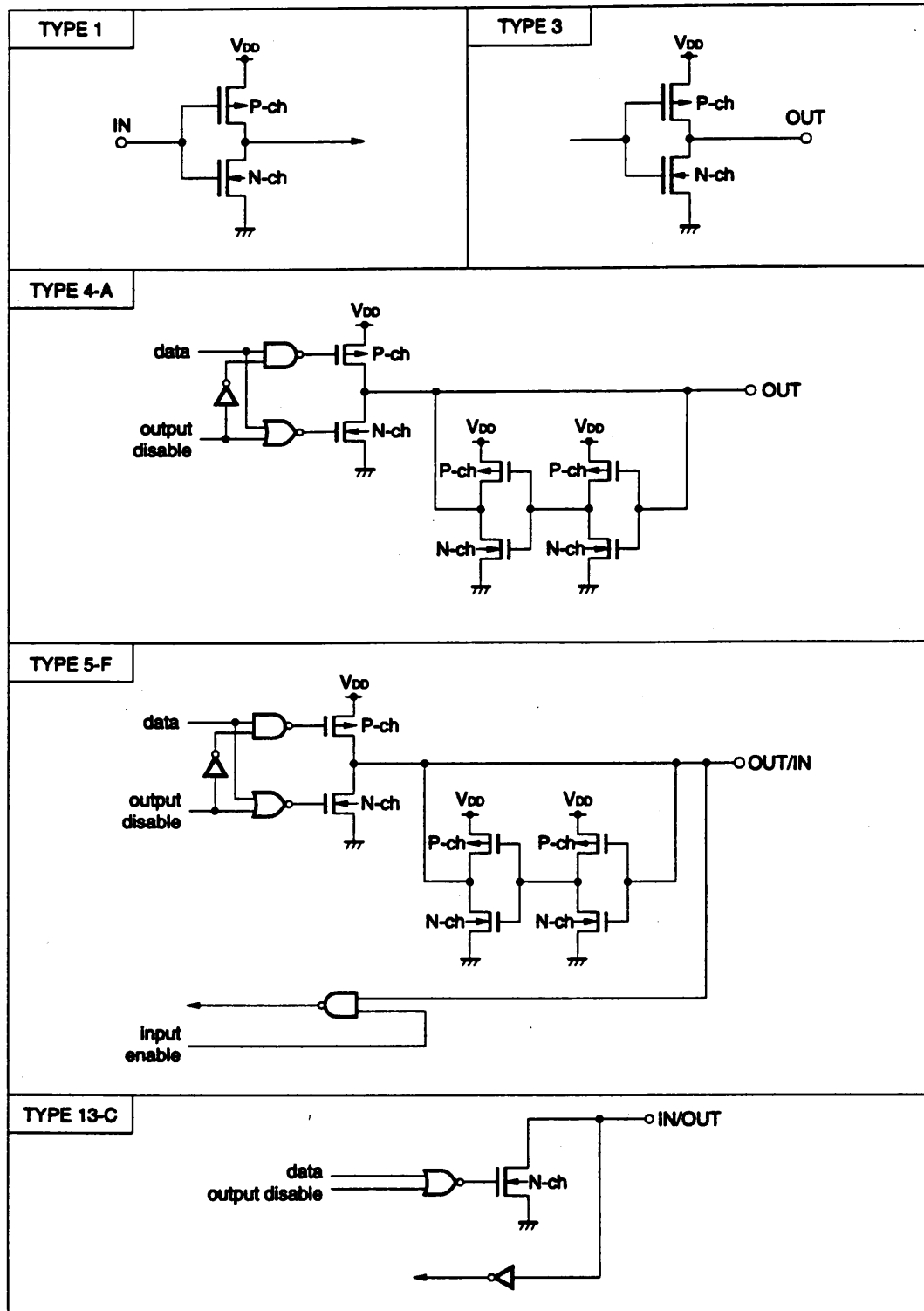
Symbol	Input/Output	Type Input/Output Circuit	Recommended Connection When Not Used
AD0 to AD15 ^{Note 1}	3-state input/output	5-F	Open
AD0 to AD7 ^{Note 2}	3-state input/output	5-F	
A8 to A15 ^{Note 2}	3-state output	4-A	
A16/PS0 to A19/PS3	3-state output	4-A	
$\overline{\text{REFRQ}}$	Output	3	
HLDRQ	Input	1	Connect to ground through a resistor.
HLDAK	Output	3	Open
$\overline{\text{RESET}}$	Input	1	—
RESOUT	Output	3	Open
READY	Input	1	Connect to V _{DD} through a resistor.
NMI	Input	1	Connect to ground through a resistor.
$\overline{\text{MRD}}$	3-state output	4-A	Open
$\overline{\text{MWR}}$	3-state output	4-A	
$\overline{\text{IORD}}$	3-state output	4-A	
$\overline{\text{IOWR}}$	3-state output	4-A	
ASTB	Output	3	
$\overline{\text{UBE}}$ ^{Note 1}	3-state output	4-A	
High ^{Note 2}	Output	4-A	

Notes 1. V50HL only.

2. V40HL only.

Table 3-2. Input and Output Circuits for Each Pin and Recommended Handling When Not Used (2 of 2)

Symbol	Input/Output	Type Input/Output Circuit	Recommended Connection When Not Used
BUSLOCK	3-state output	4-A	Open
POLL	Input	1	Connect to ground through a resistor.
BUF \overline{R} /W	3-state output	4-A	Open
BUFEN	3-state output	4-A	
CLKOUT	Output	3	Open
BS0 to BS2	3-state Output	4-A	
QS0, QS1	Output	3	
TOUT2	Output	3	
TCTL2	Input	1	Connect to ground through a resistor.
TCLK	Input	1	
INTP1 to INTP7	Input	1	Open
INTAK/SRDY/TOUT1	Output	3	
DMAAK3/TxD	Output	3	
DMARQ3/RxD	Input	1	Connect to ground through a resistor.
DMAAK0 to DMAAK2	Output	3	Open
DMARQ0 to DMARQ2	Input	1	Connect to ground through a resistor.
END/TC	Input/output	13-C	Connect separately to V_{∞} through a resistor.

Figure 3-1. Pin Input and Output Circuits

[MEMO]

CHAPTER 4 ARCHITECTURE

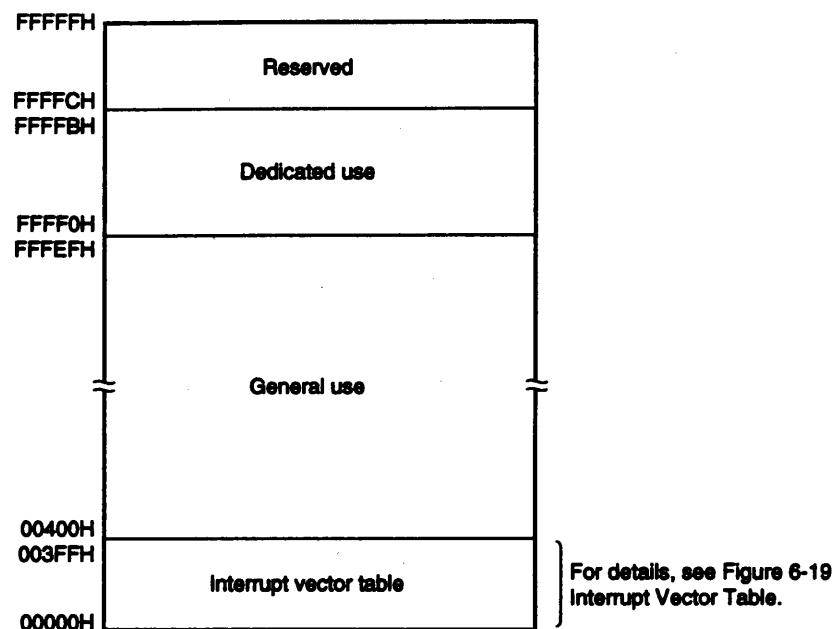
4.1 Memory and I/O Mapping

4.1.1 Memory map and access methods

The V40HL and V50HL have 20-bit address data and are able to access 1 Mbyte (512 Kwords) of memory.

Figure 4-1 shows the memory map. The one kilobytes of memory from 00000H to 003FFH have been allocated to the interrupt vector table. However, table area that is not used by the system can be used for other purposes. The 12 bytes of memory from FFFF0H to FFFFBH are automatically used by the CPU for things such as resets and starts. Therefore, they cannot be used for other purposes. The 4 bytes of memory from FFFFCH to FFFFFH are reserved for future use. As a consequence, they cannot be used.

Figure 4-1. Memory Map



Instruction code, interrupt start addresses, stack data and general variables are elements stored in memory. Some elements are in byte units while others are in word units. The addresses generated by instructions relative to these elements can be both even-number ($A0 = 0$) and odd-number ($A0 = 1$). Excluded from these addresses is the area in which the interrupt start address (interrupt vector table) is stored, which is always an even-number address ($A0 = 0$).

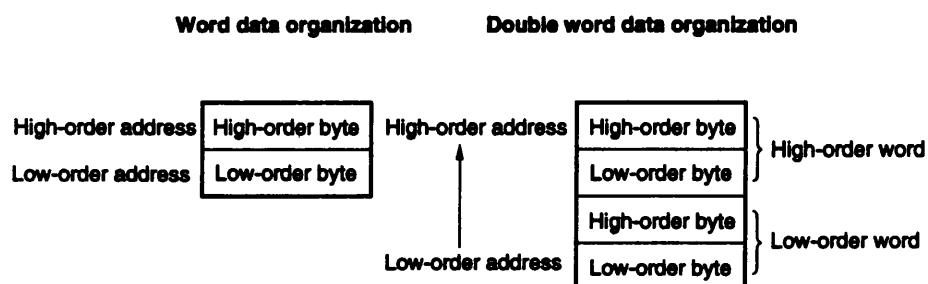
The design is such that the access of V50HL word data is possible whether the address is an even number or an odd number. Both even numbers and odd numbers are permitted as addresses generated by an instruction.

The memory elements and data organization are shown in Table 4-1.

Table 4-1. Memory Element Addresses and Data Organization

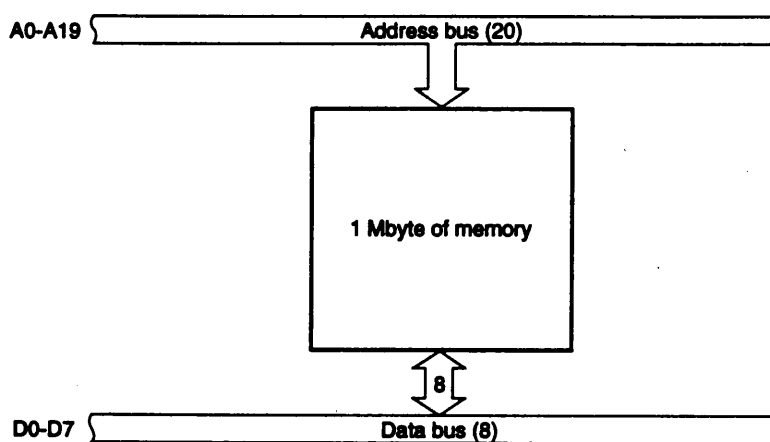
Memory Element	Address	Data Organization
Instruction code	Even/odd numbers	1/2/3/4/5/6 bytes
Interrupt vector table	Even numbers	2 words/vector
Stack	Even/odd numbers	Words
General variables	Even/odd numbers	Bytes/words/double words

Word data and double word data can be understood as follows:



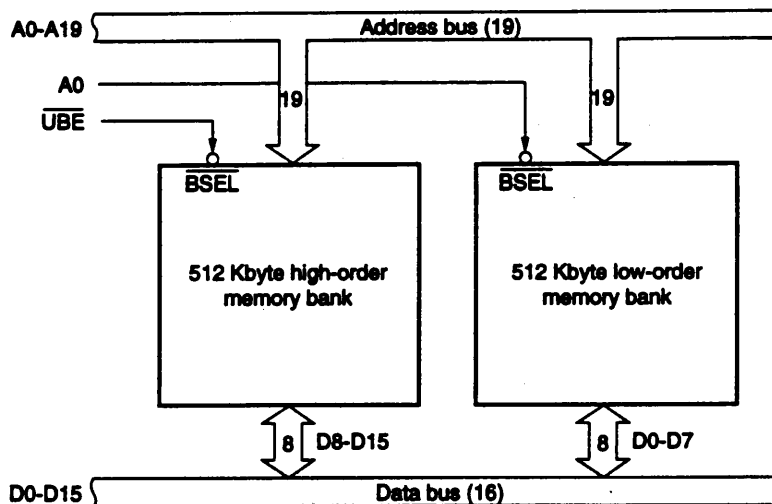
The interfacing with the memory and the CPU differs between the V40HL and V50HL. That of the V40HL is shown in Figure 4-2, while that of the V50HL is shown in Figure 4-3.

Figure 4-2. V40HL Interfacing Between CPU and Memory



In the case of the V40HL, since the width of the data bus is eight bits, the interfacing between the CPU and memory is handled simply as a 1-Mbyte area indicated by 20-bit address data. Thus, byte data will be accessed in one bus cycle and word data in two bus cycles, regardless of whether the address is an even number or odd number.

Figure 4-3. V50HL Interfacing Between CPU and Memory



In the case of the V50HL, since the width of the data bus is 16 bits, it naturally has the capability of transferring 16-bit word data in one bus cycle. However, this applies only when the address generated by the instruction is an even number ($A0 = 0$). When the address is an odd number ($A0 = 1$), two bus cycles are required for data transfer.

As indicated in Figure 4-3, the byte data of the low-order memory bank will be enabled when $A0$ is active-low. However, a \overline{UBE} signal will be output independently of the data from the address bus, and the byte data of the high-order memory bank will be enabled when this signal is active-low.

When accessing the word data of an odd-number address, \overline{UBE} will equal zero and $A0$ will equal one in the first bus cycle, and the high-order bytes will be the only bytes to be accessed. Next, \overline{UBE} will automatically equal one, and the low-order 16 bits of the address data ($A0$ to $A15$) will increment by one. That is, $A0$ will equal zero and the low-order bytes of the next address will be accessed.

The word data of an even-number address will be accessed in one bus cycle due to \overline{UBE} and $A0$ both equaling zero. This is summarized below in Table 4-2.

Table 4-2. V50HL Data Access

Operand	\overline{UBE}	A0	Number of bus cycles
Even-number address word	0	0	1
Odd-number address word	0	1	2
	1	0	
Even-number address byte	1	0	1
Odd-number address byte	0	1	1

The V50HL normally accesses an instruction code (prefetch) in word units. However, when there is a branch to an odd-number address, only one byte of the odd-number address will be fetched. Subsequent fetches will take place once again in word units.

Interrupt vector table access always takes place as the word data of even-number addresses because an even number is always generated when the vector table addresses are generated from the vector numbers (0 to 255). As a consequence, a vector table access for one interrupt always takes place in two bus cycles for two words, one for the segment base and one for the offset.

One bus cycle for a memory access requires four clocks. Consequently, every time there is one access of the word data of an odd-number address, the instruction execution time requires four extra clocks when compared with the access of the word data of an even-number address. In the description of the instruction, if the number of word data accesses is one or more, this will apply when that instruction is executed.

When word data is being transferred from one memory to another memory, two memory accesses are required, one for reading from the source and another for writing to the destination. Thus, the execution time is greatest when both are odd-number addresses.

The problem with these odd-number addresses is the same in the case of a stack operation. Registers are automatically saved to a stack through interrupt processing. However, because these are all word data, consideration must be given to the fact that the number of bus cycles doubles when they are processed at an odd-number address, as this will slow down the interrupt acknowledgment time.

Based on the foregoing information, for accessing word data using the V50HL, items that the programmer can check should all be able to be placed at even-number addresses.

Examples MOV reg,mem instruction execution time (number of clocks)

For byte data : 10 (V40HL and V50HL)

For word data: 14 (V40HL and V50HL using an odd-number address)

10 (V50HL using an even-number address)

The above are examples of word data accesses that take place one time.

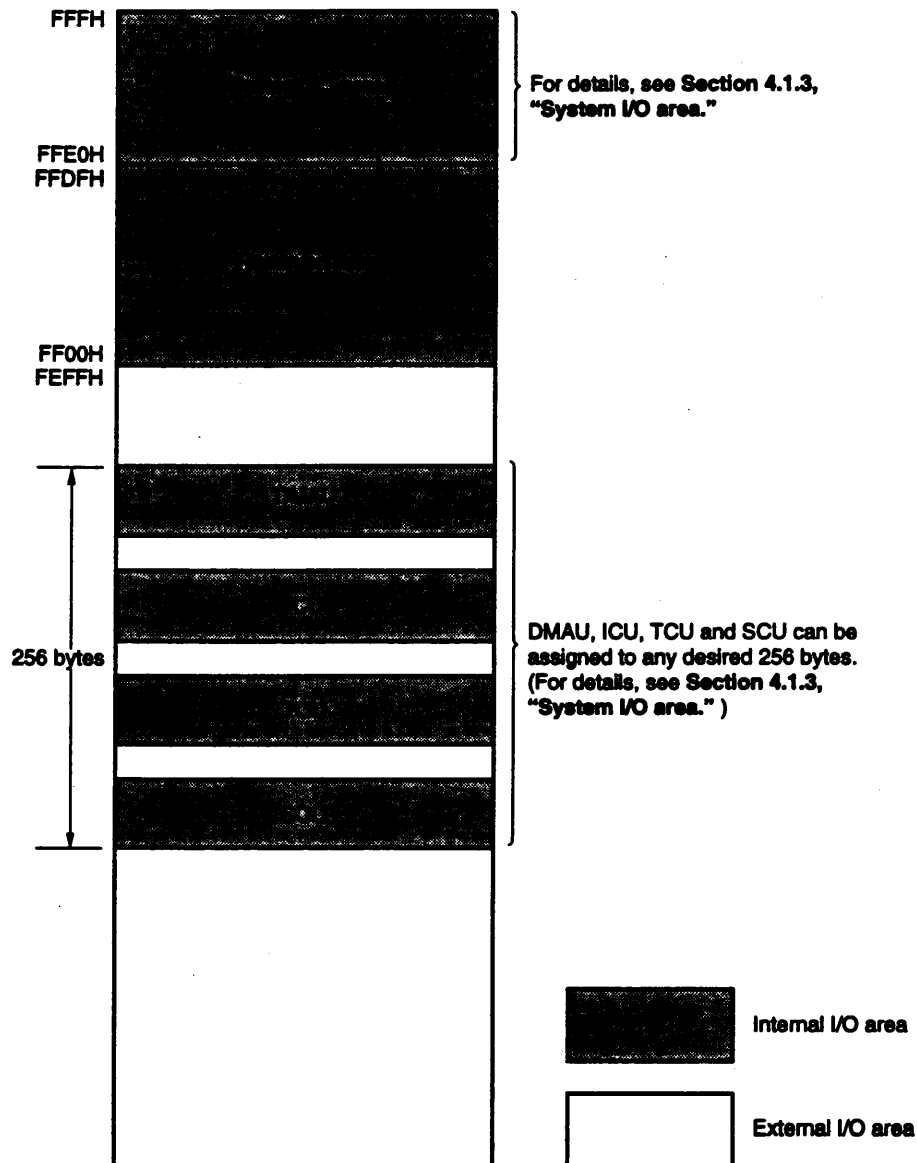
4.1.2 I/O map and access methods

The V40HL and V50HL can access I/O of up to 64 Kbytes (32K words) in areas that are independent of the memory.

The I/O is addressed by the I/O address data that is output from low-order 16 bits of the address bus.

Figure 4-4 shows an I/O map. The 256 bytes from address FF00H to address FFFFH have been set aside for use by the system. They cannot be applied to general-purpose use. This area is composed of a reserved area (FF00H to FFDFH) and a system I/O area (FFE0H to FFFFH). The internal peripheral unit related registers are assigned to the system I/O area.

Figure 4-4. I/O Map



The segment method does not apply to an I/O address as it does to memory.

When an I/O address is output, all zeros are output to the high-order four bits of the address bus (A16-A19).

Both byte units and word units are possible when transferring data between the CPU and I/O. Both 8-bit I/O devices and 16-bit I/O devices can be connected. However, in the case of the V50HL, as with memory, one bus cycle is required for a word data access if the address is an even number and two bus cycles are required if the address is an odd number.

When an 8-bit I/O device is accessed with the V50HL, among the I/O address data, A0 is used only for device selection. Values from A1 and above are used for device selection and for selecting a number of registers within one device. In other words, for an I/O device with an even-number address, its internal registers will all be selected with an even number, while for an I/O device with an odd-number address, its internal registers will all be selected with an odd number.

By using the memory mapped I/O organization (by assigning areas with memory to I/O use), I/O can be placed in the 1 Mbyte memory area, not the I/O area.

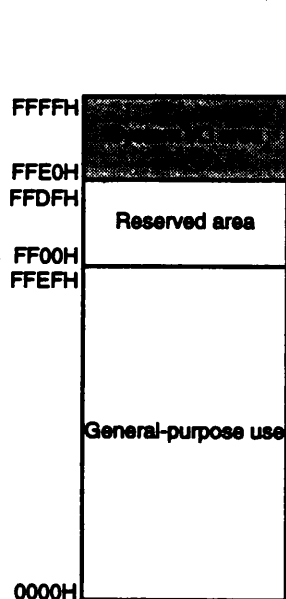
By making using of this function, the abundant addressing modes and arithmetic processing for the memory can take place directly for the I/O device. For example, by using the bit manipulation instructions for the memory, it is possible to test (1 and 0 determination), set (1), clear (0) and invert a certain line of an I/O port.

However, with memory mapped I/O, the control signals output from the CPU all become signals for the memory. I/O devices are distinguished from memory only by the address data. Therefore, special care must be taken so that addresses assigned to variables and stack data do not conflict with addresses assigned to I/O devices.

4.1.3 System I/O area

System I/O area refers to the area that occupies the highest-order 32 bytes (FFE0H-FFFFH) of the 64-Kbyte I/O area (0000H-FFFFH). The register for mapping the internal peripheral units to the I/O area and the control register are mapped within this system I/O area. To use the V40HL and V50HL, these I/Os have to be initialized first. The table below shows the registers that are mapped to the system I/O area and a summary of their major functions.

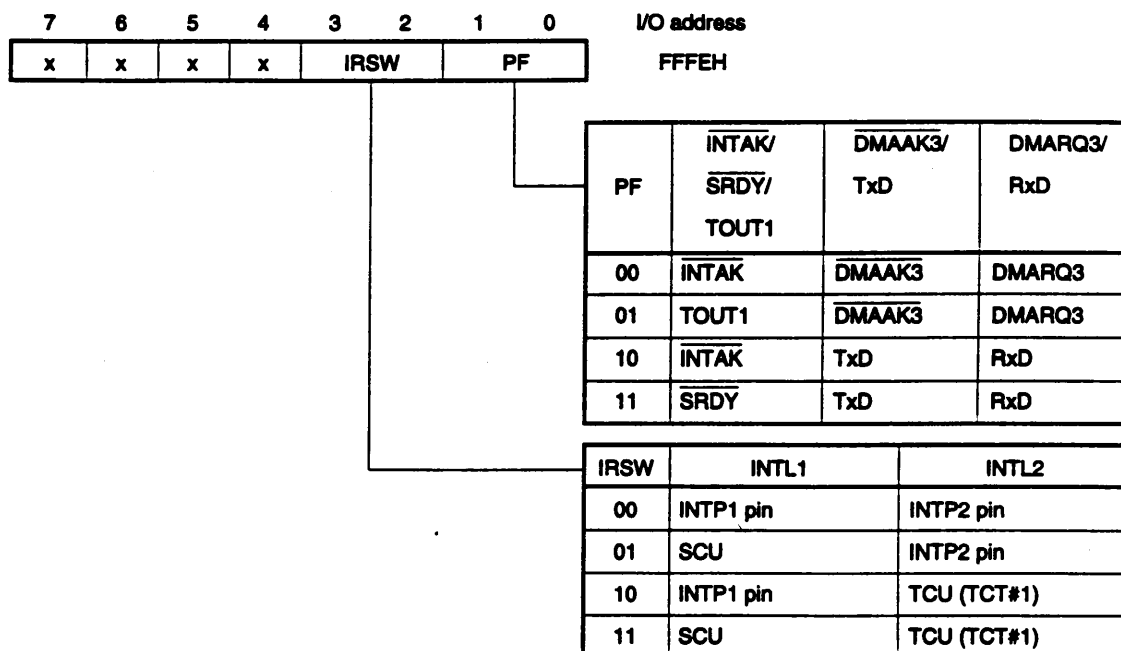
Figure 4-5 shows a list of registers in the system I/O area. For the functions of each register, refer to the sections indicated in the reference column.

Figure 4-5. List of Registers in the System I/O Area


Address	Register Name	Operation	Reference
FFFEH	OPCN	Read/write	See Figure 4-6.
FFFDH	OPSEL	Read/write	See Figure 4-7.
FFFCB	OPHA	Read/write	See Figure 4-10.
FFFBH	DULA	Read/write	See Figure 4-14.
FFFAH	IULA	Read/write	See Figure 4-13.
FFF9H	TULA	Read/write	See Figure 4-11.
FFF8H	SULA	Read/write	See Figure 4-12.
FFF7H	SCTL	Read/write	See Figure 4-8.
FFF6H	WCY2	Read/write	See Figure 6-12.
FFF5H	WCY1	Read/write	See Figure 6-10.
FFF4H	WMB	Read/write	See Figure 6-3.
FFF3H	Reserved	—	—
FFF2H	RFC	Read/write	See Figure 6-15.
FFF1H	SBCR	Read/write	See Figure 6-23.
FFF0H	TCKS	Read/write	See Figure 5-14.
FFE0H	Reserved	—	—
FFE0H	Reserved	—	—
FFEDH	EXWB	Read/write	See Figure 6-5.
FFECH	WSMB	Read/write	See Figure 6-8.
FFEBH	WIOB	Read/write	See Figure 6-9.
FFEAH	WCY3	Read/write	See Figure 6-11.
FFE9H	BRC	Read/write	See Figure 5-36.
FFE8H to FFE2H	Reserved	—	—
FFE1H	BADR	Read/write	See Figure 5-83.
FFE0H	BSEL	Read/write	See Figure 5-84.

(1) On-Chip Peripheral Connection Register (OPCN)

Figure 4-6. On-Chip Peripheral Connection Register (OPCN)



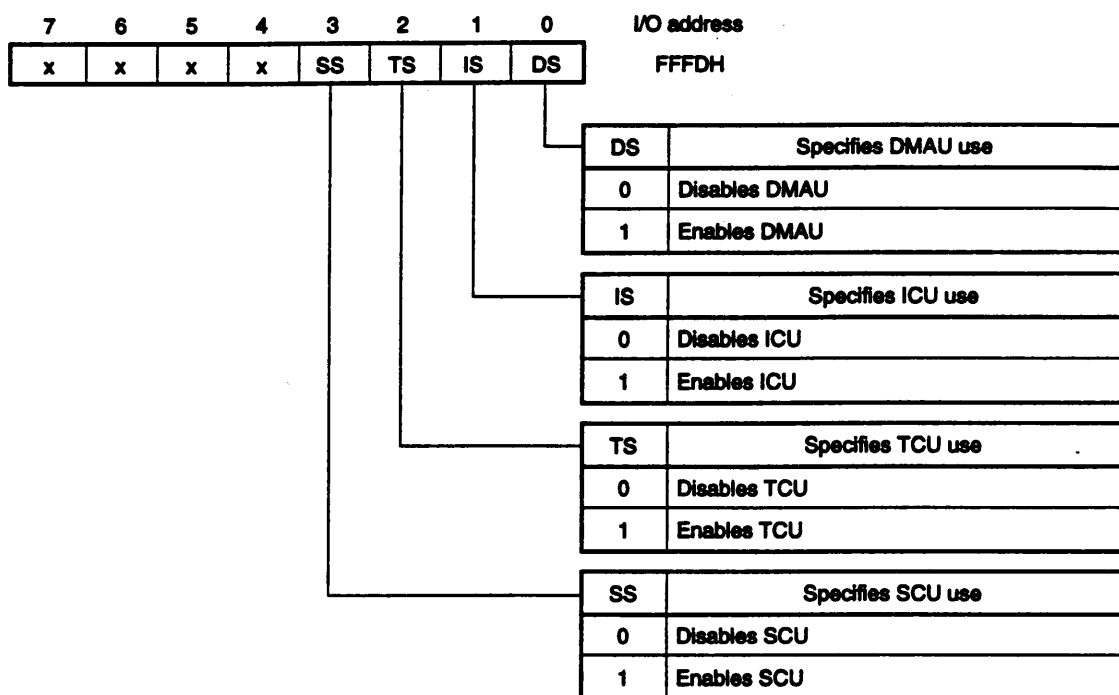
Remark x: don't care

(a) PF Bit

This bit selects the functions of three pins, $\overline{\text{INTAK}}$ / $\overline{\text{SRDY}}$ / $\overline{\text{TOUT1}}$, $\overline{\text{DMAAK3}}$ /TxD and DMARQ3/RxD. The functions of a combined-use pin cannot be selected in any combination desired. There are only four possible combinations. If DMA is to be used often, they are PF = 00 or PF = 01. For serial input and output, it is PF = 1x (where x is an optional 1 or 0.)

(b) IRSW Bit

This bit switches the signal input to INTL1 and INTL2 of the ICU. For the input signal, an external pin or the output of SCU or TCU can be selected.

(2) On-Chip Peripheral Selection Register (OPSEL)**Figure 4-7. On-Chip Peripheral Selection Register (OPSEL)**

Remark x: don't care

OPSEL has the following functions:

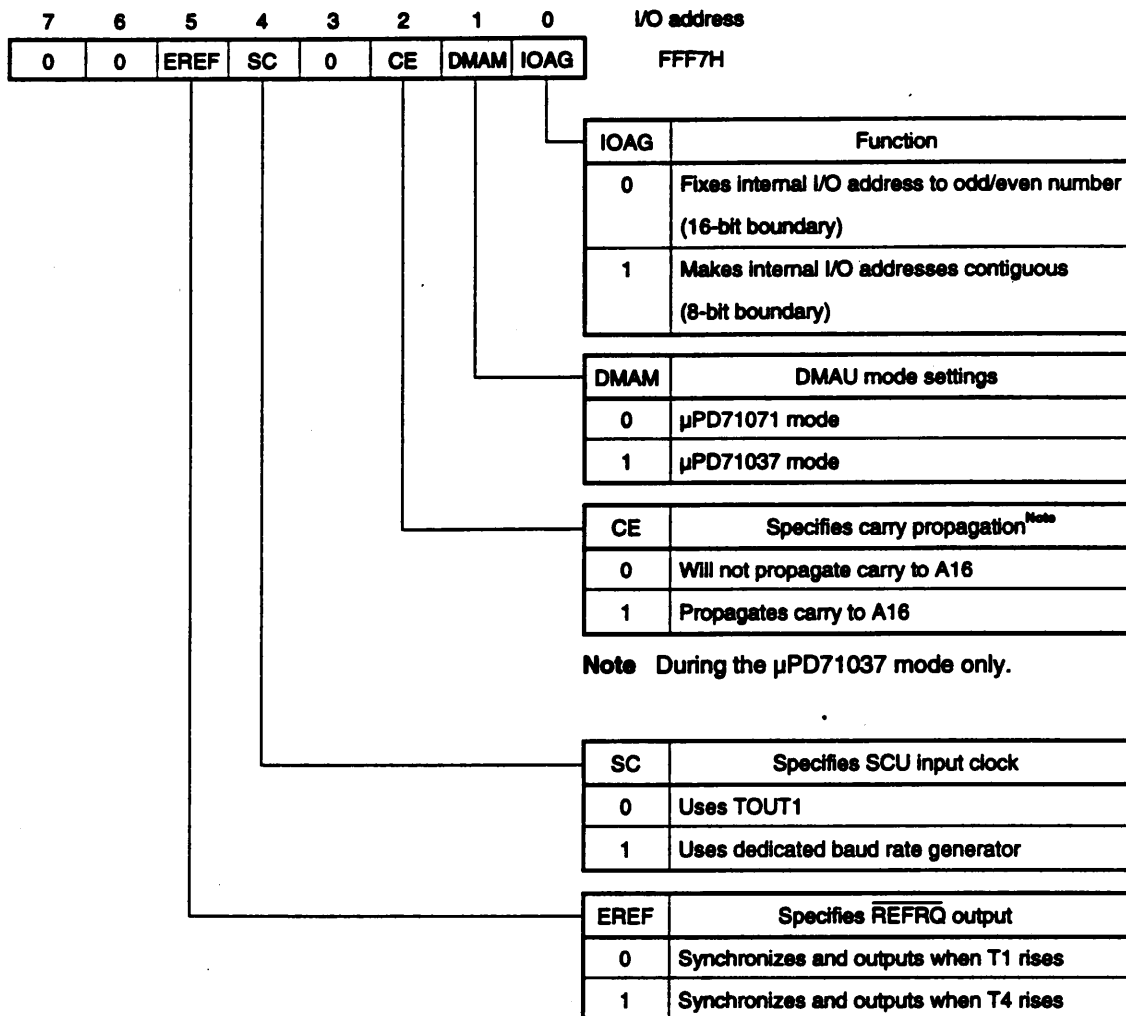
- DMAU enable/disable setting function (DS bit)
- ICU enable/disable setting function (IS bit)
- TCU enable/disable setting function (TS bit)
- SCU enable/disable setting function (SS bit)

The disabled state means that programming in that I/O area cannot take place. That is, this register has to be used to enable each internal peripheral unit before it can be programmed.

Caution Set OPHA before setting OPSEL. (See number (4).)

(3) System Control Register (SCTL)

Figure 4-8. System Control Register (SCTL)



Caution Be sure to set bits 3, 6 and 7 to zero.

(a) IOAG Bit

The I/O addresses of TCU, ICU, SCU and DMAU can be set by OPHA, DULA, IULA, TULA and SULA (see number (4)). When doing this, place the least significant bit A0 in the 16-bit boundary as fixed or place it in the 8-bit boundary without fixing it.

(b) DMAM Bit

The DMAU has a μ PD71071 mode and a μ PD71037 mode. This bit selects which of these modes will be used. When using the DMAU, be sure to set this bit.

(c) CE Bit

This bit sets whether or not to propagate a carry from DMA address A15 to A16 when using the DMAU in the μ PD71037 mode.

(d) SC Bit

This bit sets whether to use the dedicated baud rate generator or the output of the TCU TOUT1 as the SCU transmit/receive clock.

(e) EREF Bit

During the refresh cycle, this bit selects whether to synchronize the $\overline{\text{REFRQ}}$ output timing to the T4 state just before the refresh cycle or to synchronize it to the T1 state of the refresh cycle.

(4) On-Chip Peripheral Relocation Registers (OPHA, DULA, IULA, TULA, SULA)

Five registers, OPHA, TULA SULA, IULA and DULA, determine the I/O address for the internal peripheral units, TCU, SCU, ICU and DMAU.

OPHA is common to all four of these units and sets the high-order 8 bits of each of their I/O address. That is, these four peripheral units are located in the I/O area of the 256 bytes defined using OPHA. Next, the low-order address registers of DULA, IUTA, TULA and SULA determine where within these 256 bytes each peripheral unit is located. As for the low-order addresses, the bit to be set will change depending on the value of the SCTL IOAG bit (see number (3)).

Figure 4-9 shows a mapping example of an internal peripheral unit. Figures 4-10 to 4-14 show details of each register.

- Cautions**
1. In the I/O area, DMAU occupies 16 bytes and ICU, TCU and SCU each occupy 4 bytes. Be careful that they do not overlap.
 2. Use byte-type IN/OUT instructions for accesses to units other than DMAU.
 3. I/O areas that are not internal peripheral unit I/O areas, reserved areas or the system I/O area can also be used as external I/O areas.

Figure 4-9. Internal Peripheral Unit Mapping Example

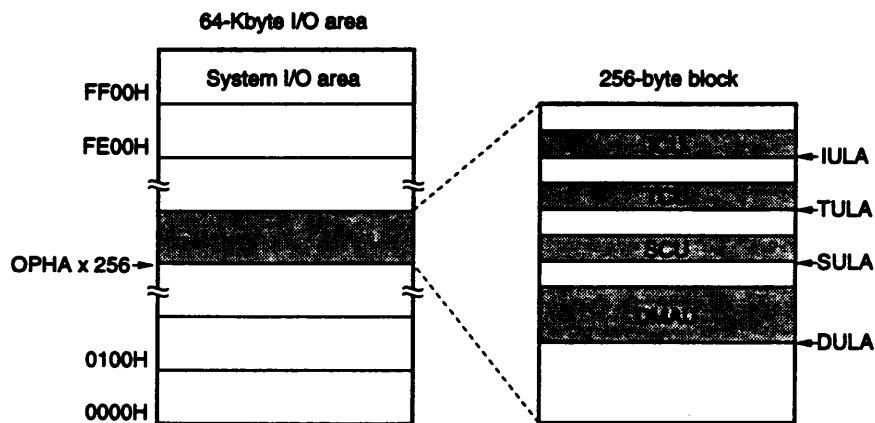


Figure 4-10. On-Chip Peripheral High Address Register (OPHA)

7	6	5	4	3	2	1	0	I/O address
A15	A14	A13	A12	A11	A10	A9	A8	FFFCH

Figure 4-11. TCU Low Address Register (TULA)**(a) When IOAG = 0**

7	6	5	4	3	2	1	0	I/O address
A7	A6	A5	A4	A3	—	—	A0	FFF9H

(b) When IOAG = 1

7	6	5	4	3	2	1	0	I/O address
A7	A6	A5	A4	A3	A2	—	—	FFF9H

Remark For the detailed mapping of all registers of TCU, see section 5.7 Timer/Counter Unit (TCU).

Figure 4-12. SCU Low Address Register (SULA)**(a) When IOAG = 0**

7	6	5	4	3	2	1	0	I/O address
A7	A6	A5	A4	A3	—	—	A0	FFF8H

(b) When IOAG = 1

7	6	5	4	3	2	1	0	I/O address
A7	A6	A5	A4	A3	A2	—	—	FFF8H

Remark For the detailed mapping of all registers of SCU, see section 5.8 Serial Control Unit (SCU).

Figure 4-13. ICU Low Address Register (IULA)

(a) When IOAG = 0

7	6	5	4	3	2	1	0	I/O address
A7	A6	A5	A4	A3	—	—	A0	FFFAH

(b) When IOAG = 1

7	6	5	4	3	2	1	0	I/O address
A7	A6	A5	A4	A3	A2	—	—	FFFAH

Remark For the detailed mapping of all registers in ICU, see section 5.9 Interrupt Control Unit (ICU).

Figure 4-14. DMAU Low Address Register (DULA)

(a) When in the μ PD71071 mode. IOAG may equal 1 or 0.

7	6	5	4	3	2	1	0	I/O address
A7	A6	A5	A4	—	—	—	—	FFFBH

(b) When in the μ PD71037 mode. When IOAG = 0.

7	6	5	4	3	2	1	0	I/O address
A7	A6	A5	—	—	—	—	A0	FFFBH

(c) When in the μ PD71037 mode. When IOAG = 1.

7	6	5	4	3	2	1	0	I/O address
A7	A6	A5	A4	—	—	—	—	FFFBH

Remark For the detailed mapping of all registers in DMAU, see section 5.10 DMA Controller Unit (DMAU).

4.1.4 Memory I/O read and write timing

The read and write timing of the memory and I/O of the V40HL and V50HL are shown in Figures 4-15 to 4-22.

The access of one memory or I/O operand (a read or write) takes place all in one bus cycle. Basically, one bus cycle consists of four states (four clocks), from T1 to T4. At 16-MHz operating time, one state takes 62.5 nanoseconds.

With the V40HL and V50HL, an instruction fetch and a data read are processed in exactly the same timing (Figures 4-15 to 4-19).

If a lot of internal processing time is required for the execution of a certain instruction, inside of the CPU, after the execution unit (EXU) fetches the code of that instruction from the queue and begins its execution, the bus controller unit (BCU) will continue to prefetch from the queue until the queue becomes full.

Even when the queue is full, if EXU is executing an instruction and does not fetch the instruction code from the queue, BCU will terminate the next prefetch and automatically insert an idle state (T1) after the T3 state. The T1 state will continue to be inserted until the instruction executing at that time terminates and the next instruction code is fetched from the queue. After this, BCU will move to the T4 and T1 states and the bus cycle state.

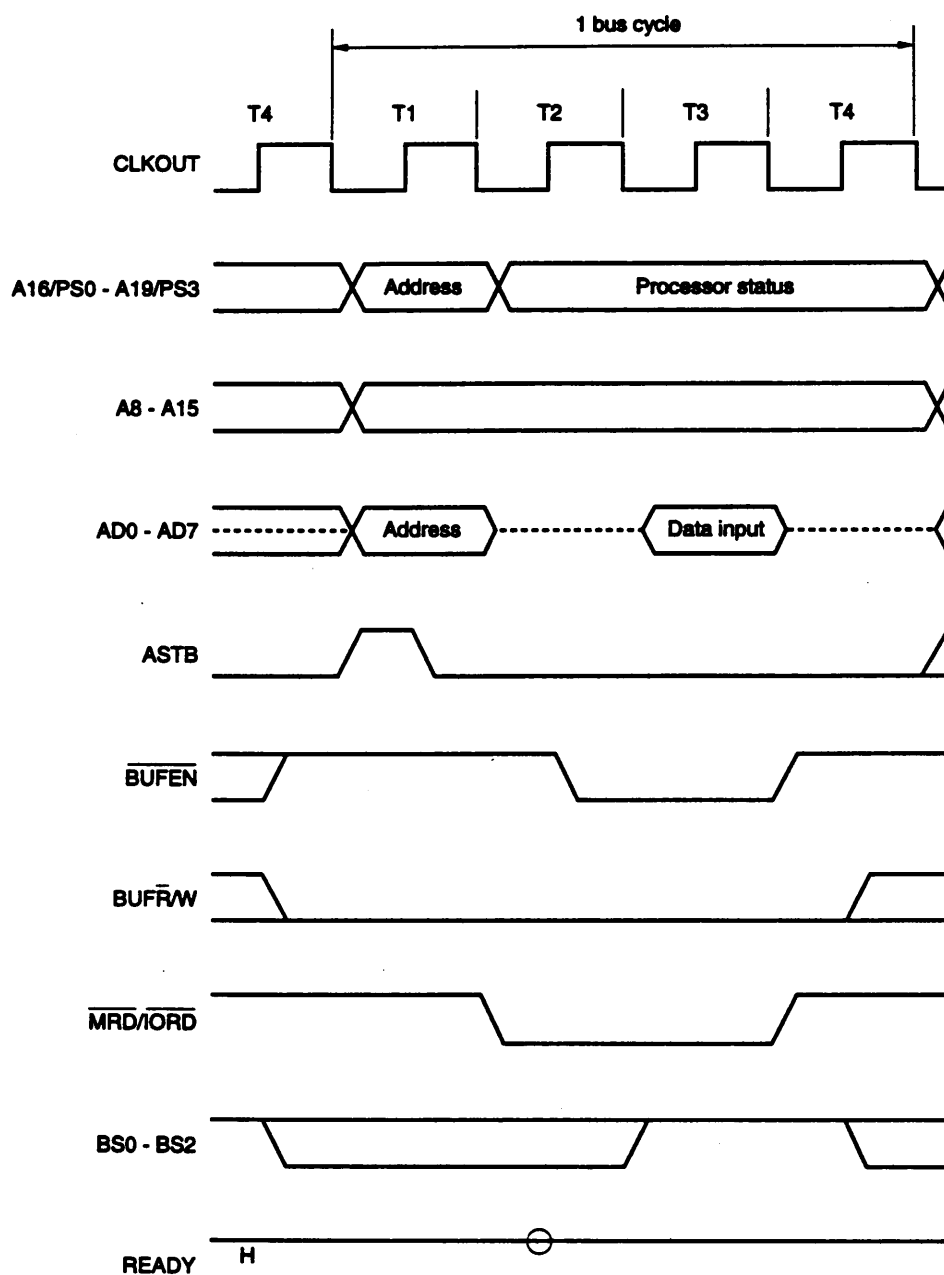
In addition, when a memory or I/O device with a slow access time is used, BIU will detect the READY signal sent from the memory, I/O device and WCU. If the READY signal is inactive, it will insert a wait state (TW) and not advance to the T4 state, the state after T3. When the READY signal becomes high, it will advance the states of T4, T1 and the bus cycle. For details, refer to section 6.1.3 Relationship between WCU and READY pin.

During the period in which TW is inserted, all signals are maintained at the same level and the read and write timing will expand.

In the case of an I/O access, the timing is the same for an external I/O access and a V40HL/V50HL internal I/O access. However, when there is an internal access, the $\overline{\text{IORD}}$, $\overline{\text{IOWR}}$ and $\overline{\text{BUFEN}}$ pins will not become active.

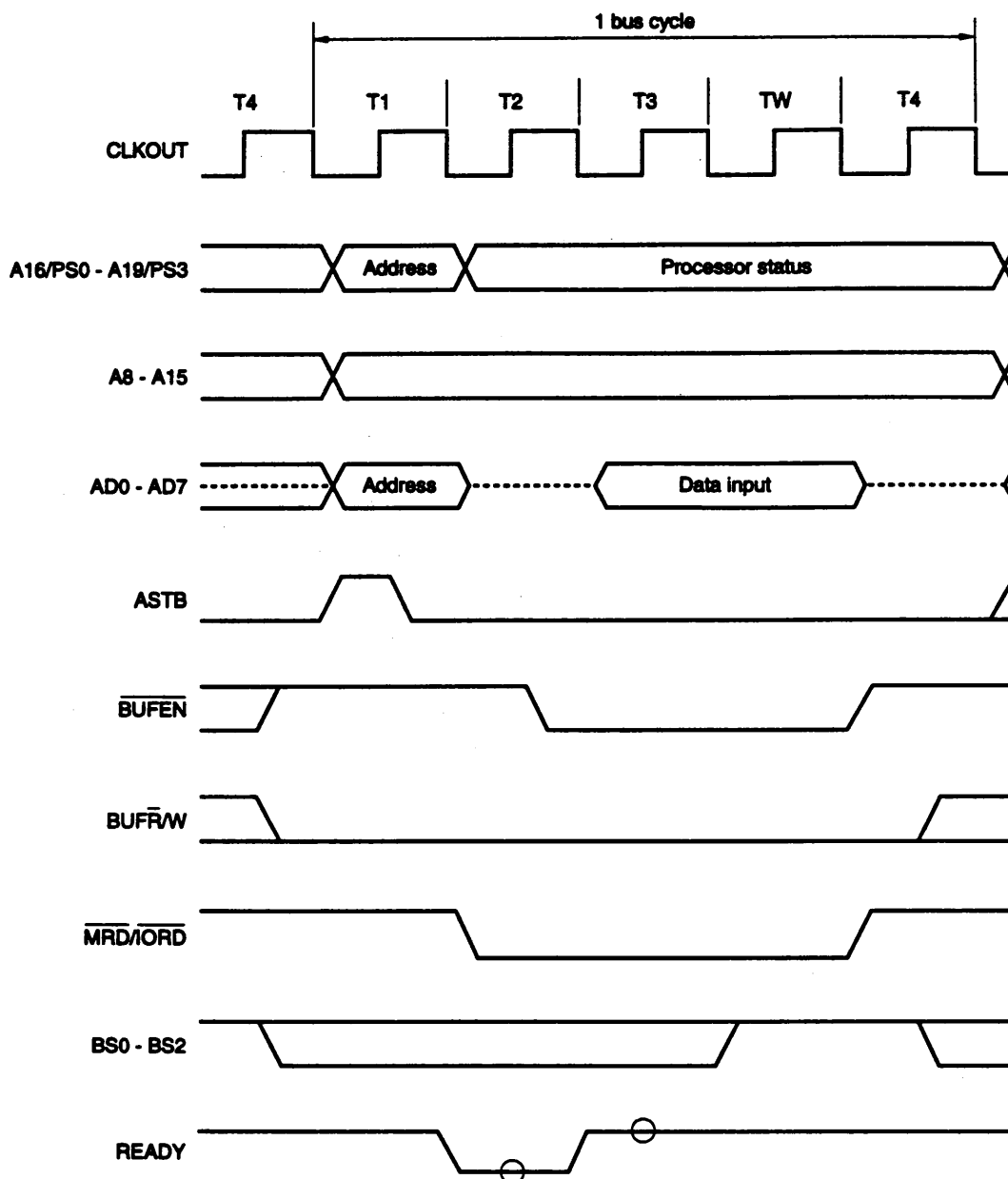
Figure 4-15. Memory and External I/O Read Timing (V40HL) (1 of 2)

(a) No wait



Remarks 1. Broken lines indicate high impedance.

2. A circle indicates the timing of the sampling.

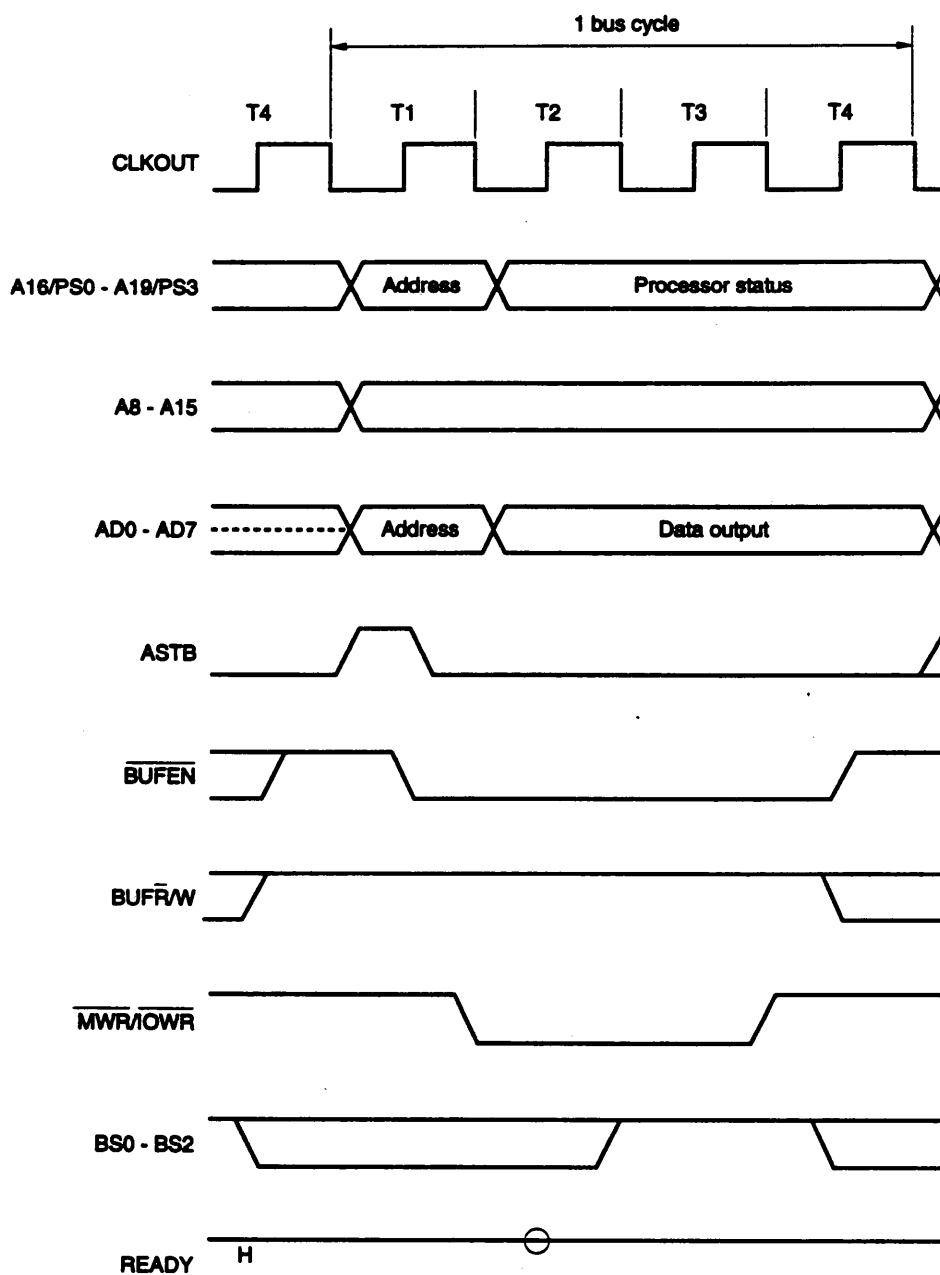
Figure 4-15. Memory and External I/O Read Timing (V40HL) (2 of 2)**(b) One wait**

Remarks 1. Broken line indicates high impedance.

2. Circles indicate the sampling timing when the wait cycle is set to zero in WCU. Refer to **section 6.1.3 Relationship between WCU and READY pin.**

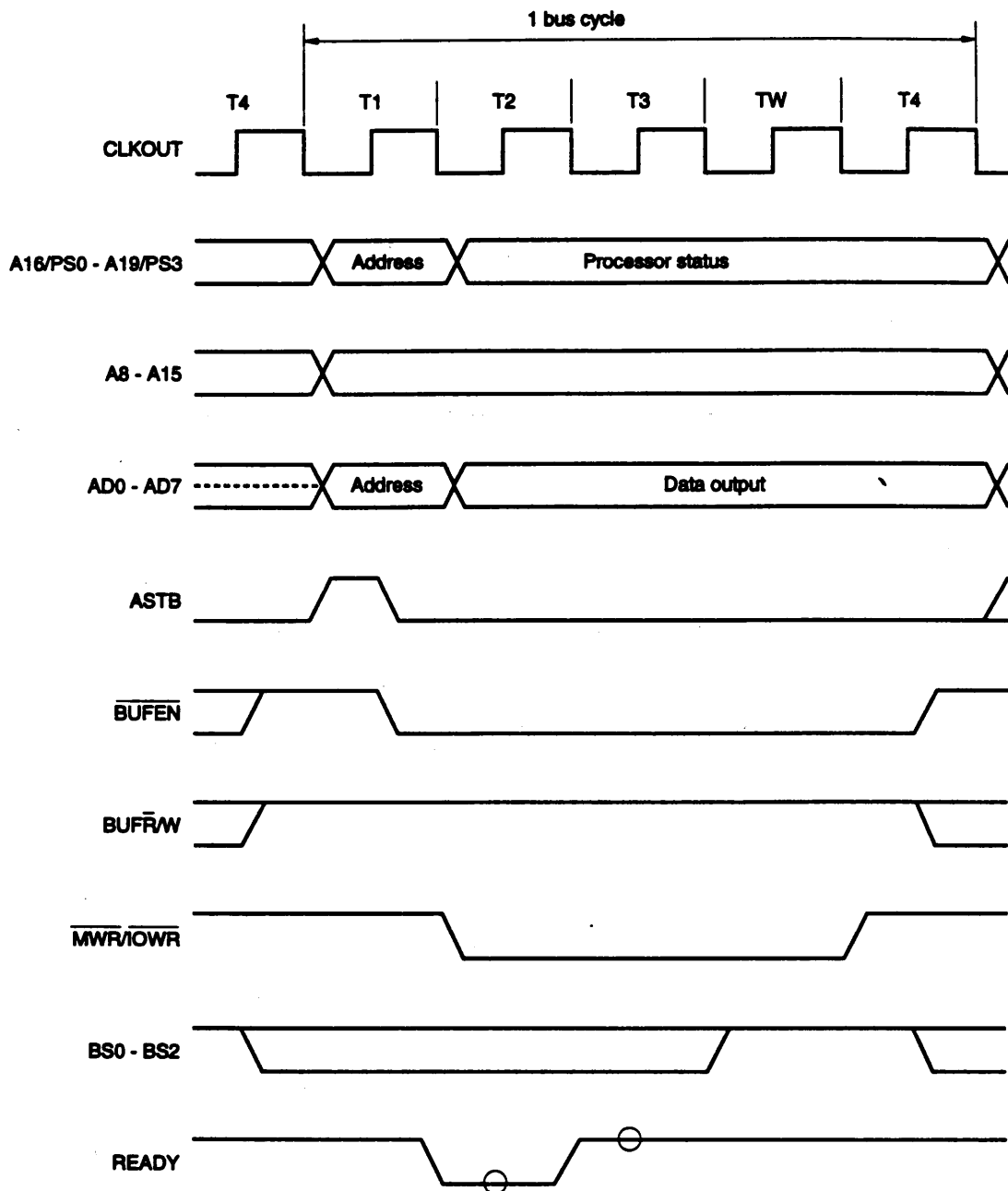
Figure 4-16. Memory and External I/O Write Timing (V40HL) (1 of 2)

(a) No wait



Remarks 1. Broken line indicates high impedance.

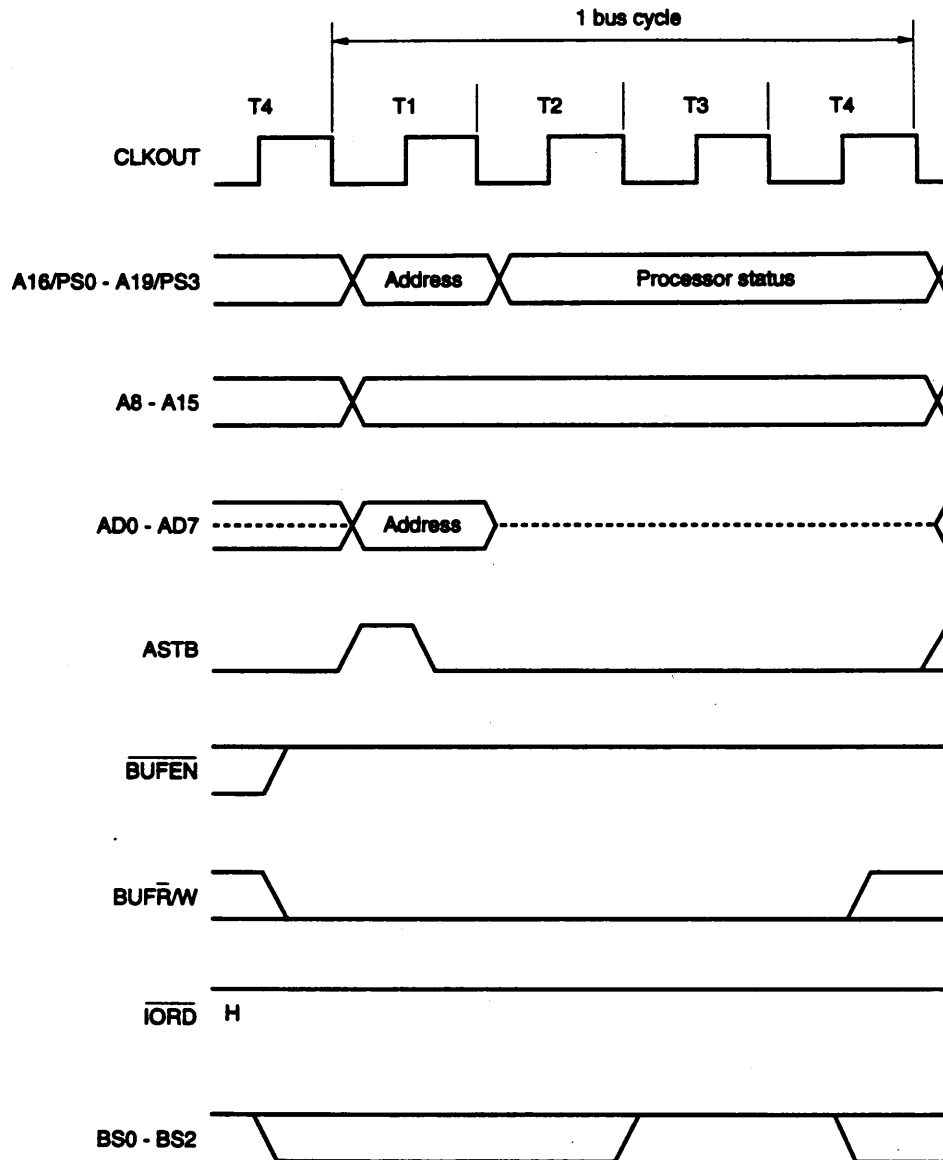
2. A circle indicates the timing of the sampling.

Figure 4-16. Memory and External I/O Write Timing (V40HL) (2 of 2)**(b) One wait**

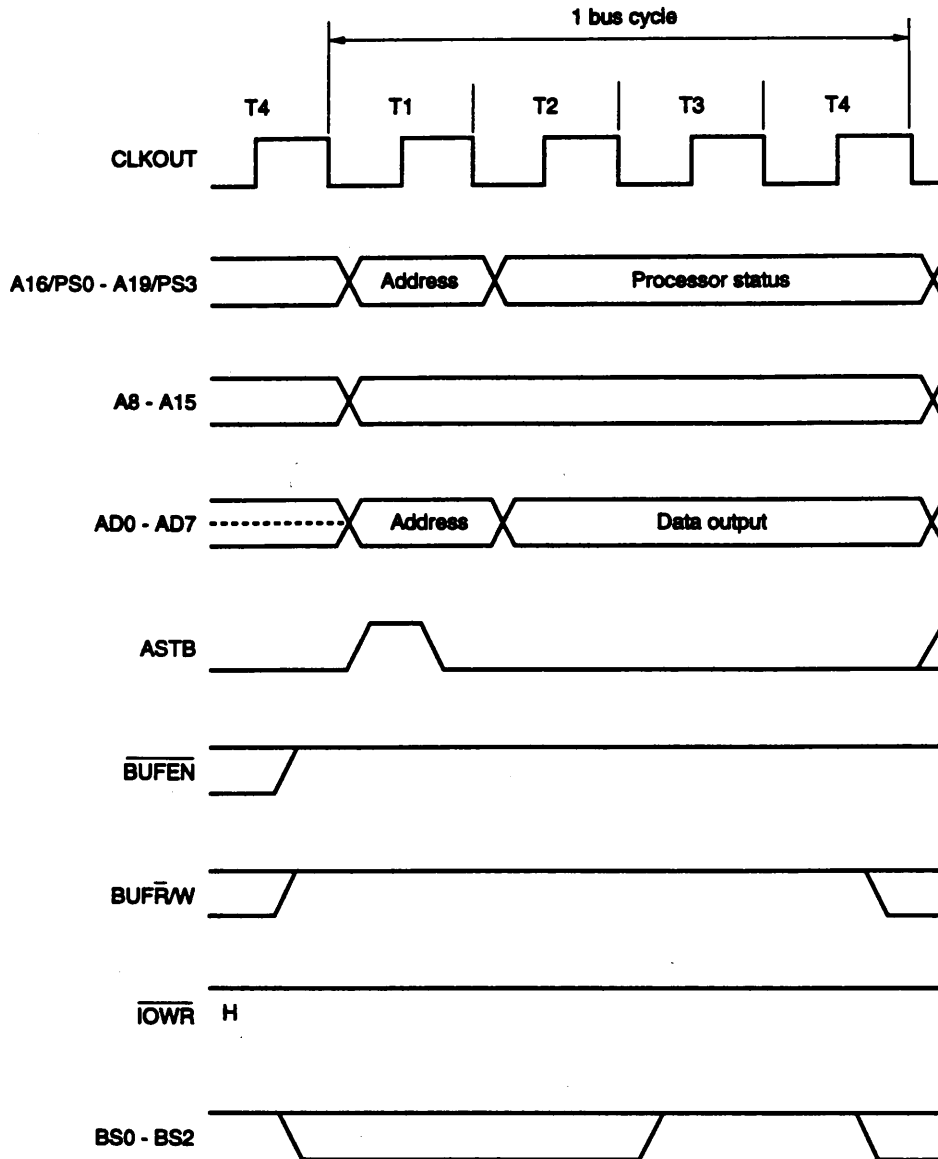
Remarks 1. Broken lines indicate high impedance.

2. Circles indicate the sampling timing when the wait cycle is set to zero in WCU. Refer to section 6.1.3 Relationship between WCU and READY pin.

Figure 4-17. Internal I/O Read Timing (V40HL)

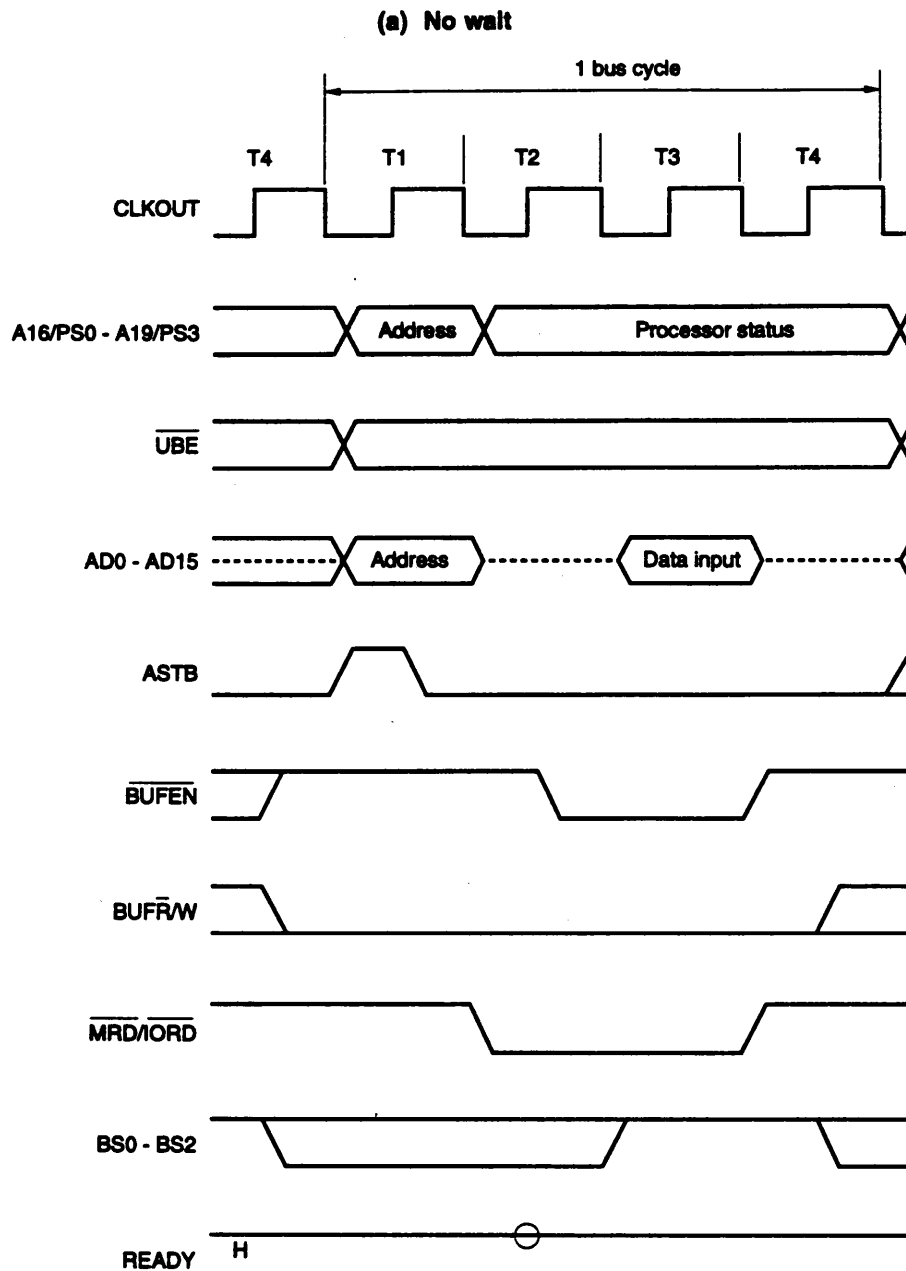


Remark Broken lines indicate high impedance.

Figure 4-18. Internal I/O Write Timing (V40HL)

Remark Broken line indicates high impedance.

Figure 4-19. Memory and External I/O Read Timing (V50HL) (1 of 2)

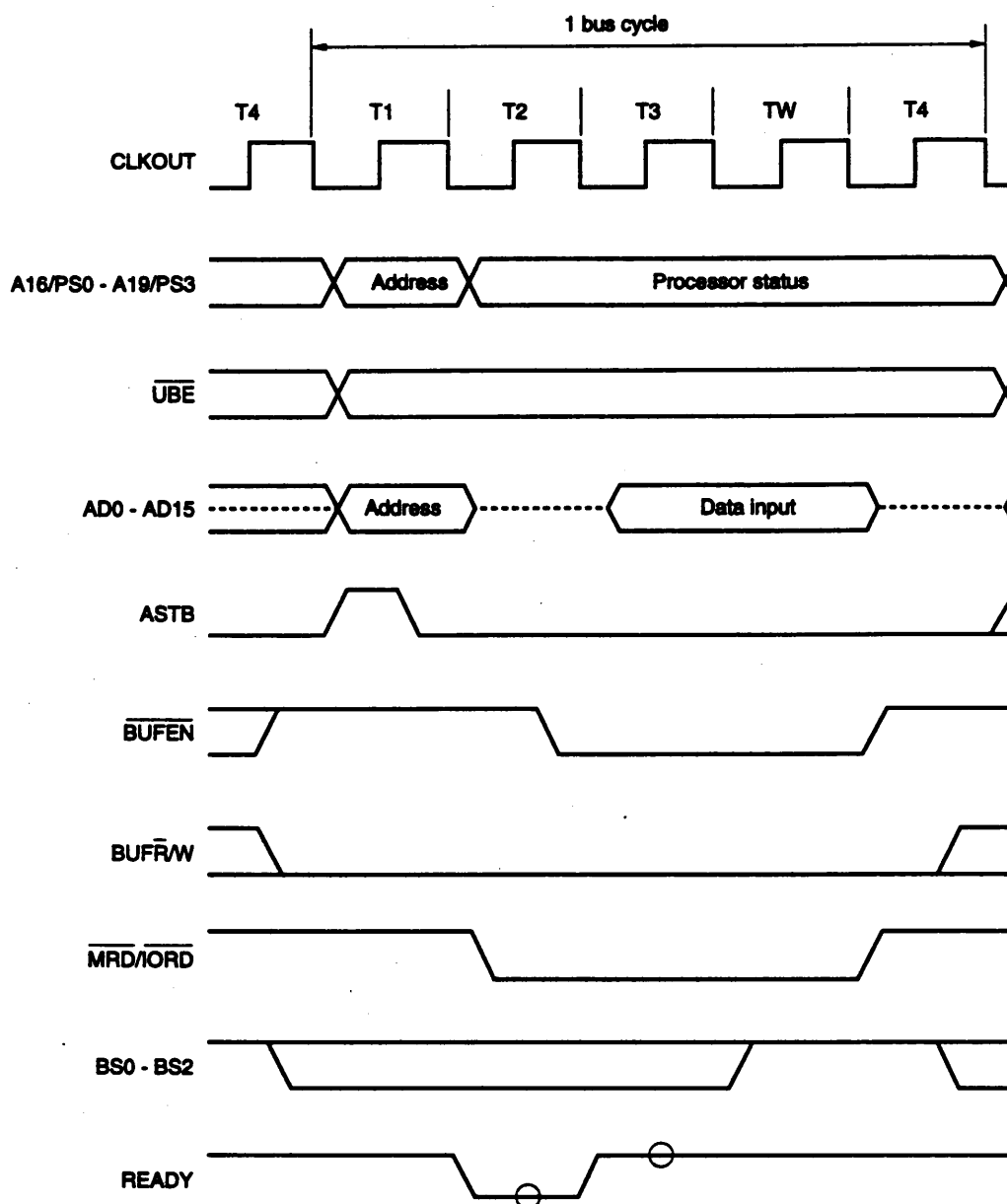


Remarks 1. Broken lines indicate high impedance.

2. A circle indicates the timing of the sampling.

Figure 4-19. Memory and External I/O Read Timing (V50HL) (2 of 2)

(b) One wait

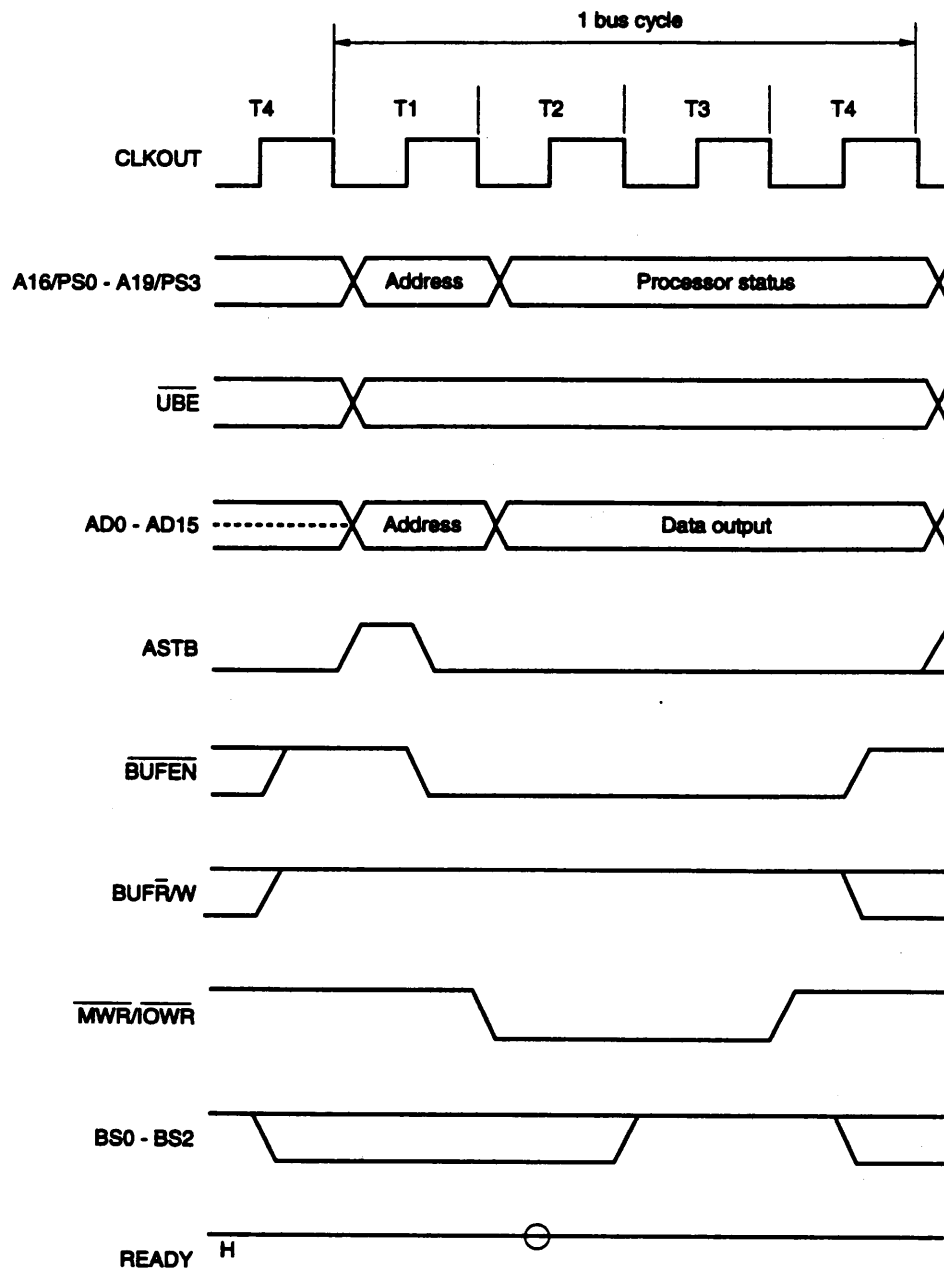


Remarks 1. Broken lines indicate high impedance.

2. Circles indicate the sampling timing when the wait cycle is set to zero in WCU. Refer to section 6.1.3 Relationship between WCU and READY pin.

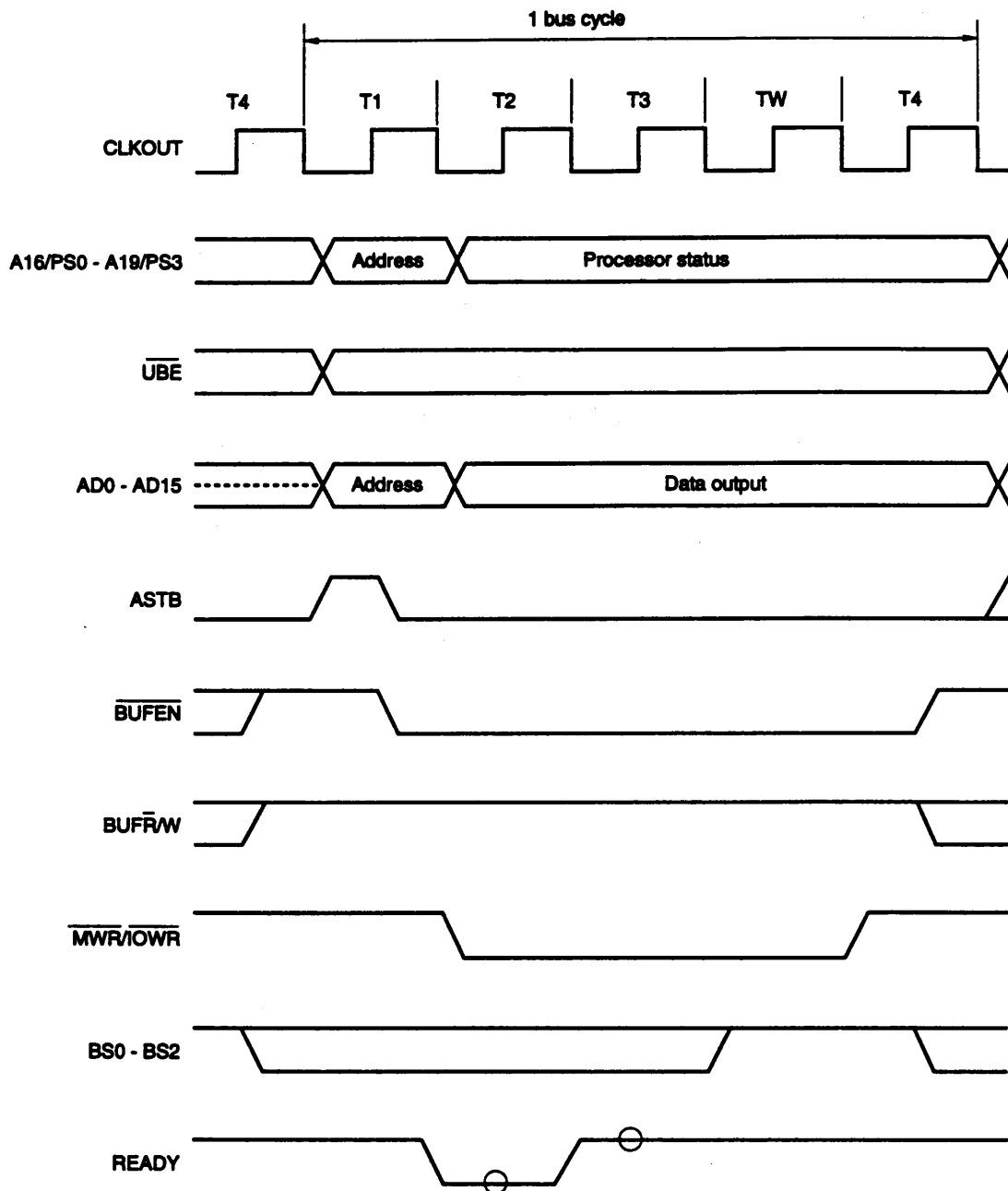
Figure 4-20. Memory and External I/O Write Timing (V50HL) (1 of 2)

(a) No wait



Remarks 1. Broken line indicates high impedance.

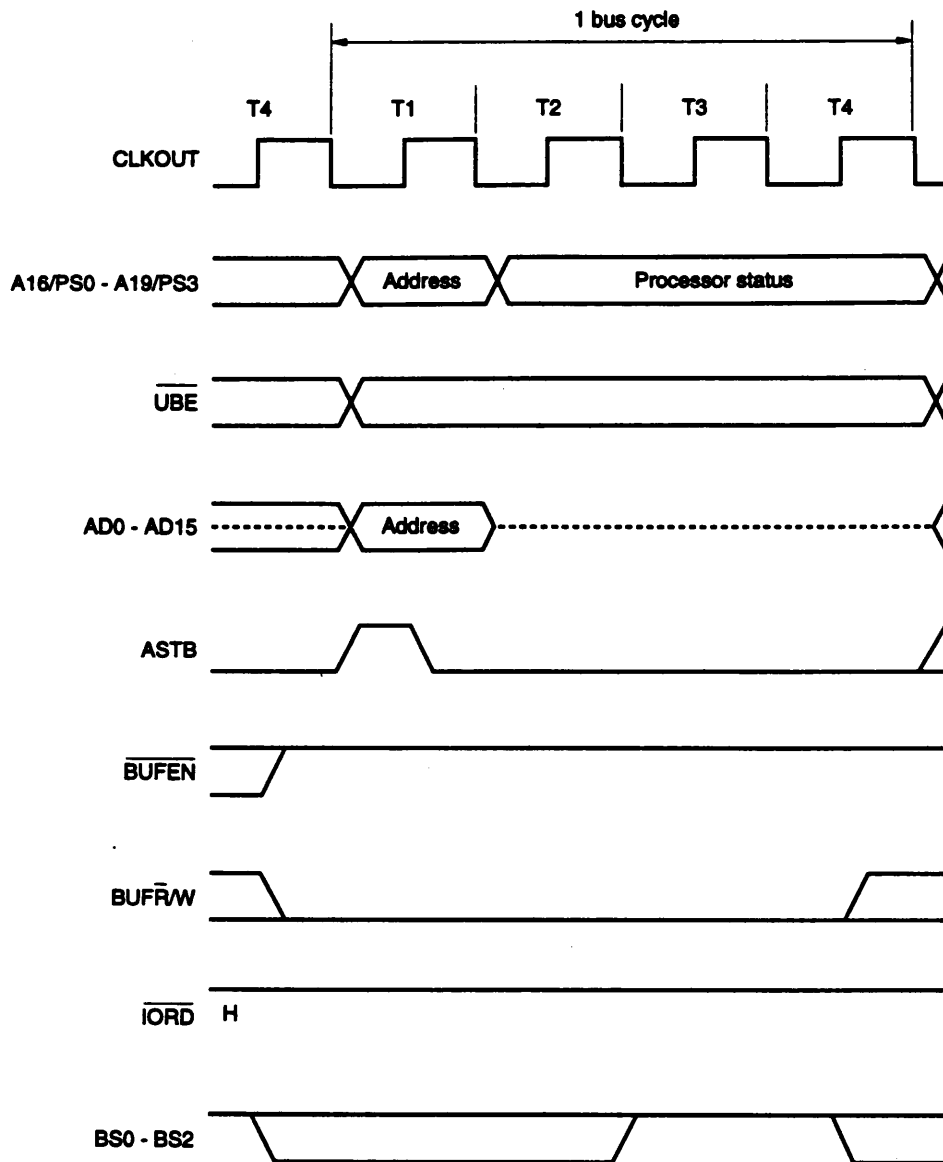
2. A circle indicates the timing of the sampling.

Figure 4-20. Memory and External I/O Write Timing (V50HL) (2 of 2)**(b) One wait**

Remarks 1. Broken line indicates high impedance.

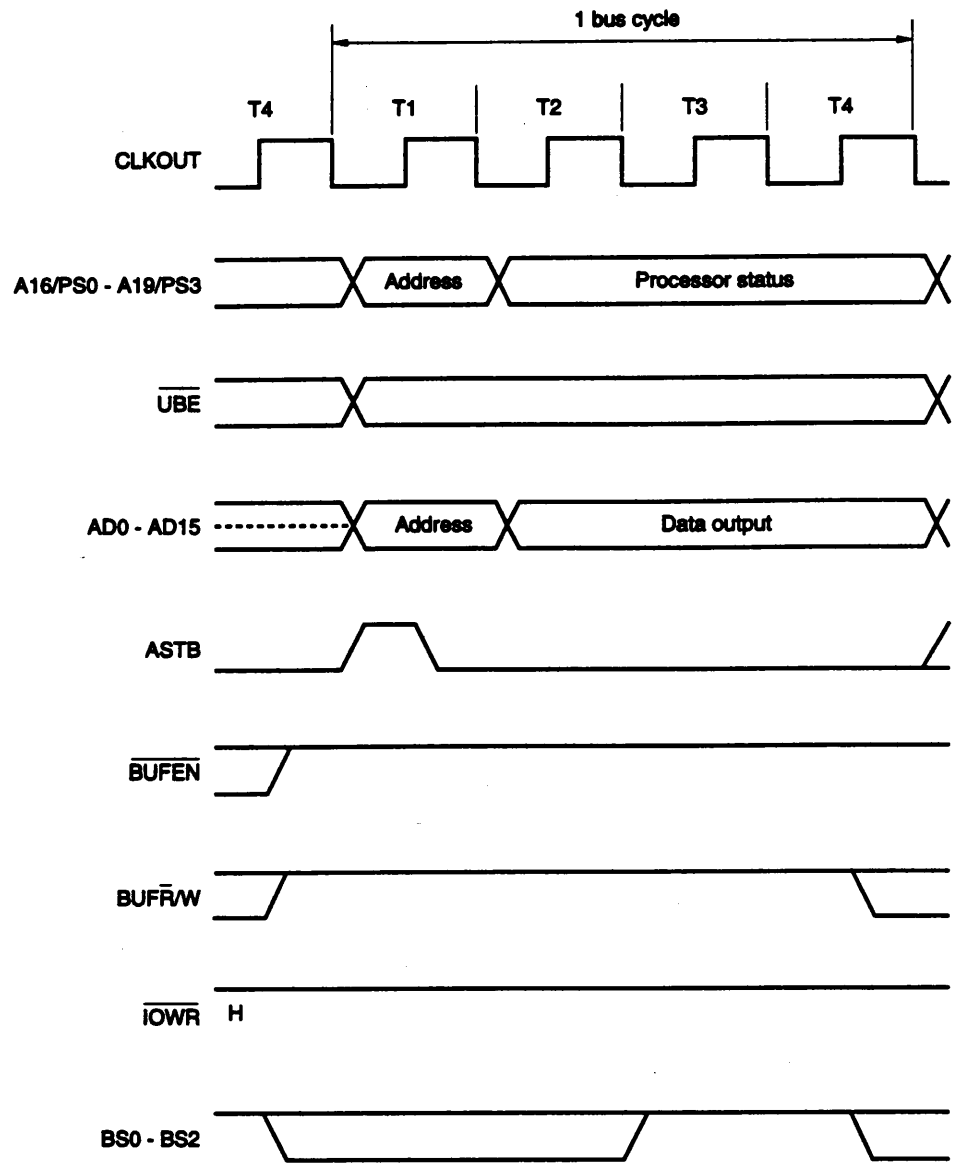
2. Circles indicate the sampling timing when the wait cycle is set to zero in WCU. Refer to section 6.1.3 Relationship between WCU and READY pin.

Figure 4-21. Internal I/O Read Timing (V50HL)



Remark Broken lines indicate high impedance.

Figure 4-22. Internal I/O Write Timing (V50HL)



Remark Broken line indicates high impedance.

4.2 Logical Addresses and Physical Addresses

The V40HL and V50HL have been provided with a 20-bit address bus in which the low-order 8/16 bits are also used as a data bus. They are able to access a 1-Mbyte memory area.

However, it is virtually impossible for a programmer to create a program while managing a 1 Mbyte address (called a physical address) that corresponds directly to the hardware.

For this reason, the segment approach is used with the V40HL and V50HL so that when creating a program the segments are handled as a group of small-unit logical address areas (of 64 Kbytes) that do not depend directly on the physical address.

There are four types of segments: a program segment, a stack segment, data segment 0 and data segment 1. The address of each is specified by using the offset from the first address of the logical segment, which determines the 16-bit segment register for each (PS, SS, DS0, DS1).

Offset \ Segment Register	Default	Override
PFP	PS	None
SP	SS	None
Effective address (BP base)	SS	PS, DS0, DS1
Effective address (non-BP base)	DS0	PS, SS, DS1
Instruction group A in IX	DS0	PS, SS, DS1
Instruction group B in IY	DS1	None

Instruction group A: Primitive block transfer instruction, primitive output instruction, BCD string instruction, EXT instruction

Instruction group B: Primitive block transfer instruction, primitive input instruction, BCD string instruction, INS instruction

The operation of each segment will be described next.

(1) Program Segment

The first address of this program is determined by the program segment register (PS). The offset from the first address is specified by the prefetch pointer (PFP).

Instruction codes and table data are located in this segment.

By using the segment override prefix (PS:), the program segment can be used as a general-purpose variable area and as source data when executing instruction group A (primitive block transfer instruction, primitive output instruction, BCD string instruction, EXT instruction).

(2) Stack Segment

The first address of this segment is determined by the stack segment register (SS). The offset from the first address is specified by the effective address when it uses the stack pointer (SP) plus uses the base pointer (BP) as the base address.

When processing an interrupt or subroutine, it can be used as the save area of a return address (content of PS and PC) or the content of a program status word (PSW) or general-purpose register, used as an area for passing parameters or as an area for local variables.

By using the segment override prefix (SS:), the stack segment can be used as a general variables area or a source data area for executing instruction group A (primitive block transfer instruction, primitive output instruction, BCD string instruction, EXT instruction).

(3) Data Segment 0

The first address of this segment is determined by the data segment 0 register (DS0). The offset from the first address is specified by the effective address when BP is not used as the base address.

This segment is used as the storage area for general-purpose variables.

When executing instruction group A (primitive block transfer instruction, primitive output instruction, BCD string instruction, EXT instruction), it is used as the source data area. However, in this case, the content of the index register (IX) becomes the offset.

In the case of an effective address when BP is used as the base address, the stack segment can be used as the default. However, by using the segment override prefix (DS0:), data segment 0 can be used.

(4) Data Segment 1

The first address of this segment is determined by the data segment 1 register (DS1).

It can be used as the destination data area when instruction group B (primitive block transfer instruction, primitive input instruction, BCD string instruction, INS instruction) is executed. In this case, the content of the index register (IY) becomes the offset.

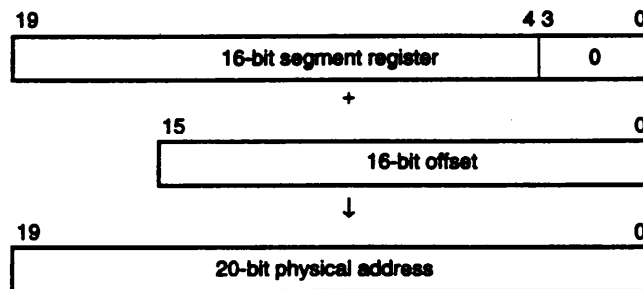
By using the segment override prefix (DS1:), data segment 1 can be used as the general-purpose variables area or the source data area when executing instruction group A.

The programmer can create a program by simply viewing the content of the segment register to be used (the one for default or one with an override specification) and the offset value from the values in the segment register. (That is, if the segment register value is address 0, the address within that segment will become the logical address without any other processing.)

A number of programs created as segment groups specified by such a logical address will be compiled or assembled individually to become a number of object modules. Each individual object module contains the segment name, size, content demarcation, and control data, and becomes the specification to the linker.

A multiple of object modules are placed in the linker, a segment base that corresponds to the physical address is specified, which puts them in a state that can actually be loaded into memory.

The relationship between the segment register, offset and physical address is as shown below.



In other words, the physical address is a value that adds the offset to the segment register, which has a content that is 16 times larger. Here, the content and offset of the segment register are handled as unsigned data.

If a program has not executed an instruction for changing the segment base within it (a branch or variable reference is within the segment), the addresses within the program can all be determined by the offset from the segment register values. Therefore, the content of the memory can be loaded to a desired area simply by matching the content of the segment register to the physical address of the memory to which one desires to load the program.

By making use of this method, to execute programs in external files, such as on floppy disks, the operating system can control the memory area and segment register, load such programs in a desired empty memory area, and execute them.

If a program is either spread across other files or gathered in one place and is remapped in this manner to a memory area that happens to be empty each time it is executed, such remapping is called dynamic relocation.

4.3 Address Generation

4.3.1 Instruction address

An instruction address is an address where an instruction code is read. Normally, it increments automatically each time it reads an instruction code. However, in the case of an instruction that controls the sequence of instruction execution, such as a jump instruction or a subroutine call instruction, the address of the instruction to shift to is specified by an operand.

(1) Direct Addressing

The four-byte data in the instruction code becomes the instruction address, which is loaded to both the PS and the PC registers.

This mode is used by the following instructions:

```
CALL    far_proc
BR      far_label
```

(2) PC Relative Addressing

The one-byte or two-byte data within an instruction code becomes the displacement from the first address of the next instruction (PC value) and is added to PC.

This mode is used by the following instructions:

```
CALL      near_proc
BR        near_label
BR        short_label
Bcondition short_label ; Example: BZ      short_label
                        BNC      short_label
```

(3) Register Indirect Addressing

Register indirect register addressing is specified by the register specification within the instruction code. The content of an optional 16-bit register becomes the instruction address and is loaded to PC.

This mode is used by the following instructions:

```
CALL    regptr16      ; Example: CALL    AW
BR      regptr16      ; Example: BR      IX
```

(4) Memory Indirect Addressing

Memory indirect addressing can be specified using the memory addressing (section 4.3.2 Data addressing) indicated by the addressing mode specification in the instruction code. Either two-byte or four-byte data within the memory becomes the instruction address, and this is loaded directly either to PC or to PS and PC.

This mode is used by the following instructions:

CALL memptr16	; Example: CALL word_var [BW]
CALL memptr32	; Example: CALL dword_var [BW+IX]
BR memptr16	; Example: BR word_var [BR+2]
BR memptr32	; Example: BR dword_var [BP+IY]

4.3.2 Data addressing

A data address is an address for reading or writing the operands of each instruction. Normally, the address is a concept that is relative to memory or I/O. However, the operand address in this case contains data in registers, immediate data and I/O data.

(1) Non-Memory Addressing

Non-memory addressing specifies data within registers, immediate data and I/O data.

(a) Register Addressing

Register addressing specifies a register that reads or writes operand data that the register specification field specifies within the instruction code. Register addressing is expressed in the following notation:

General Notation	Expressible Registers
reg, reg'	AW, BW, CW, DW, SP, BP, IX, IY, AL, AH, BL, BH, CL, CH, DL, DH
reg8, reg8'	AL, AH, BL, BH, CL, CH, DL, DH
reg16, reg16'	AW, BW, CW, DW, SP, BP, IX, IY
sreg	PS, SS, DS0, DS1
acc	AW, AL

Example:

In the case of reg16 : MOV AW, IX ; AW ← IX
 In the case of reg8 : ADD AL, CH ; AL ← AL + CH

(b) Immediate Addressing

One-byte or two-byte data within an instruction becomes read-only operand data. Immediate addressing cannot be used for the instruction destination operand. Immediate data is expressed in the following notation:

General Notation	Expressible Values
imm8	0 to FFH (0 to 255 or -128 to 127)
imm16	0 to FFFFH (0 to 65535 or -32768 to 32767)
imm	0 to FFFFH (0 to 65535 or -32768 to 32767)
pop_value	0 to FFFFH (0 to 65535) Normally even numbers

Examples:

In the case of imm16 : MOV AW, 216 ; AW ← 216

In the case of imm8 : SHL, AL, 5 ; Shift AL five bits to the left.

In the case of pop_value: RET 16 ; Deletes the unnecessary 16 bytes from the stack.

(c) I/O Addressing

I/O addressing specifies the data within the 64-Kbyte I/O area. The two methods of specification are as shown below. They are used with input and output instructions.

<1> imm8

The 8-bit data within the instruction code indicate the I/O address. With this method of specification, the specification is limited to the low-order 256 byte area within the 64-Kbyte I/O area. The following two instructions use this method of specification:

```
IN      acc, imm8
OUT     imm8, acc
```

<2> DW

The content of the 16-bit DW register indicates the I/O address. With this method of specification, all of the 64-Kbyte I/O area can be specified. The following four instructions use this method of specification:

```
IN      acc, DW
INM     dst_block, DW
OUT     DW, acc
OUTM    DW, src_block
```

(2) Memory Addressing

Memory addressing specifies certain operand data within the memory. This memory addressing is classified further into a number of modes by means of the 5-bit memory addressing specification field placed after the operation code. All modes of memory addressing specify a 16-bit offset address from the segment base, which is specified by the default or by the segment override. Memory addressing is expressed in the following notation:

Notation	Data Length
dmem ^{Note}	8/16-bit data
mem	8/16-bit data
mem8	8-bit data
mem16	16-bit data

Note Notation for an instruction with no memory addressing specification field.

(a) Direct Addressing

In direct addressing, the two-byte data in the instruction code specifies the memory address that will be subject to operand data reading and writing.

Example:

MOV byte_var, 216 ; bytemem (offset (byte_var)) ← 216

(b) Register Indirect Addressing

In register indirect addressing, a 16-bit register (BX or IX or IY) indicated by the memory addressing specification field in the instruction code becomes the memory address subjected to the reading and writing of the operand data.

Example:

MOV word ptr [BX], 10 ; wordmem (BX) ← 10
ADD AL, byte ptr [IX] ; AL ← AL + bytemem (IX)

(c) Based Addressing

In based addressing, a value that sign-extends the displacement indicated by the one-byte or two-byte data in the instruction code is added to the 16-bit base register (BX or BP) specified in the memory addressing specification field in the instruction code, and this becomes the memory address subjected to the reading and writing of the operand data.

When BP is selected as the base register, the default segment register will become SS. Therefore, when there is a procedural call, it can be used to access data from within the procedure, data that is pushed onto the stack as an argument.

Example:

```
MOV  word_var [BW + 2], AW    ; wordmem (offset (word_var) + BW + 2) ← AW
SUB  AW, [BP + 6]             ; AW ← AW - wordmem (BP + 6)
```

(d) Indexed Addressing

In indexed addressing, a value that sign-extends the displacement indicated by the one-byte or two-byte data in the instruction code is added to the 16-bit index register (IX or IY) specified in the memory addressing specification field in the instruction code, and this becomes the memory address subjected to the reading and writing of the operand data.

Example:

```
MOV  word_var [IY + 2], 0      ; wordmem (offset (word_var) + IY + 2) ← 0
SUB  AW, [IX + 6]             ; AW ← AW - wordmem (IX + 6)
```

(e) Based Indexed Addressing

In based indexed addressing, a value that adds the 16-bit index register (IX or IY) to the value that sign-extends the displacement indicated by the one-byte or two-byte data in the instruction code is added to the 16-bit base register (BW or BP) specified in the memory addressing specification field in the instruction code, and this becomes the memory address subjected to the reading and writing of the operand data. That is, the type of addressing that takes place is a combination of based addressing and indexed addressing. This type of addressing can be used to access data that has a structure, such as that of a two-dimensional array.

Example:

```
MOV  word_var [BW + 6] [IY + 2], 0 ; wordmem (offset (word_var) + BW + 6 + IY + 2) ← 0
SUB  AW, [BP + 6 + IX]             ; AW ← AW - wordmem (BP + IX + 6)
```

(3) Bit Addressing

Bit addressing specifies the one-bit data in an 8/16-bit register or in an 8/16-bit memory. In bit addressing, data is specified using two operands, one operand for the register or memory, another for the bit offset, which is an optional one bit in the register/memory operand. The desired type of addressing between register addressing and memory addressing can be used for the operand that specifies the register or memory. The following three types of addressing can be used for the bit offset.

Notation	Specified 1 bit
imm3	1-bit data within the byte register/memory.
imm4	1-bit data within the word register/memory.
CL	One-bit data within the byte/word register/memory

In the case of the byte register/memory, the bit offset is the low-order 3 bits. In the case of the word register/memory, only the low-order 4 bits are valid. Thus, the ranges are from zero to 7 and zero to 15, respectively.

Example:

```

TEST1  word ptr [BW + 2], 5    ; Checks bit 5 of wordmem (BW + 2).
CLR1   byte var, CL           ; Specified by CL of bytemem (offset (byte_var))
                                   One bit is set to zero.
  
```

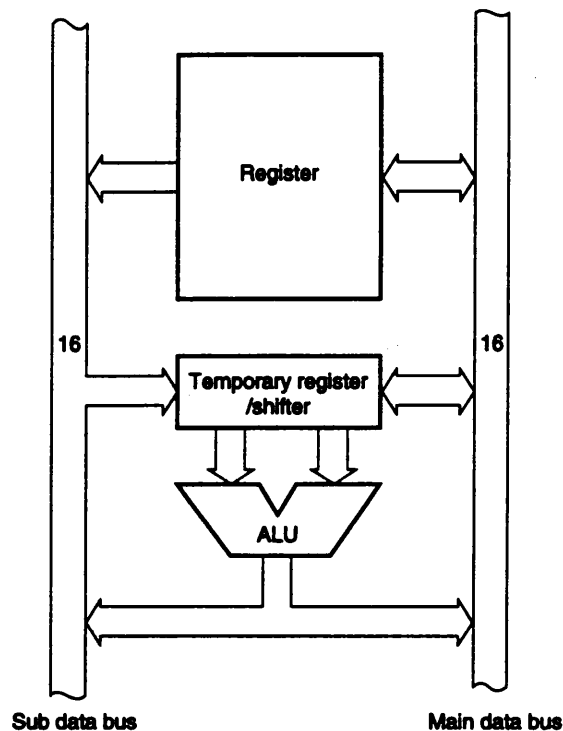
4.4 Speeding Up Instruction Execution

The CPUs of the V40HL and V50HL have been equipped with the hardware functions shown below to speed up the instruction execution times. (For details, see section 5.1 CPU.)

- Dual data buses within the EXU
- Effective address generation circuit
- Temporary register/shifter (TA/TB)
- Loop counter (LC)
- Program counter (PC) and prefetch pointer (PFP)

4.4.1 Dual data buses

In order to decrease the number of processing steps required to execute an instruction, dual data buses, a main data bus (16 bits) and a sub-data bus (16 bits) are used. This method has brought about a 30 percent reduction in processing time over the single data bus method for addition/subtraction, logical operations and comparison instructions.



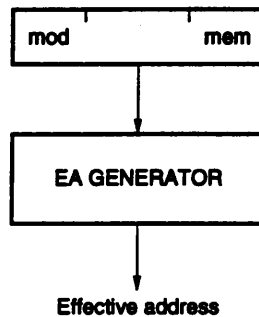
Example: ADD AW, BW: $AW \leftarrow AW + BW$

	Single bus	Dual bus
Steps 1	$ALU \leftarrow AW$	$ALU \leftarrow AW, BW$
2	$ALU \leftarrow BW$	$AW \leftarrow ALU$
3	$AW \leftarrow ALU$	

4.4.2 Effective address generation circuit (EAG)

This is a circuit that calculates at high speed the effective address required when accessing memory.

With the microprogram method, from 5 to 12 clocks are required to calculate the effective address. However, by using this dedicated hardware, calculation is several times faster in all address modes.



4.4.3. Temporary registers/shifters A/B (TA/TB)

Temporary registers/shifters (TA, TB) have been provided and are used for multiply/divide instructions and shift/rotate instructions.

The use of this circuit has especially allowed an increase in multiply/divide instruction processing speed, a four-fold increase over the microprogram method.

TA + TB: 32-bit temporary registers/shifters for multiply/divide instructions

TB: 16-bit temporary register/shifter for shift/rotate instructions

4.4.4 Loop counter (LC)

The loop counter counts the number of primitive block transfer/input/output instruction loops that are controlled by the repeat prefix instruction and the number of shifts of the multiple bit shift/rotate instruction.

For example, in the case of a multiple bit rotation of a register, the required number of clocks is as indicated below. Speed has doubled over that of the microprogram method.

RORC AW, CL; CL = 5

Microprogram method
 $8 + 4 \times 5 = 28$ clocks

LC method
 $7 + 5 = 12$ clocks

4.4.5 Program counter (PC) and prefetch pointer (PFP)

By providing in hardware the prefetch pointer (PFP), which addresses the program memory when there is a prefetch, and the program counter (PC), which addresses the program memory that will be executed now, the instruction execution times for branch, call, return and break instructions have been shortened by several clocks over having only one prefetch pointer.

[MEMO]

CHAPTER 5 INTERNAL BLOCK FUNCTIONS

5.1 CPU

5.1.1 Features

The V40HL and V50HL CPUs have the following features:

- (1) V20HL and V30HL equivalents.
20-MHz maximum operating frequency.
- (2) 1-Mbyte memory area and 64-Kbyte I/O area.
- (3) Powerful instruction set.
Fully-compatible with the V40 and V50.
- (4) Abundant memory addressing modes.
- (5) 14 x 16-bit register set.
- (6) Bit-field operation instructions.
Data is transferred between 1- to 16-bit memory bit field and the accumulator.
- (7) Packed BCD operation instructions.
Addition, subtraction and comparison of BCD strings that are 1 to 255 digits in length.
- (8) Bit operation instructions.
Setting, clearing, inverting and testing of an optional bit of an 8/16-bit register/memory.
- (9) High-speed block transfer instructions between memory
- (10) High-speed multiply/divide instructions (in the dedicated hardware)
- (11) High-speed effective address calculation (in the dedicated hardware)
- (12) Abundant interrupt processing functions:
 - External interrupts: Non-maskable interrupts (NMI) and maskable interrupts (INT)
 - Software interrupts: BRK (unconditional)
BRKV (when V = 1)
BRKEM (emulation)
CHKIND (array index check)
CALLN (native routine call)
- (13) IEEE-796 bus-compatible interface
- (14) Two types of operating modes.
Mode changes due to a mode switch instruction (BRKEM, RETEM, etc.) and due to an external interrupt.
 - Native mode: Executes the V40HL and V50HL instruction set.
 - Emulation mode: Executes the μ PD8080AF instruction set.

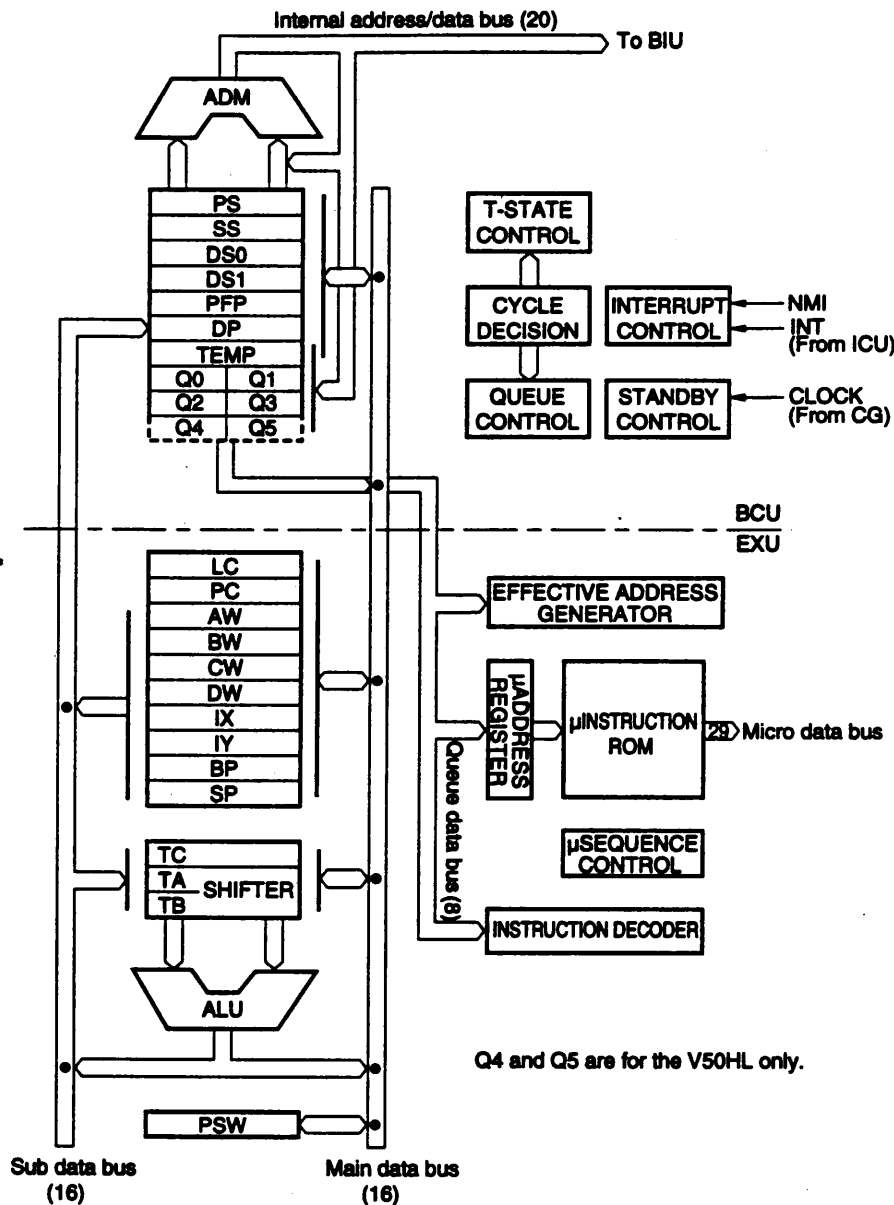
5.1.2 Internal organization

The CPU consists of two independent processing units, the bus control unit (BCU) and the execution unit (EXU). Each has the functions indicated below.

BCU: Prefetching of instructions that use the instruction queue (4 bytes for the V40HL, 6 bytes for the V50HL).

EXU: Data processing (microprogram execution)

Figure 5-1. CPU Internal Block Diagram



(1) Prefetch Pointer (PFP)

The prefetch pointer (PFP) is a 16-bit binary counter that maintains the program memory address offset data that BCU tries to prefetch for the instruction queue. PFP increments each time BCU prefetches an instruction byte from the program memory. Also, a new location is loaded when a branch, call, return, or break instruction is executed, and the content of PFP at this time is the same as that of the program counter (PC). PFP is always used together with a program segment (PS).

(2) Prefetch Queue (Q0-Q3/Q0-Q5)

The V40HL and V50HL CPUs have a 4/6-byte instruction queue (FIFO), allowing them to store a maximum of 4/6 bytes of instruction code that BCU prefetches. The instruction code stored in the queue is fetched by EXU and executed.

When a branch, call, return or break instruction is executed or when an external interrupt is processed, the content of the queue is cleared and the instruction of a new location is prefetched.

Normally, when there is a byte or more of space in a queue, the V40HL will prefetch. Normally, when there is a word (2 bytes) or more of space in a queue, the V50HL will prefetch.

If the average execution time of a multiple of instructions that execute one after the other exceeds the number of clocks required to prefetch the instruction codes of each individual instruction by a certain amount, when EXU completes the execution of one instruction, another instruction code that EXU can execute will be readied in the queue next so that the fetch time from the external memory is eliminated from the instruction execution time. The result is an increase in processing speed over a CPU that fetches and executes for each instruction.

The more there are instructions that allow the queue to be cleared, as when executing the branch instruction described above, the more there is a decline in the effectiveness of the queue. It also declines when there is a series of instructions with short execution times.

(3) Data Pointer (DP)

The data pointer (DP) is a 16-bit register that points to the read/write addresses of variables. The effective address that is generated by the effective address generator (EAG) and the content of the register containing the memory address offset are transferred to DP.

(4) Temporary Communications Register (TEMP)

This is a 16-bit temporary register between the external data bus and EXU. TEMP is able to independently read or write the high-order bytes and the low-order bytes, which allows byte access.

Basically, EXU terminates a write operation by transferring data to TEMP and terminates a read operation by confirming that data will be transferred to TEMP from an external data bus.

(5) Segment Registers (PS, SS, DS0, DS1)

The CPU divides the memory area into logical segments of 64 Kbyte units and controls up to four segments simultaneously. A start address is specified for each segment by the four registers indicated below.

PS (program segment)
 SS (stack segment)
 DS0 (data segment 0)
 DS1 (data segment 1)

The offset within a logical segment is such that a special register or an effective address specifies it. The offset and segment register combinations are shown in the table below. When the offset is PFP or the stack pointer (SP), or when it is the index register (IY) when there is a primitive block transfer instruction, a primitive input instruction, a BCD string instruction or an INS instruction, the segment registers that can be used in a combination are fixed at PS, SS and DS1, respectively. The other segment register cannot be used. For other offsets, in addition to the default segment register, any desired segment register can be specified by using the segment override prefix.

Table 5-1. Offset and Segment Register Combinations

Offset	Segment registers	
	Default	Override
PFP	PS	None
SP	SS	None
Effective address (BP base)	SS	PS, DS0, DS1
Effective address (non-BP base)	DS0	PS, SS, DS1
IX for instruction group A	DS0	PS, SS, DS1
IY for instruction group B	DS1	None

Instruction group A: Primitive block transfer instruction, primitive output instruction, BCD string instruction, EXT instruction

Instruction group B: Primitive block transfer instruction, primitive input instruction, BCD string instruction, INS instruction

(6) Address Modification Circuit (ADM)

The address modification circuit (ADM) generates the physical address (adds the segment register, PFP or DP) and increments the PFP.

(7) General-Purpose Registers (AW, BW, CW, DW)

There are four 16-bit registers that are general-purpose registers. They, of course, can be accessed as 16-bit registers. However, by dividing them up into the high-order 8 bit and low-order 8 bits, they also can be accessed as 8-bit registers (AH, AL, BH, BL, CH, CL, DH, DL). Therefore, these registers have a wide range of uses as 8-bit or 16-bit registers, such as for transfer instructions, calculation instructions, and logical operations.

In addition, as indicated below, each register can be used as a default register for special instruction processing.

AW : Word multiply/divide calculations, word input/output, data conversion

AL : Byte multiply/divide, byte input/output, translation, BCD rotation, data conversion

AH : Byte multiply/divide

BW : Translation

CW : Loop control branch, repeat prefix

CL : Shift instruction, rotation instruction, BCD operations

DW : Word multiply/divide calculation, indirect addressing of input/output

(8) Pointers (SP and BP)

The pointers consist of two 16-bit registers (stack pointer (SP) and base pointer (BP)). Each register can be referenced during instruction execution, and can be used as index registers for generating effective addresses when referencing memory data. Also, for special processing, they have the following special functions:

BP : The base register for generating the effective address when the memory operand is referenced.

SP : Stack pointer

(9) Index Registers (IX and IY)

The index registers are composed of two 16-bit registers (IX and IY). Each register can be referenced during instruction execution and can be used as index registers for generating effective addresses when referencing memory data. Also, for special processing, they have the following special functions:

IX : A source operand address register used for block data operation instructions.

A base register for variable length bit field operation instructions.

Source operand address register for a BCD string operation instructions.

IY : A destination operand address register used for block data operation instructions.

A base register for variable length bit field operation instructions.

A destination operand address register for a BCD string operation.

(10) Arithmetic Logic Unit (ALU)

The arithmetic logic unit (ALU) consists of a full adder and a logical operations circuit. It conducts arithmetic operations (addition, subtraction, multiplication and division, and increment, decrement and complement operations) and logical operations (test, AND, OR, XOR, and bit-unit test, set, clear and invert).

(11) Temporary Register/Shifter A/B (TA/TB)

The temporary registers/shifters (TA/TB) are 16-bit registers/shifters that are used for multiply/divide and shift/rotate instructions (including BCD rotate). When multiply/divide instructions are executed, TA and TB function together as a 32-bit temporary register/shifter. When shift/rotate instructions are executed, only TB functions as a 16-bit temporary register. Both the high-order bytes and low-order byte of TA and TB can be read from and written to independently between the internal buses. TA and TB become the input of the ALU.

(12) Temporary Register C (TC)

Temporary register C (TC) is a 16-bit register that is used for multiply/divide calculations and other internal processing. TC also becomes the input of the ALU.

(13) Program Status Word (PSW)

The program status word is composed of six types of status flags and four types of control flags. These are listed below.

Status flags:

- V (overflow flag)
- S (Sign flag)
- Z (Zero flag)
- AC (Auxiliary carrier flag)
- P (Parity flag)
- CY (Carry flag)

Control flags:

- MD (Mode flag)
- DIR (Direction flag)
- IE (Interrupt enable flag)
- BRK (Break flag)

The status flags are automatically set (1) and reset (zero) according to the results (data values) of a variety of instruction executions.

The carry flag can be directly set, reset and inverted by instructions.

The control flag is set and reset by instructions to control the operation of the CPU.

The interrupt enable and break flags are reset when interrupt processing begins.

The mode flag is set but all of the other flags are reset (0) by a RESET input. When PSW is processed in byte or word units, its image is as shown below. Byte unit processing only takes place for the low-order 8 bits (except for the overflow flag, this includes the status flags).

Figure 5-2. PSW Bit Fields

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M	1	1	1	V	D	I	B	S	Z	0	A	0	P	1	C
D					R	E	R	K			C				Y

The low-order 8 bits of PSW can be stored and restored in AH by means of a MOV instruction.

All bits of PSW will be saved to the stack when there is an external or software interrupt or when a call instruction is executed. They are restored^{Note} with a return instruction (RET1 and RETEM).

In addition, PSW can be independently pushed onto the stack with a PUSH PSW instruction and popped off the stack^{Note} with a POP PSW instruction.

Note The mode flag has a write enabled state and a write disabled state. Even if a RETI or POP PSW instruction is executed while the mode flag is in the disabled state, the mode flag will not restore, but it will maintain its prior value. The mode flag becomes write disabled by means of a reset operation and a RETEM instruction, but enabled with a BRKEM instruction.

(a) Carry Flag (CY)

(I) Binary Add and Subtract

In the case of a byte operation, when there is a carry or borrow from operation result bit 7, the carry flag sets. When this does not happen, it resets.

In the case of a word operation, when there is a carry or borrow from operation result bit 15, this flag sets. When this does not happen, it resets.

This flag does not change due to an increment or decrement instruction.

(II) Logical Operations

The carry flag resets regardless of the operational result.

(iii) Binary Multiplication

If the result of an unsigned byte operation is that AH is zero, the carry flag resets. If the result is other than zero, it sets.

If the result of a signed byte operation is that AH becomes a sign-extension of AL, the flag resets. If other than this, it sets.

If the result of an unsigned word operation is that DW is zero, the carry flag resets. If the result is other than zero, it sets.

If the result of a signed word operation is that DW becomes a sign-extension of AX, the flag resets. If other than this, it sets.

In the case of an 8-bit immediate operation, the carry flag resets if the product is within 16 bits. If the product exceeds 16 bits, the flag sets.

(iv) Binary Division

Binary division is not defined.

(v) Shift/Rotate

In the case of a shift or rotate that includes the carry flag, if the bit that shifts to the carry flag is a 1, the flag sets. If the bit is a zero, the flag resets.

(b) Parity Flag (P)**(i) Binary Addition/Subtraction, Logical Operations and Shifting**

If the number of bits of the low-order 8 bits of the operation result that are set to 1 is an even number, the parity flag sets. If this number is an odd number, the parity flag resets.

If the result is all zeros, the parity flag sets.

(ii) Binary Multiplication and Division

Binary multiplication and division are not defined.

(c) Auxiliary Carry Flag (AC)**(i) Binary Addition and Subtraction**

In the case of byte operations, if there is a carry from the low-order 4 bits to the high-order 4 bits, or if there is a borrow from the high-order 4 bits to the low-order 4 bits, the auxiliary carry flag sets. In situations other than these, the flag resets.

In the case of word operations, the same operations are carried out for the low-order bytes as in byte operations.

(ii) Logical Operations, Binary Multiplication/Division and Shift/Rotate

Logical operations, binary multiplication/division and shift/rotate are not defined.

(d) Zero Flag (Z)**(I) Binary Addition/Subtraction, Logical Operations and Shift/Rotate**

In the case of a byte operation, if the 8 bits of the result are all zero, the zero flag sets. In the case of a word operation, if the 16-bit result is all zeros, the zero flag sets. For both byte and word operations, any other result will reset the zero flag.

(II) Binary Multiplication and Division

Binary multiplication and division are not defined.

(e) Sign Flag (S)**(I) Binary Addition/Subtraction, Logical Operations and Shift/Rotate**

In the case of a byte operation, if bit 7 of the result is 1, the sign flag sets. If bit 7 is zero, the sign flag resets. In the case of a word operation, if the bit 15 of the result is 1, the sign flag sets. If bit 15 is zero, the sign flag resets.

(II) Binary Multiplication and Division

Binary multiplication and division are not defined.

(f) Overflow Flag (V)**(I) Binary Addition and Subtraction**

In the case of a byte operation, if the carry from bits 7 and 6 are different, the overflow flag sets. If the carry from bits 7 and 6 are the same, the overflow flag resets.

In the case of a word operation, if the carry from bits 15 and 14 are different, the overflow flag sets. If the carry from bits 15 and 14 are the same, the overflow flag resets.

(II) Binary Multiplication

If the result of an unsigned byte operation is that AH is zero, the overflow flag resets. If the result is other than zero, it sets. If the result of a signed byte operation is that AH becomes a sign-extension of AL, the flag resets. If other than this, it sets.

If the result of an unsigned word operation is that DW is zero, the overflow flag resets. If the result is other than zero, it sets. If the result of a signed word operation is that DW becomes a sign-extension of AX, the flag resets. If other than this, it sets.

In the case of an 8-bit immediate operation, the overflow flag resets if the product is within 16 bits. If the product exceeds 16 bits, the flag sets.

(III) Binary Division

In the case of binary division, the overflow flag resets.

(IV) Logical Operations

In the case of logical operations, the overflow flag resets.

(v) Shift/Rotate

In the case of a shift/rotate one bit to the left, the operational result will be as indicated below.

If CY is the most significant bit, the overflow flag resets. If CY is not the most significant bit, the overflow flag sets.

In the case of a shift/rotate one bit to the right, the operational result will be as indicated below.

If the most significant bit is the next lower bit after the most significant bit, the overflow flag resets. If the most significant bit is not the next lower bit after the most significant bit, the overflow flag sets.

A multiple bit shift/rotate is not defined.

(g) Break Flag (BRK)

The break flag can be set by a memory manipulation instruction only when saved to a stack as a part of a PSW. After being set, the break flag becomes valid when it is restored to the PSW.

If the break flag is set, a software interrupt (interrupt vector 1) will be generated automatically when one instruction is executed. Then, one instruction at a time can be traced.

(h) Interrupt Enable Flag (IE)

The interrupt enable flag (IE) is set by the EI instruction and this puts the ICU interrupts in the enabled state. This flag is reset by the DI instruction, which puts the ICU interrupts in the disabled state.

(I) Direction Flag (DIR)

The direction flag (DIR) is set by the SET1 DIR instruction and reset by the CLR1 DIR instruction.

When the DIR flag is set, processing takes place from the high-order addresses to the low-order addresses for a block transfer/input/output system instruction. When the flag is reset, processing takes place from the low-order addresses to the high-order addresses.

(J) Mode Flag (MD)

The mode flag (MD) is set by $\overline{\text{RESET}}$ input, which puts the CPU in the native mode. The BRKEM instruction resets the mode flag and sets the CPU to the emulation mode.

In addition, the mode flag is set by the CALLN instruction and RETEM instruction, which places the CPU in the native mode.

$\overline{\text{RESET}}$ input and the RETEM instruction put the mode flag in the write disabled state. In this state, even if a RETI, POP PSW instruction is executed, the mode flag will not be restored. The BRKEM instruction puts the mode flag in the write enabled state.

(14) Loop Counter (LC)

The loop counter (LC) is a 16-bit register that counts the number of loops of a primitive block transfer/input-output instruction (MOV BK, OUTM, etc.) controlled by a repeat prefix instruction (REP, REPC, etc.) and counts the number of shifts of a multiple bit shift/rotate instruction.

(15) Program Counter (PC)

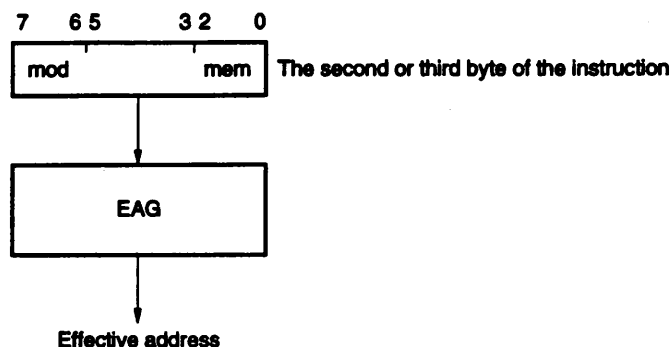
The program counter is a 16-bit binary counter that maintains the program memory address offset data that EXU is currently attempting to execute.

PC increments each time the microprogram fetches an instruction byte from the instruction queue. In addition, when a branch, call, return or break instruction is executed, a new location is loaded. At this time, the content of the program counter is the same as that of PFP.

(16) Effective Address Generator (EAG)

The effective address generator is a circuit that calculates the required effective address at high-speed when there is a memory access. It completes this calculation in two clocks for all of the addressing modes.

Figure 5-3. Effective Address Generation



The effective address generator takes the byte specified by the instruction operand (second or third byte) and generates the control signals for the ALU and the relevant register operation when a memory access is required. It then calculates the effective address and transfers it to DP. In addition, when required, it controls the request to BIU (external to CPU) for starting up the bus cycle.

(17) Instruction Decoder

The instruction decoder classifies the first byte of the instruction code to a group with special functions and maintains it while a macro instruction is being executed.

(18) Micro Address Register

The micro address register specifies the address of the next micro instruction ROM to be executed.

When a macro instruction has started executing, the first byte of the instruction stored in the queue is placed in this register as the start address, and the start address of the designated micro instruction sequence is specified.

(19) Micro Instruction ROM

The micro instruction ROM maintains 1024 words of the 29-bit wide micro instruction.

(20) Micro Instruction Sequence Circuit

The micro instruction sequence circuit manages the control of the micro address register, the micro ROM output and the synchronization of EXU and BCU.

5.2 Clock Generator (CG)

The clock generator generates a clock signal that is one half the frequency of the crystal resonators connected to the X1 and X2 pins.

It supplies a clock that is the same as CLKOUT to the CPU, DMAU, REFU and SCU as an internal clock. It also supplies a clock with one half the oscillating frequency to the baud rate counter and clocks with 1/4, 1/8, 1/16 and 1/32 the oscillating frequency (selected by TCKS in the system I/O area) to the timer.

In addition, it reduces the frequency of the internal clock by means of commands and is equipped with a variable instruction cycle time function that is able to reduce the power consumption. When the internal clock frequency divider rate changes due to the variable instruction cycle time function, the status of the CLKOUT pin changes at the same time (1/2-1/16 of the input clock).

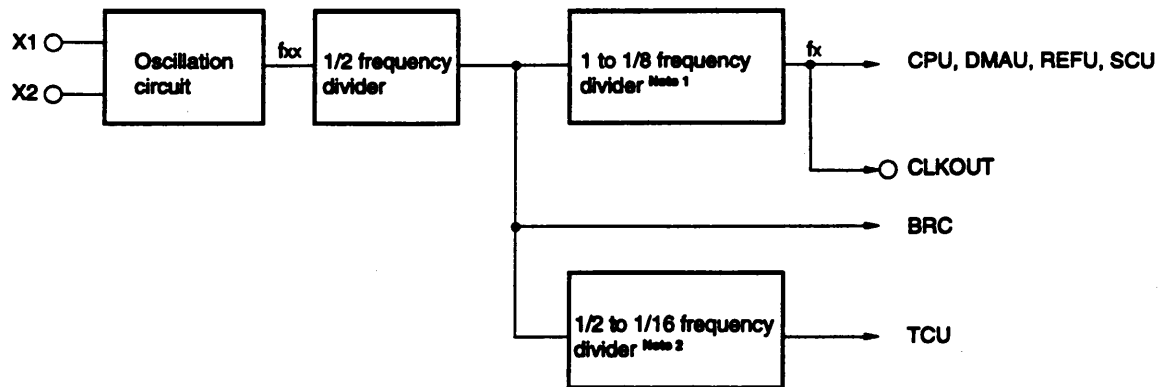
5.2.1 Features

- 40 MHz input (maximum)
- 20 MHz CLKOUT output (maximum)
- Variable instruction cycle times

5.2.2 Differences with the V40 and V50

- Variable instruction cycle times:
V40 and V50: No
V40HL and V50HL: Yes
- Maximum input frequencies:
V40 and V50: 20 MHz
V40HL and V50HL: 40 MHz

5.2.3 Internal block



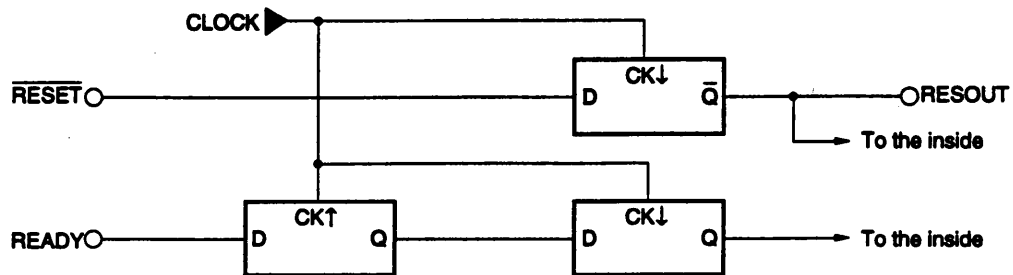
- Notes**
1. The frequency division rate is specified by the SBCR CLKC bit. (See Figure 6-23, Standby Control Register (SBCR).)
 2. The frequency division rate is specified by the TCKS PS bit. (See Figure 5-14, Timer Clock Selection Register (TCKS).)

5.3 Bus Interface Unit (BIU)

The bus interface unit (BIU) controls the pins of the data bus, address bus and control bus. Also, in the BIU, the CLOCK signal generated by CG is used to synchronize the $\overline{\text{RESET}}$ input signal and the READY input signal.

The synchronized reset signal becomes active-high and is supplied to the V40HL and V50HL as well as supplied to the outside from the RESOUT pin. The synchronized READY signal is supplied to the internal CPU, DMAU and REFU.

Figure 5-4. Synchronizing the $\overline{\text{RESET}}$ and READY Signals



5.4 Bus Arbitration Unit (BAU)

The bus arbitration unit (BAU) arbitrates the priority of use between the bus masters.

There are four types of bus masters (those functions with bus control authority), as indicated in the following table:

Table 5-2. Bus Masters

Bus Master	Bus Cycle
CPU	Program fetch and data read/write.
DMAU	DMA cycle
REFU	Refresh cycle
External bus master (HLDRQ pin input)	Bus cycles that drive external devices.

5.4.1 Priority

The priority of the bus masters is as shown below.

High	CPU (When it has a BUSLOCK prefix)
	REFU (Highest priority: When it reaches a set number of requests)
	DMAU
	HLDRQ pin
	CPU (During normal CPU cycles)
Low	REFU (Lowest priority: When at cycle steal)

REFU has two levels of priority. Normally, the refresh cycle is the lowest level of priority and cannot start unless the bus is completely idle. However, if the number of retained requests reaches seven or more, the refresh cycle becomes the highest-level of priority and requests the surrender of the bus.

The other bus masters cannot use the bus while the CPU is executing an instruction with a the BUSLOCK prefix. For this reason, when a BUSLOCK is attached to an instruction with a long execution time, such as a block instruction with a repeat, the refresh cycle must not exceed the period in which the DRAMs are refreshed.

5.4.2 Bus wait operation

As stated in the prior section, a priority is attached to each bus master. When a low priority bus master is using the bus and there is a request for use from a higher priority bus master, the lower priority bus master must relinquish the bus immediately.

Bus masters that are incorporated in the V40HL and V50HL, such as the CPU, DMAU and REFU, normally release the bus as soon as the current bus cycle terminates, as shown in Figure 5-5. However, in the case of an external bus master that is connected to the HLDRQ pin or cascaded external DMA controllers, the bus operation is as shown in Figure 5-6.

The V40HL and V50HL will inactivate the acknowledge signal (HLDK) and request the return of the bus. Therefore, an external bus master with priority to use the bus must accept this request and withdraw its bus hold request signal (HLDRQ). The higher priority bus master within the V40HL and V50HL must wait until the bus hold request signal is withdrawn.

This situation is called a bus waiting operation. If this wait is too long, it will have an impact on other operations, such as the DRAM refresh operation.

Figure 5-5. Internal Bus Cycles

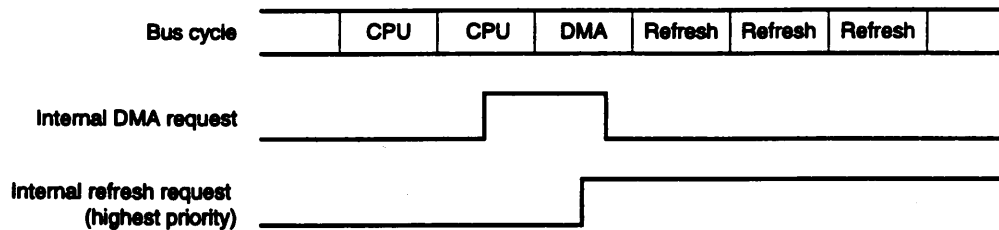
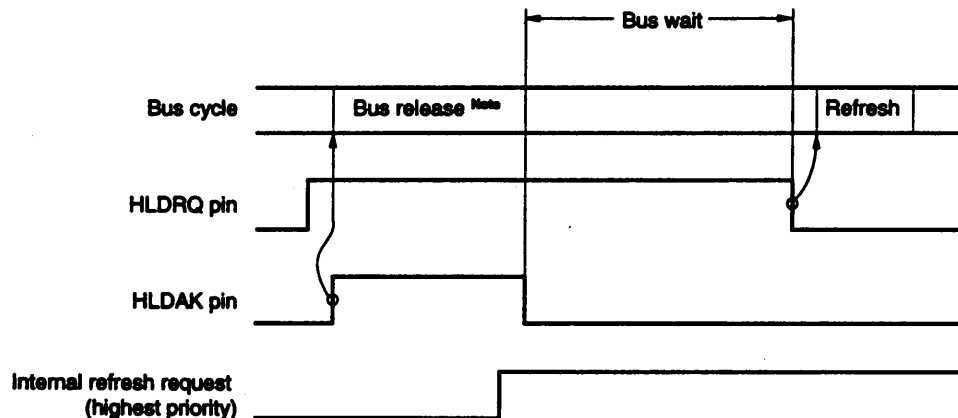


Figure 5-6. Bus Waiting Operation



Note This is the period in which the V40HL and V50HL release the bus and in which external bus masters that have received bus priority can use the bus.

5.4.3 Timing of bus priority switching

The priority for using the bus switches at each bus cycle and in the idle state (except for the T1 state immediately before the bus cycle).

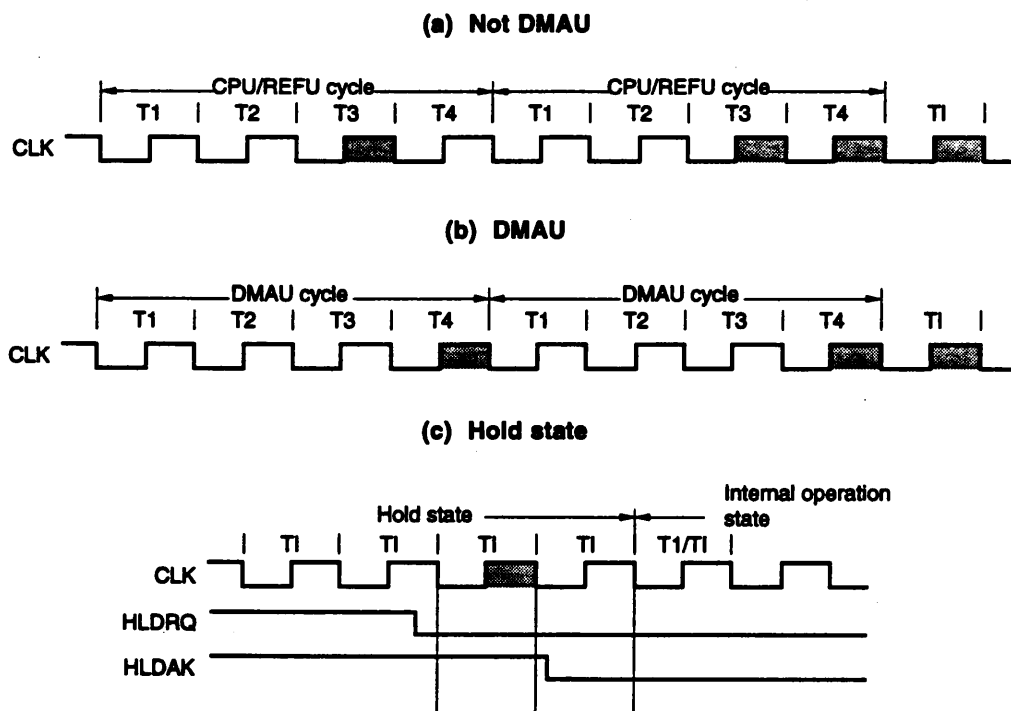
In the bus cycle, in the state immediately prior to the bus cycle termination state (T4, but T3 if there is no wait) and when entering the idle state following the bus cycle, the bus priority switches during the clock high level of the bus cycle termination state (T4).

However, during the operation of DMAU, switching always takes place in the bus cycle termination state (T4).

Also, when an external hold request has been received (when HLDAR is being output), the use of the bus switches during the T1 state clock high level, which is when HLDAR is inactive.

Bus use priority does not switch when the $\overline{\text{BUSLOCK}}$ signal is active.

Figure 5-7. Timing of Bus Priority Switching



Remark Bus control switches during the shaded high-level period.

Each bus master starts using the bus at the end of the state following the state in which the bus use priority was obtained. For this reason, except for DMAU, switching from one bus master to another bus master takes place without the unnecessary T1 state.

However, one T1 state cycle is inserted when switching from DMAU to another bus master.

Figure 5-8. Timing of Bus Master Switch Transition from Non-DMAU

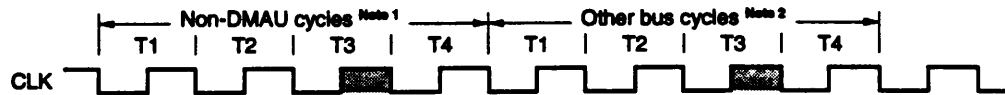


Figure 5-9. Timing of Bus Master Switch Transition from DMAU



- Notes 1. CPU, REFU and external bus masters, etc.
2. DMAU, CPU, REFU and external bus masters, etc.

Remark Bus control switches during the shaded high-level period.

5.4.4 DMA request and hold request acceptance waiting times

DMAU requests the use of the bus in the state after it samples a valid DMARQ. In addition, the HLDRQ pin is synchronized at the fall of the clock and then requests the use of the bus from BAU. For this reason, in the case of DMARQ, the DMA cycle starts at least three states after the sampled state. In the case of HLDRQ, HLDARQ is output at least two states after the sampled state.

The maximum wait time depends on how the BUSLOCK instruction is used. If the BUSLOCK instruction is not used, in the case of DMAU, the number of highest priority refresh cycles right after the interrupt acknowledge cycle is a maximum of five cycles^{Note}. In the case of HLDRQ, HLDRQ also depends on the DMAU operating state. But, when DMAU is not operating, the conditions are the same as the maximum DMAU wait time.

When the bus cycle is in a no-wait condition, the maximum number of wait until a DMA request or a hold request is received are as shown below:

V40HL:

DMARQ : 30.5 states ($2.5 + 4 \times 2 + 5 \times 4$)

HLDRQ : 30 states ($2 + 4 \times 2 + 5 \times 4$)

V50HL:

DMARQ : 34.5 states ($2.5 + 4 \times 3 + 5 \times 4$)

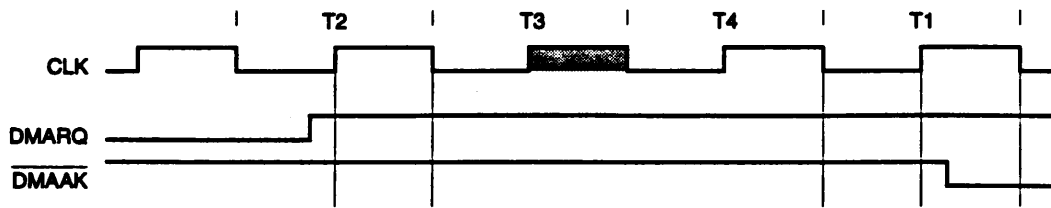
HLDRQ : 34 states ($2 + 4 \times 3 + 5 \times 4$)

Note When the next refresh request comes during the highest priority refresh cycle, it will take five cycles. It normally takes four cycles.

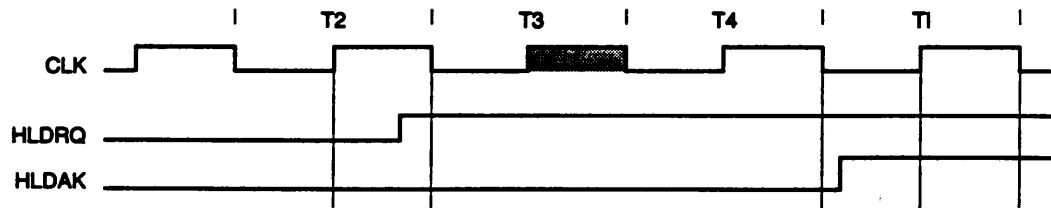
Remark The last 5T1 state of the interrupt acknowledge timing does not have to be included in the above calculation because it will be replaced by the refresh cycle.

Figure 5-10. Minimum Wait Time

(a) At start of DMA cycle



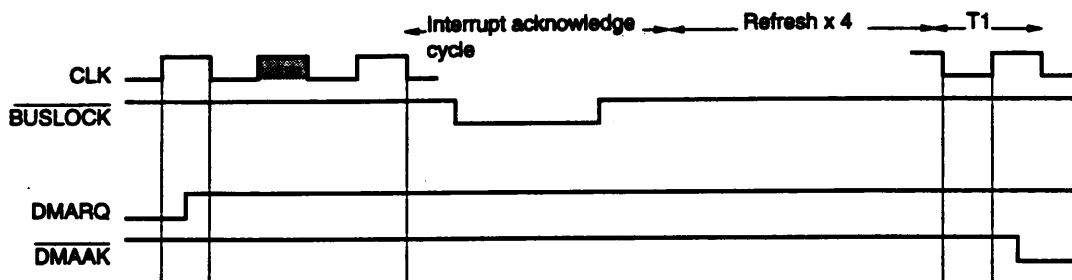
(b) At start of hold cycle



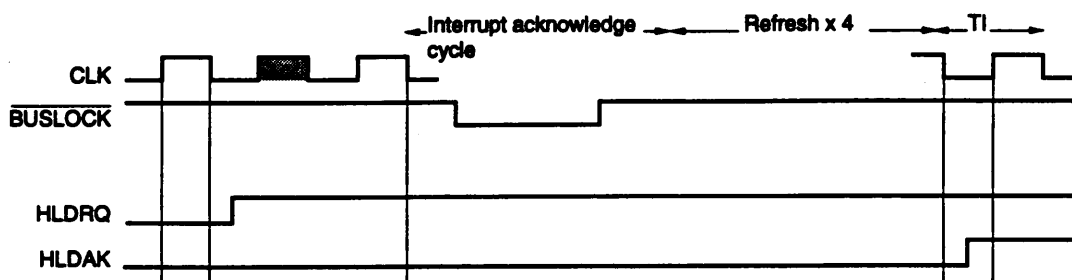
Remark Bus control switches during the shaded high-level period.

Figure 5-11. Maximum Wait Time

(a) At start of DMA cycle



(b) At start of hold cycle



Remark Bus control switches during the shaded high-level period.

5.5 Wait Control Unit (WCU)

The wait control unit (WCU) has the function of inserting from zero to three clocks of wait states (TW) for low-speed memory and I/O devices. The number of wait clocks inserted for the CPU cycle, external I/O device cycle, DMA cycle and refresh cycle can be programmed independently. For details on this function see section 6.1 Wait Functions.

5.5.1 Features

- The 1-Mbyte memory space can be divided into five partitions (three partitions immediately after a reset).
- 64-Kbyte I/O space can be divided into three partitions (none immediately after a reset).
- Automatic programming of from zero to three waits for the CPU memory cycle.
- Automatic programming of from zero to three waits for the external I/O device cycle.
- Automatic programming of from zero to three waits for the refresh cycle.
- Automatic programming of from zero to three waits for the DMA cycle.

5.5.2 Differences with the V40 and V50

- Number of memory space partitions:
V40 and V50 : Three partitions possible
V40HL and V50HL: Five partitions possible
- Number of I/O space partitions:
V40 and V50 : No partitions
V40HL and V50HL: Three partitions possible
- System I/O space registers:
V40 and V50 : WCY1, WCY2, WMB
V40HL and V50HL: WCY1 to WCY3, WMB, EXWB, WSMB, WIOB

5.6 Refresh Control Unit (REFU)

The refresh control unit (REFU) generates a 16-bit refresh address and a refresh signal ($\overline{\text{REFRQ}}$) that indicates a refresh cycle to support the refresh operations of DRAMs.

For details on this function, see section 6.2 Refresh Function.

5.6.1 Features

- Lowest-priority and highest-priority refresh operations.
- Seven refresh queues.
- 16-bit refresh address.
- Supports $\overline{\text{REFRQ}}$ expansion timing ($\overline{\text{REFRQ}}$ becomes active from the T1 state).

5.6.2 Differences with the V40 and V50

- Refresh address length:
 - V40 and V50 : 9 bits
 - V40HL and V50HL: 16 bits
- Support of $\overline{\text{REFRQ}}$ expansion timing:
 - V40 and V50 : No support
 - V40HL and V50HL: Supports

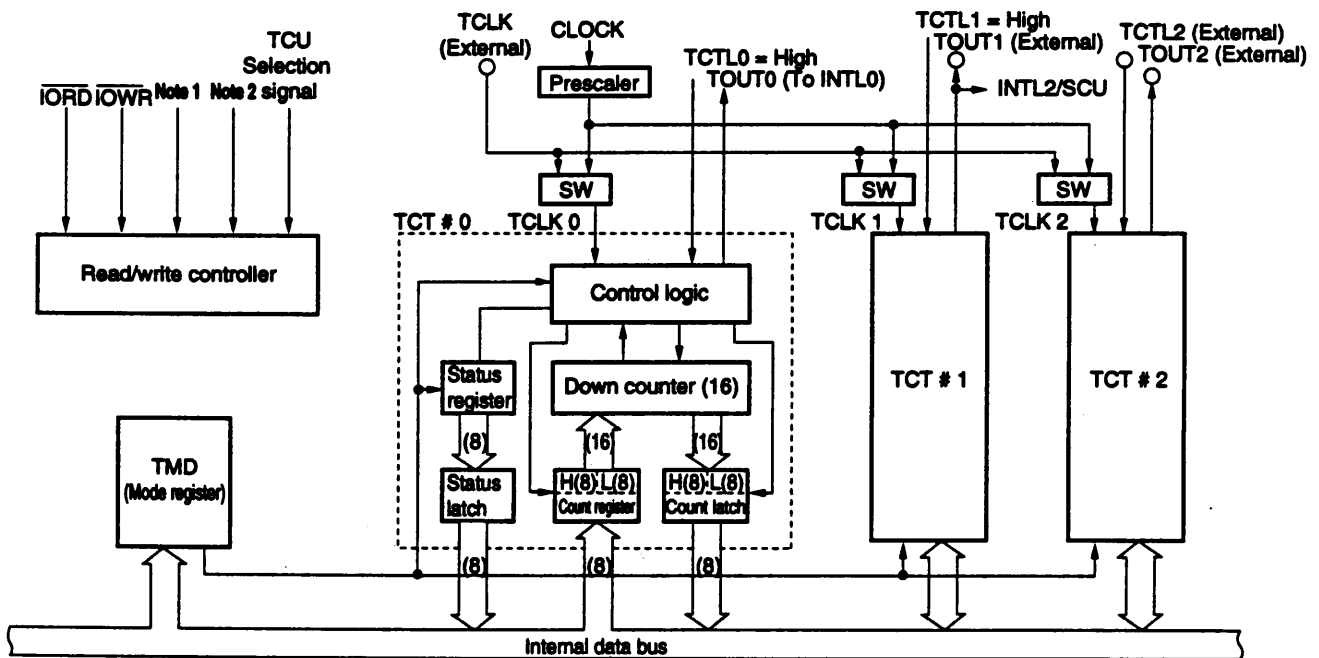
5.7 Timer/Counter Unit (TCU)

The timer/counter unit (TCU) has three sets of counters and is functionally the same as the μ PD71054.

5.7.1 Features

- Three timer channels
- Three 16-bit counters
- Six programmable counter modes
- Binary/BCD counting
- Multiple latch command
- Count latch command
- Selectable internal/external input clock

5.7.2 Internal block diagram



- Notes 1. A0 (when IOAG = 1) or A1 (when IOAG = 0)
 2. A1 (when IOAG = 1) or A2 (when IOAG = 0)

5.7.3 Addressing

The TCU internal registers are mapped to addresses determined by IOAG bits of OPHA, TULA, and the SCTL set in the system I/O area. (For the methods of pointing to addresses, refer to section 4.1.3 System I/O Area.)

Table 5-3. TCU Internal Registers and Command Addresses

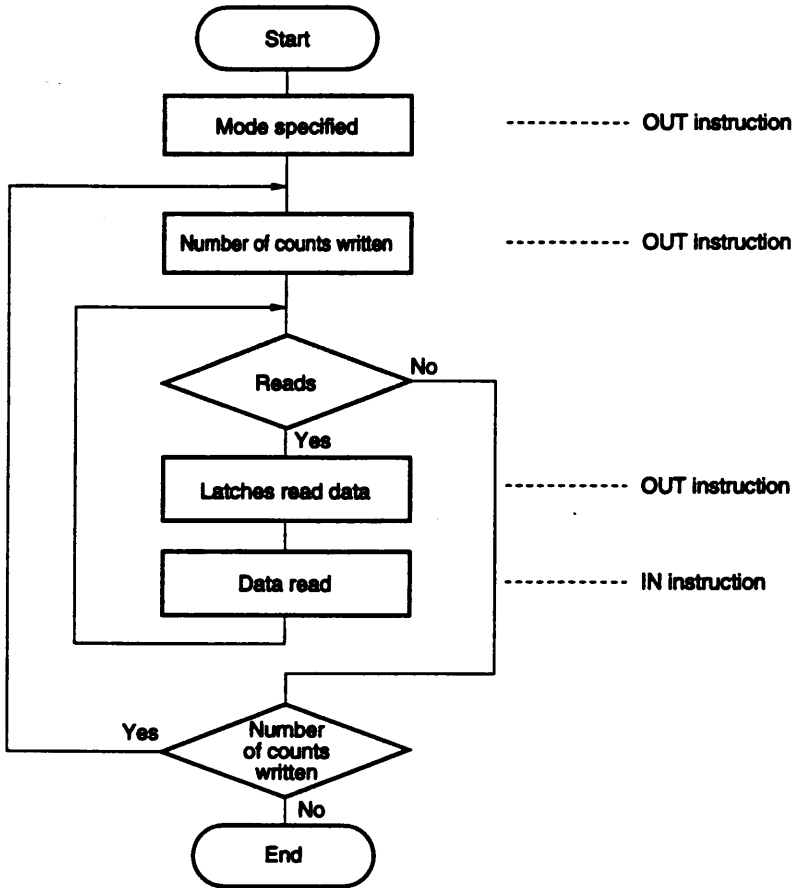
<div></div>	High-order Addresses	Low-order Addresses						Registers and Commands		Operation
When IOAG = 1	A15 - A8 (OPHA)	A7 A6 A5 A4 A3 A2 (TULA)	0	0			TCT#0	Read/write		
							TST0	Read		
			0	1			TCT#1	Read/write		
							TST1	Read		
			1	0			TCT#2	Read/write		
							TST2	Read		
1	1			TMD	Write					
When IOAG = 0	A15 - A8 (OPHA)	A7 A6 A5 A4 A3 (TULA)	0	0	A0			TCT#0	Read/write	
								TST0	Read	
			0	1			TCT#1	Read/write		
							TST1	Read		
			1	0			TCT#2	Read/write		
							TST2	Read		
1	1			TMD	Write					
—	1 1 1 1 1 1 1 1	1 1 1 1 0 0 0 0					TCKS	Read/write		

Remark The address of TCKS is FFF0H regardless of the content of the IOAG bit, OPHA and TULA.

5.7.4 Operating procedures

After power is turned on, TCU is in an undefined state. For this reason, to use it requires programming the target counter to specify the operating mode. Once a counter is programmed to a certain mode, it will operate in that mode until another mode is specified for it. If a certain number of counts is written to a counter and that is transferred to the down counter, a new count will begin. During a count, count data that is a value of the current counter and the status data that indicates the counter condition can be read.

Figure 5-12. TCU Basic Operational Procedures



5.7.5 Registers and commands

The issuance of read/write commands from the TCU register takes place by means of the I/O instruction for the address set in the system I/O area. The selection of register commands takes place by using A1 and A0 (when IOAG = 1) or A2 and A1 (when IOAG = 0).

Table 5-4. TCU Registers and Commands Addresses

When IOAG = 0		When IOAG = 1		Registers and Commands	Operation
A2	A1	A1	A0		
0	0	0	0	TCT#0	Read/write
				TST0	Read
0	1	0	1	TCT#1	Read/write
				TST1	Read
1	0	1	0	TCT#2	Read/write
				TST2	Read
1	1	1	1	TMD	Write
Note				TCKS	Read/write

Note The address of TCKS is fixed at FFF0H.

To write to the timer mode register (TMD), involves setting the operating mode of each counter within TCU (counter mode, binary BCD, read/write mode) and issuing a command to latch the counter value (counter latch command, multiple latch command).

Timer/counter register 0 to timer/counter register 2 (TCT#0 to TCT#2) are used to write the number of counts to each counter and to read the count data. Normally, reading the count data takes place after the issuing a count latch command or multiple latch command beforehand and then latching the count data of the target counter.

For reading TST0 to TST2, the status data of each counter is read. Reading of statuses takes place after latching the status of the target counter using the multiple latch command.

If the status and count data of one counter are both latched, the first read will yield the status data and the next read will yield the count data.

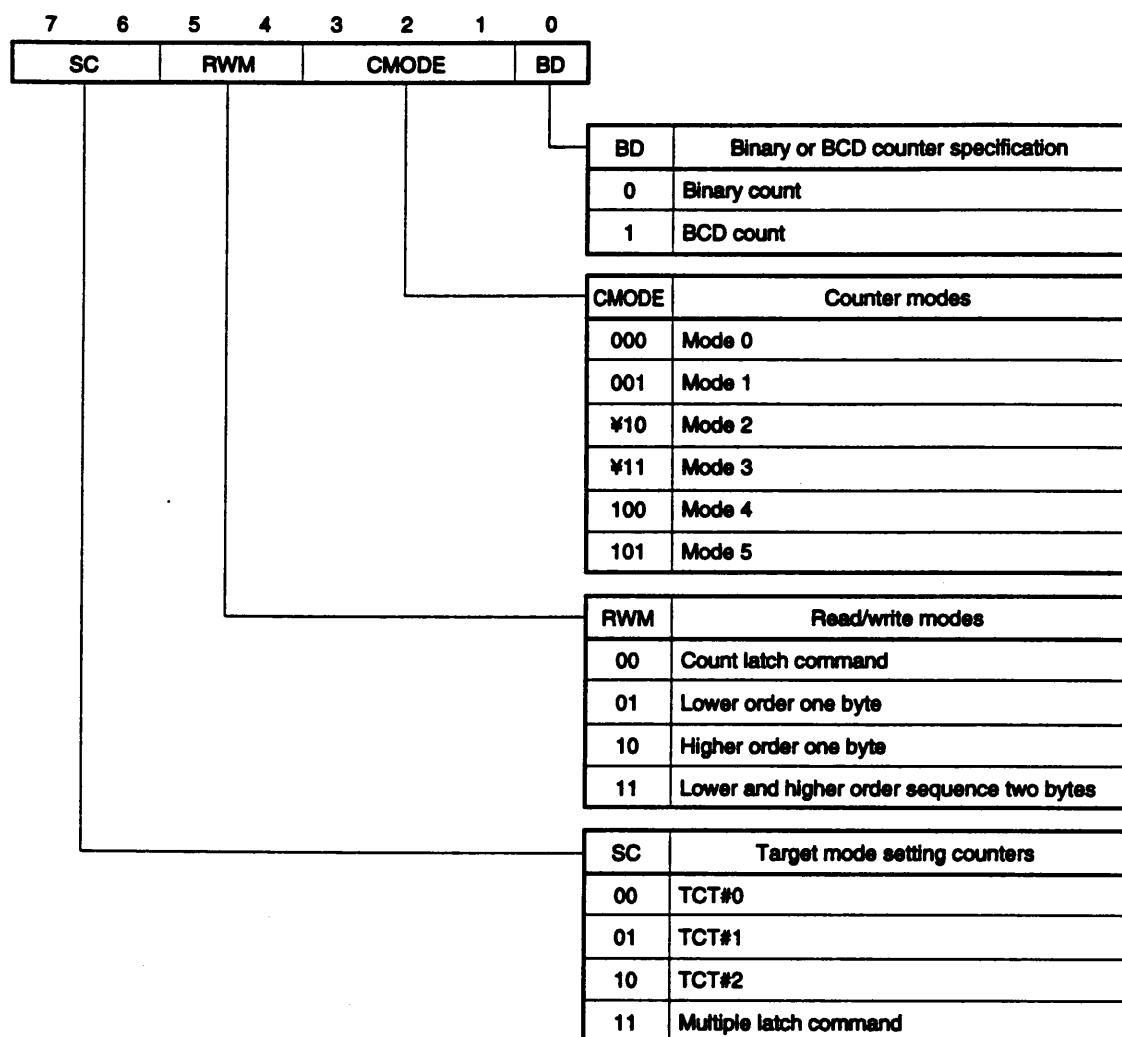
(1) Timer Mode Register (TMD)

The timer mode register (TMD) contains the mode word that sets the operating mode of each of the counters as well as the following commands, which latch the counter value:

- Counter latch command (See number (4))
- Multiple latch command (See number (5))

To operate the counter, first write the mode word to TMD to set the mode in each counter.

Figure 5-13. TMD (For Mode Word)



Remark x: don't care.

(a) SC Bit

The SC bit specifies the counters subject to mode setting (TCT#0 to TCT#2) or it specifies the multiple latch command. When a counter is specified, the setting of each bit of RWM, CMODE and BD are valid for the counter specified.

When a multiple latch command is specified, the meaning of bits 0 to 5 will be different. For details, refer to number (5), which concerns multiple latch commands and number (6), which concerns precautions when issuing multiple latch commands.

(b) RWM Bit

The RWM bit specifies a write to the count register, the read/write mode that specifies the reading of the count latch, or specifies the count latch command. For details, refer to number (4), which concerns the count latch command.

(c) CMODE Bit

The CMODE bit specifies count modes zero to five. For the operation of each count mode, refer to **section 5.7.6 Count modes**.

(d) BD Bit

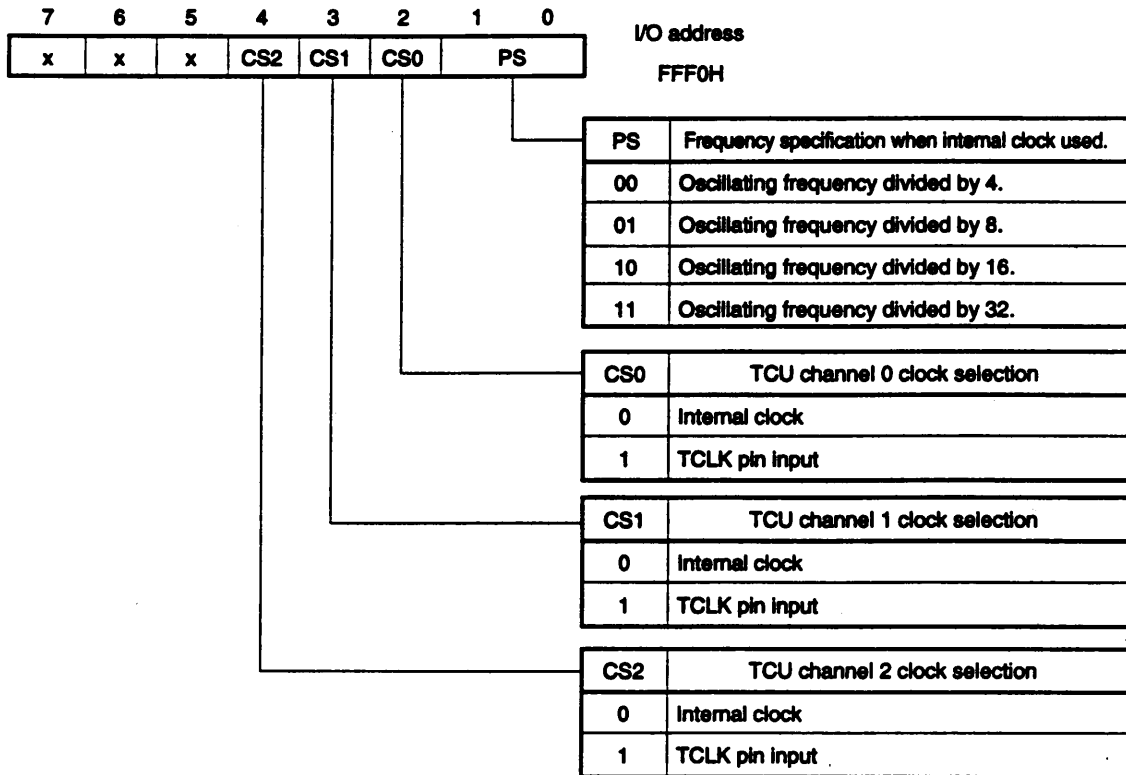
The BD bit specifies a binary count and a BCD count. For a binary count, count settings from 0000H to FFFFH are possible, and binary counting takes place. For BCD, decimal count settings from zero to 9999 are possible.

(2) Timer Clock Selection Register (TCKS)

The timer clock selection register (TCKS) selects either the external clocks (TCLK0 to TCLK2) or the internal clock (a clock in which the frequency of the crystal that connects to the X1 and X2 pins has been divided) as the source for supplying a clock to the three counters within TCU (TCT#0 to TCT#2).

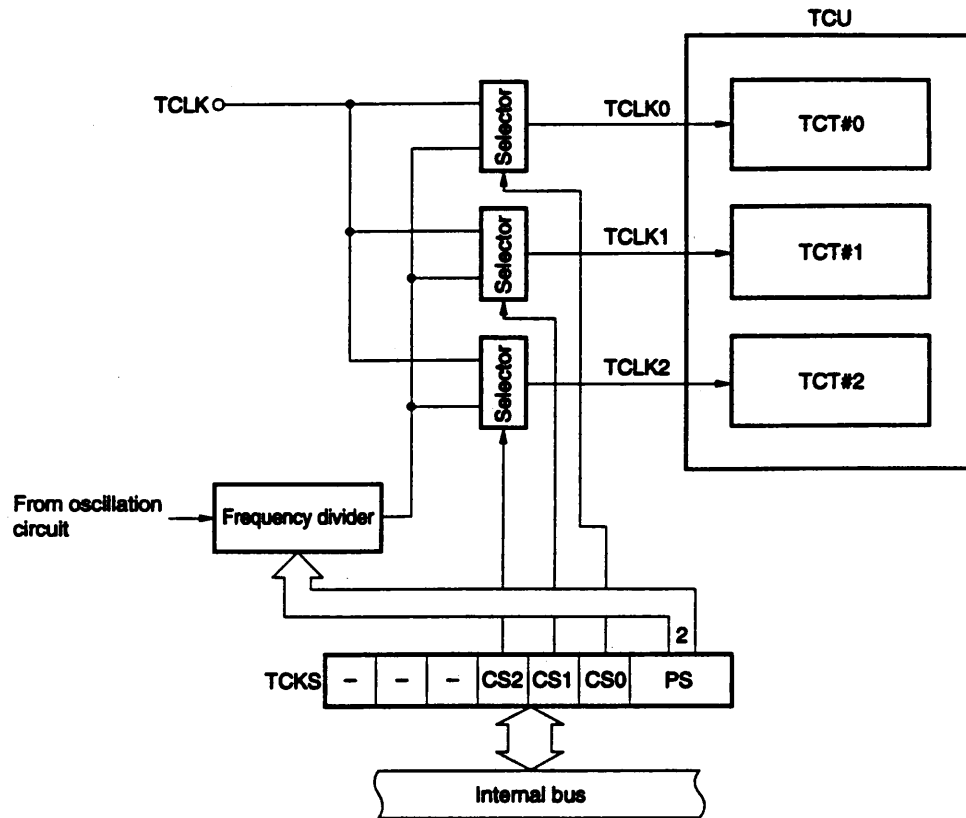
The TCKS address is fixed at FFF0H of the system I/O area. This is different from other TCU internal registers.

Figure 5-14. Timer Clock Selection Register (TCKS)



Remark x: don't care

Figure 5-15. Supply a Clock to the TCU



(3) Timer/Counter Register #0 to Timer/Counter Register #2 (TCT#0-TCT#2)

Each channel has one 16-bit length count register. Writing to and reading from these registers takes place according to the read/write mode set by means of a mode word. If set to the low-order one byte and high-order one byte modes, the high-order byte or the low-order byte of the respective count register will be written to in one write operation. In this case, the remaining high-order byte or low-order byte will become 00H. In the low-order and high-order two-byte mode, the first write operation will write to the low-order byte, while the second write operation will write to the high-order byte at the same address.

Table 5-5. Writing to the Count Register

Read/Write Mode	Number of Writes	Count Register	
		High-order byte	Low-order byte
Low-order 1 byte	1	00H	xxH
High-order 1 byte	1	xxH	00H
Low- and high-order 2 bytes	2	xxH (second time)	xxH (first time)

Reading from a counter is basically the same as writing to one. In the two-byte mode, for reading the low- and high-order bytes, after reading the first low-order byte with a read operation, the high-order byte of the same address will be read in the second read operation.

The following are the three read sequences:

- (a) A direct read from the counter.
- (b) A read after a counter latch command.
- (c) A read after a multiple latch command.

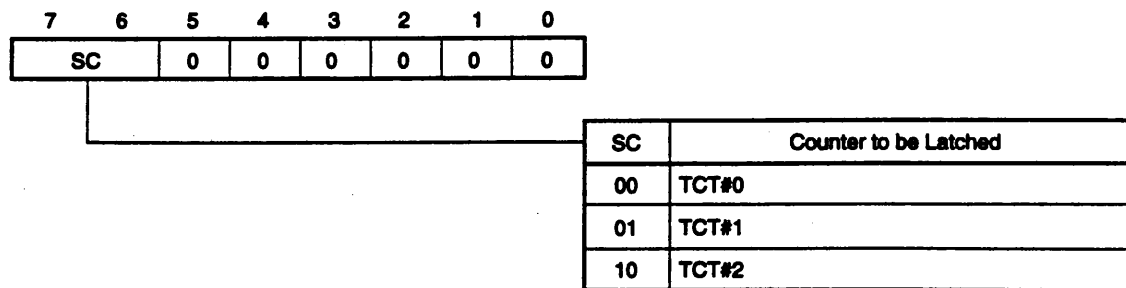
In the case of reading directly from the counter, the count latch following the down counter will be read. Consequently, this value may change while it is being read, and the value obtained will be inaccurate. To obtain an accurate value, the TCLKn input or TCTL input must be manipulated so that reading can take place while the counting has stopped. Normally, reading takes place using methods (b) and (c). Method (c) allows both the count data and the status data to be read.

Table 5-6. Reading the Counter

When IOAG = 0		When IOAG = 1		Counter Read
A2	A1	A1	A0	
0	0	0	0	TCT#0, TST0
0	1	0	1	TCT#1, TST1
1	0	1	0	TCT#2, TST2

(4) Counter Latch Command

Figure 5-16. TMD (When there is a count latch command)

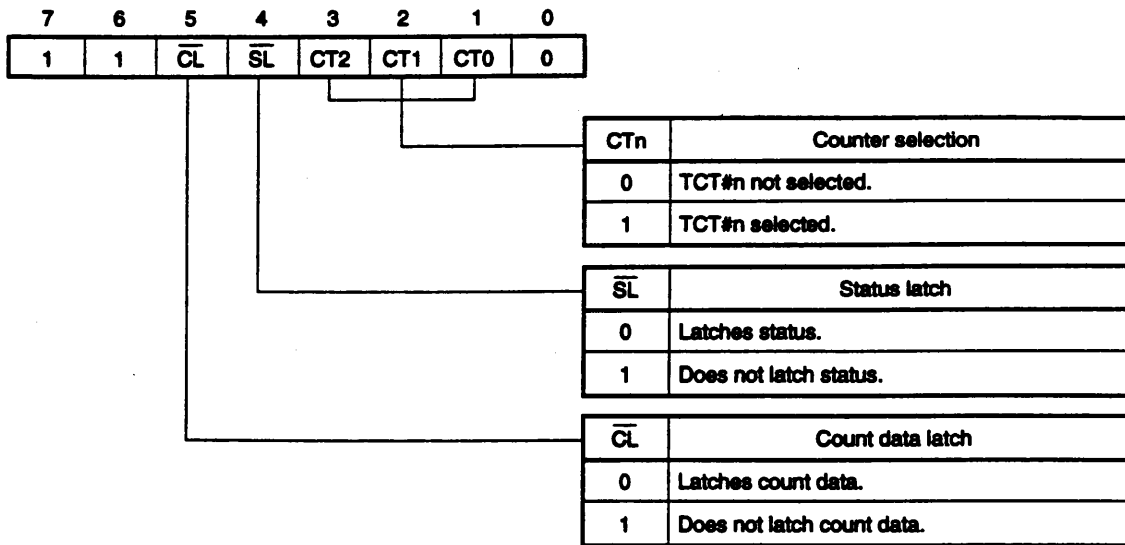


The count latch command is issued by writing all zeros, from bit zero to bit 5, to TMD. When a count latch command is issued, the content of the down counter selected by the SC bit will be latched by the count latch. Latched count data will be maintained until it is read or another mode setting is made. This count latch command allows accurate count data when the command is issued without having an impact on the counting operation.

If a count latch command is issued and another count latch command is issued for the same count before the original command is read, the latter command will be ignored and the value that was latched previously will be maintained. If the latched count data is read, the latch will be released and the counter will return to its original following down counter state.

(5) Multiple Latch Command

Figure 5-17. TMD (When there is a multiple latch command)



The multiple latch command is issued by writing data that has bit 7 and bit 6 of 1.

When a multiple latch command is issued, all of the selected count data and the status will be latched to the count latch and status latch. If the counter is read while in the latched condition, the counter data and status can be read in that condition without having an impact on counter operation.

(a) \overline{CL} Bit

Defines whether the count data of all counter selected will be latched or not latched.

(b) \overline{SL} Bit

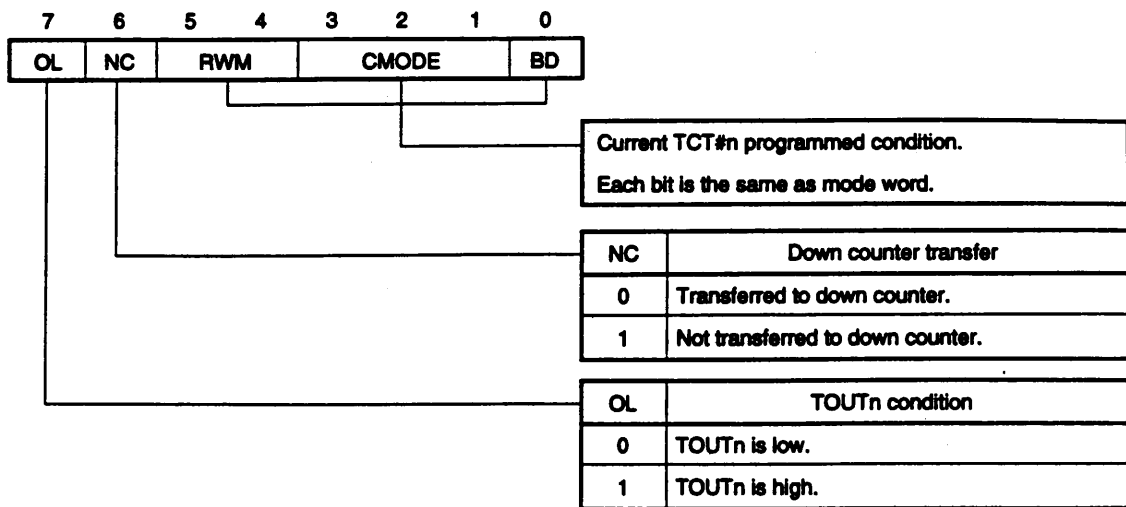
Defines whether the statuses of the counters selected will be latched or not latched.

(c) CT0 to CT2

Defines the counter to be latched. A multiple of counter count data and statuses can be latched simultaneously.

The status indicates the counter operating condition when a multiple latch command is issued. The content of the latched status can be obtained by reading TSTn. TSTn is shown in Figure 5-18.

Figure 5-18. TSTn



The OL bit shows the counter output state when a multiple latch command is issued.

The NC bit is no count flag, and indicates whether or not the most recently written number of counts have been transferred to the down counter.

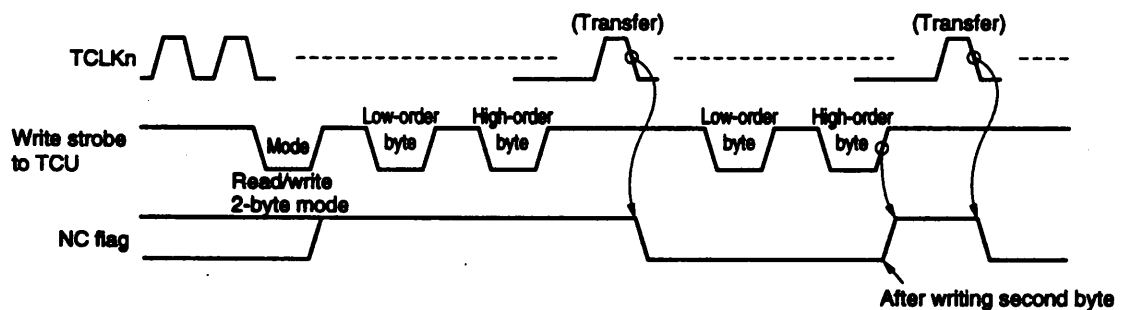
The BD, CMODE, and RWM bits indicate the programmed condition of the counter and is the same as the value set by the mode word.

Table 5-7. NC Flag Changes

Counter Operation	NC Flag
Writing of mode word (to corresponding counter)	1
Writing of count to count register ^{Note}	1
Transferring count from count register to down counter	0

Note If the read or write is in the two-byte mode, the flag will set to one when the second byte is written.

Figure 5-19. Example of NC Flag Changes



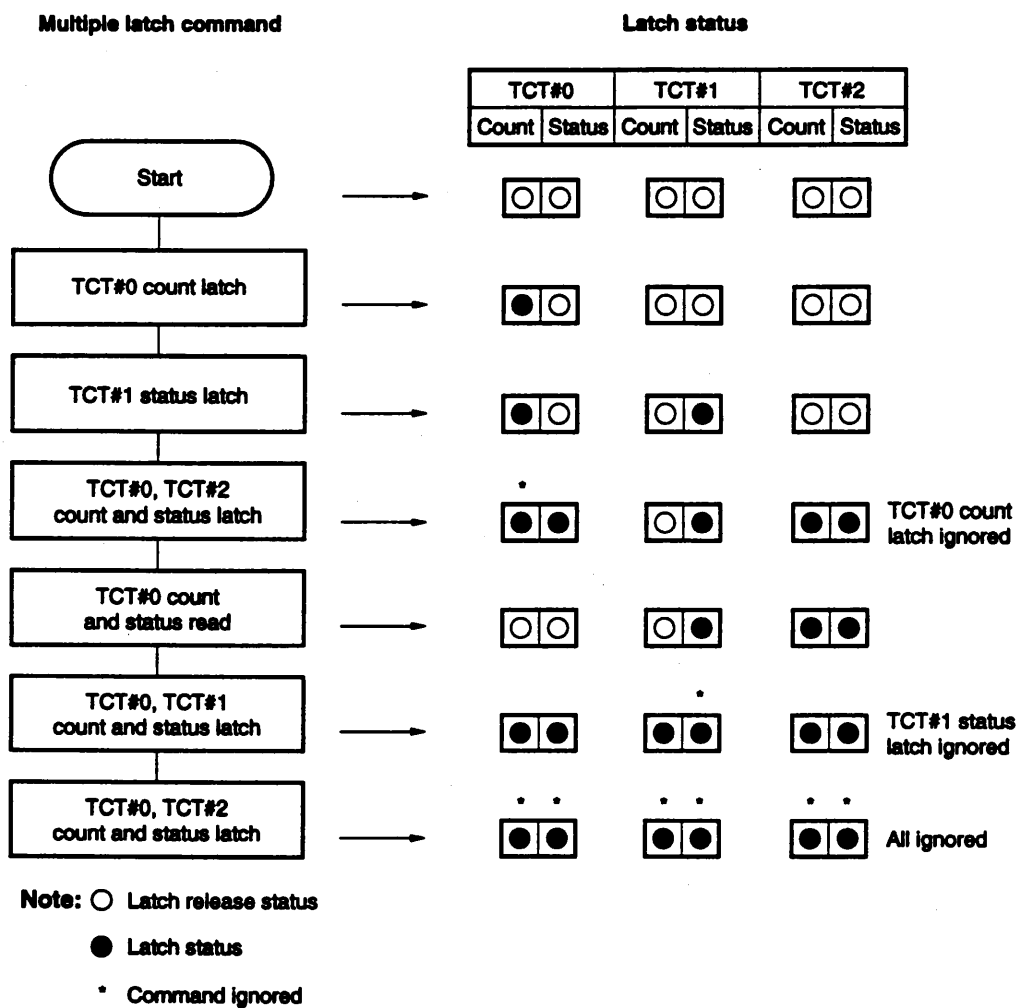
(6) Precautions When Issuing the Multiple Latch Command

The count latch and status latch that are latched using the multiple latch command maintain their data until either they are read or a new mode setting is made.

That is, after they are latched, if another multiple latch command is issued before they have been read, that command will be ignored. Moreover, once either the count latch or the status latch is read, the latch will be released. A multiple latch command execution example is shown in Figure 5-20.

With the multiple latch command, even if either of the count data or status data is latched first, the status will always be read at the first read operation. The count data will be read at the subsequent one or two readings (which differ depending on the read/write mode). Further, if the reading continues, the count value in the state following the down counter, which has not been latched, will be read.

Figure 5-20. Multiple Latch Command Execution Example



5.7.6 Count mode

Here, the six count modes that are set by the mode word CMODE bit will be described. Definitions of the terms used in the description of each mode are as follows:

TCLKn pulse : From the rise of the TCLKn input rise to the next fall.



Trigger : The TCTL input rising edge.



TCTL : Sampled each time TCLKn rises. Among the sampling are level detection and rise detection (trigger).

TOUT initial state : The TOUT output state determined instantly when the count mode is set using a mode word.

Count transfer : The transfer of the count from the count register to the down counter.

Decrement : Down counter operation (which takes place when the TCLKn pulse rises).

Count zero : State when the content of the down counter becomes zero.

LB : Low-order byte of the count.

HB : High-order byte of the count.

The operating examples (timing charts) in each count mode are the instances when TCT#2 was executed with the read/write one-byte mode binary counter. If the TCTL2 signal is omitted, the TCTL2 input is high. The hexadecimal numbers below the TOUT signal express the count at that point. A question mark right after the mode specification value indicates an undefined value. In other locations, it indicates a continuation of a previous count. The maximum count setting in each mode is zero.

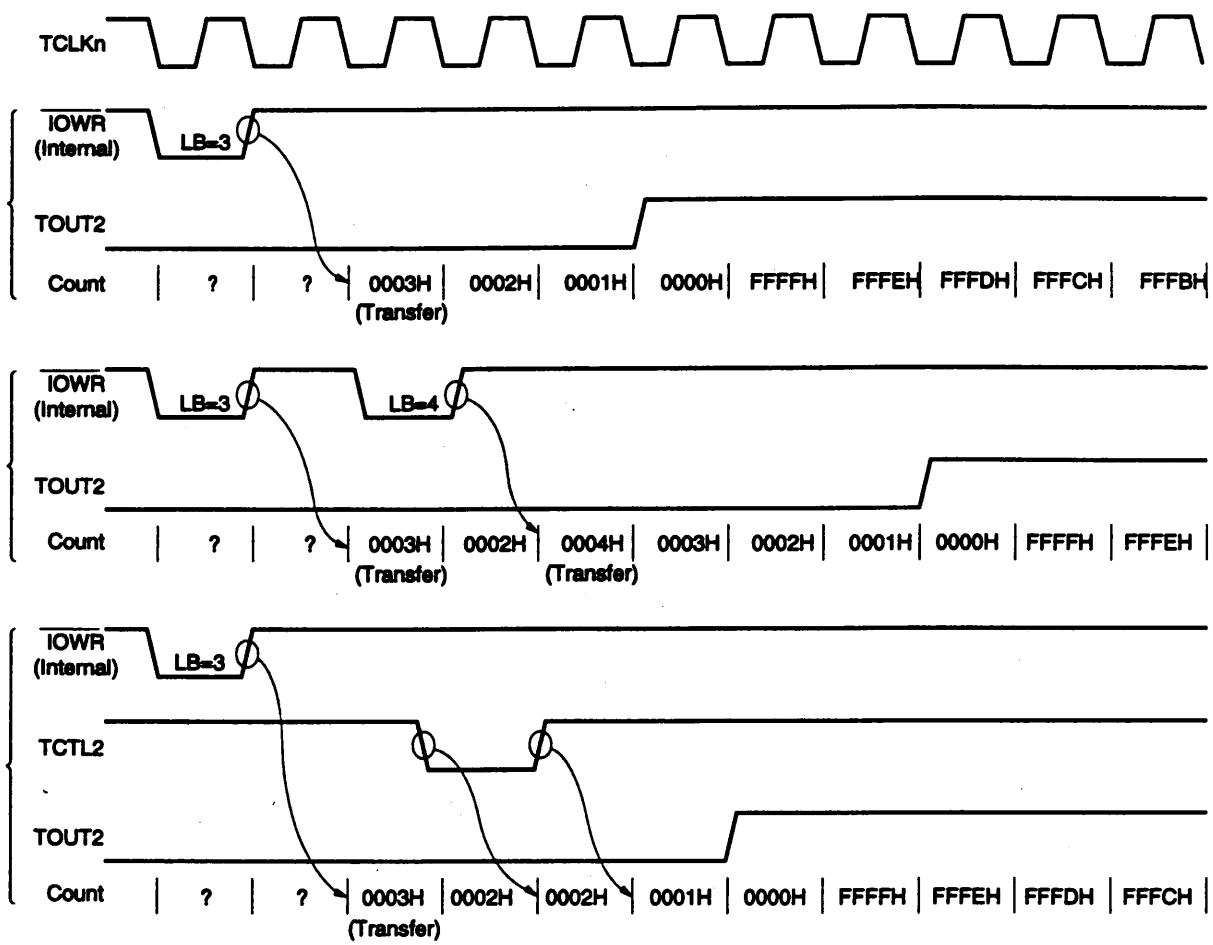
(1) Mode 0: Count End Signal Output

When the specified count has ended, the TOUT pin changes at the same time from low to high.

Table 5-8. Mode 0 Operation

Item		Description
TOUT initial state		Low level
TCTL input	High level	Count state
	Low level	Count prohibited state
Count writing		The TOUT pin goes low (TCLKn pulse irrelevant). If the read/write is in the 2-byte mode, the count is canceled when the first byte is written and the TOUT pin goes low.
Count transfer and counting		<p>If the count is written when TCTL is high, transfer takes place at the next TCLKn pulse in which the count is written. Decrementing begins from the TCLKn pulse that follows the transfer. Thus, if the count is N, the TOUT pin goes low during the $(N + 1) \times \text{TCLKn}$ pulse.</p> <p>If the count is written when TCTL is low, transfer takes place at the next TCLKn pulse in which the count is written. When TCTL goes high, decrementing begins at the following TCLKn pulse. Thus, if the count is N, the TOUT pin goes low during the $N \times \text{TCLKn}$ pulse.</p>
Count zero		The TOUT pin goes high. The counting does not stop. If the counting is binary, the count down takes place to FFFFH, if BCD, it takes place to 9999.
Minimum count		1

Figure 5-21. Mode 0 Operation Example



(2) Mode 1: Control Pin Triggerable One-Shot

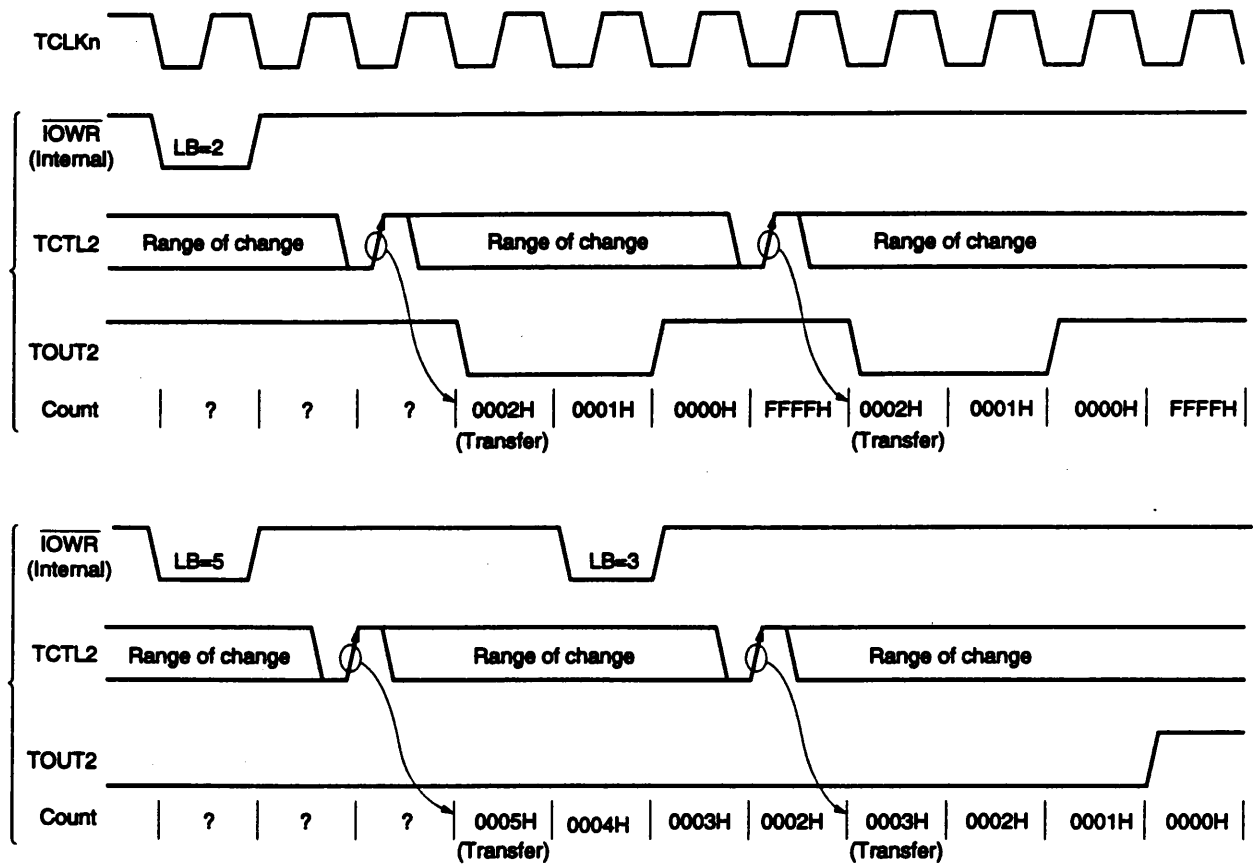
A designated-length low-level one-shot pulse is output to the TOUT pin. Retriggering is possible through TCTL input.

Table 5-9. Mode 1 Operation

Item		Description
TOUT initial state		High level
TCTL input	Trigger ^{Note}	Transfer of the count start with the next TCLKn pulse after the trigger.
Writing the count		Will be written without influencing the current operation.
Count transfer and count		If there is a trigger, transfer takes place at the next TCLKn pulse and the TOUT pin goes low and the one-shot pulse begins at the same time. Decrementing takes place at the next from the next TCLKn pulse. Thus, if the count is N, The TOUT pin one-shot output will continue during the Nx TCLKn pulse.
Count zero		The TOUT pins goes high. The counting does not stop. If the counting is binary, the count down takes place to FFFFH, if BCD, it takes place to 9999.
Minimum count		1

Note The trigger will be ignored if the count has not been written right after the mode specification and if only one byte has been written in the read/write two-byte mode.

Figure 5-22. Mode 1 Operation Example



(3) Mode 2: Rate Generator

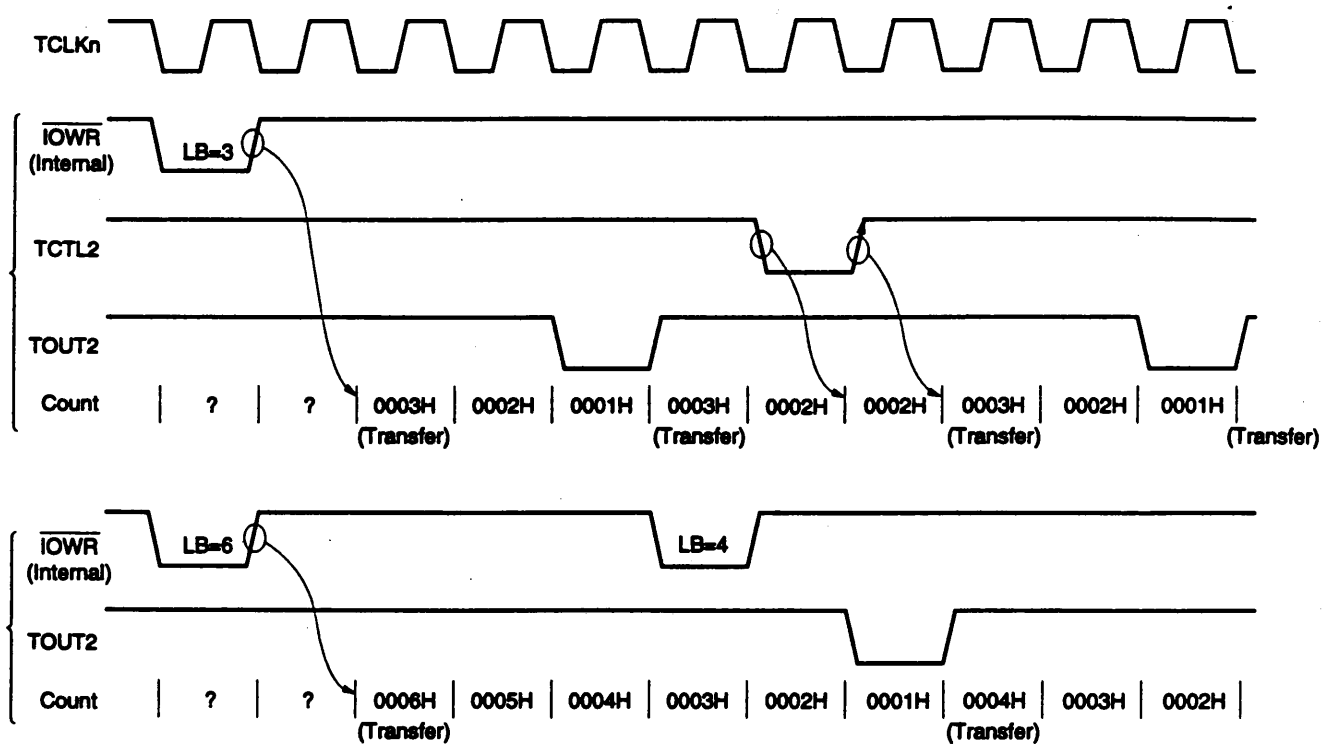
This is a frequency divider counter that cyclically places the TOUT pin at the low level for the final 1TCLKn pulse of a designated counter.

Table 5-10. Mode 2 Operation

Item		Description
TOUT initial condition		High level
TCTL input	High level	Count state
	Low level	The count prohibited state. If TCTL goes low when the TOUT pin is low, the TOUT pin will go high. (TCLKn pulse is irrelevant)
	Trigger ^{Note}	Count transfer will take place at the TCLKn pulse after the trigger.
Writing the count		Will be written without influencing the current operation.
Count transfer and count		Will be transferred at the TCLKn pulse after writing the count that follows the mode specification. After that, the count will decrement, and transfer will take place at the TCLKn pulse after the count becomes 1. Also, if there is a trigger, transfer will also take place at the TCLKn pulse right after it. As for the TOUT pin, at the same time that the content of the down count goes to one, the TOUT pin will go low during 1TCLKn and then return to high. Thus, if the count is N, the operation will repeat in N x TCLKn pulse.
Count zero		Will not occur in this mode.
Minimum count		2

Note The trigger will be ignored if the count has not been written right after the mode specification and if only one byte has been written in the read/write two-byte mode.

Figure 5-23. Mode 2 Operation Example



(4) Mode 3: Square Wave Generator

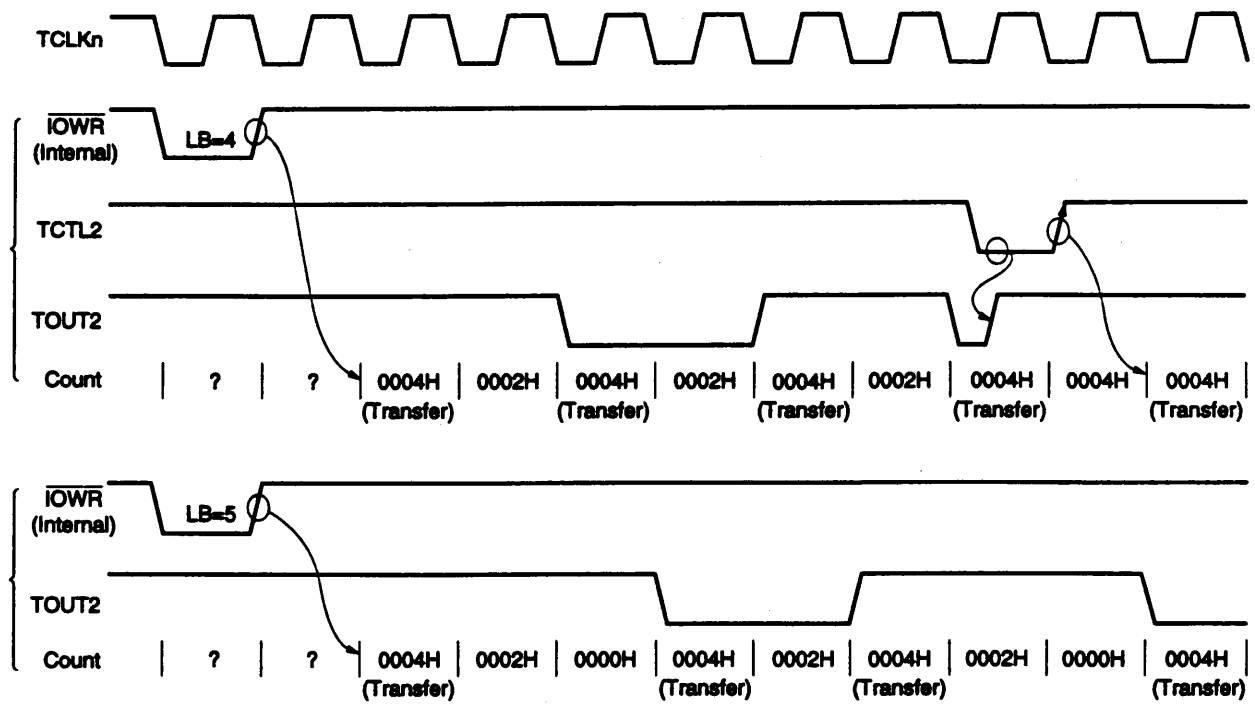
This is the same type of frequency counter as that of mode 2, but the duty is different.

Table 5-11. Mode 3 Operation

Item		Description
TOUT initial condition		High level
TCTL input	High level	Count state
	Low level	The count prohibited state. If TCTL goes low when the TOUT pin is low, the TOUT pin will go high. (TCLKn pulse is irrelevant)
	Trigger ^{Note}	Count transfer will take place at the TCLKn pulse after the trigger.
Writing the count		Without influencing the current operation, the count transfer will take place at half cycles of the current square wave. At the same time, the TOUT signal will invert.
Count transfer and count		<p>The count will be transferred at the TCLKn pulse after the count that follows the mode specification is written. After that, transfer will continue after the completion of the current half cycle. Then, the TOUT pin output will invert. Also, a transfer will take place at the TCLKn pulse after the trigger.</p> <p>The operation will vary depending on whether count N is an even number or an odd number. If count N is an even number, it will decrement two at a time after it is transferred. When the count reaches 2, there will be a transfer at the next TCLKn pulse, and the state of the TOUT pin will invert. The subsequent operations will repeat with this as a half cycle.</p> <p>If count N is an odd number, N-1 will be transferred, and the count will decrement two at a time. In the half cycle when the TOUT pin is high, the count is zero. N-1 will once again be transferred at the next TCLKn pulse. In the half cycle in which TOUT goes low, the count will only continue until it reaches 2. Therefore, N-1 will be transferred again at the next TCLKn pulse. The result is that at the half cycle when the TOUT pin is high, the cycle is longer by one TCLKn.</p>
Count zero		Can happen only if the count is an odd number.
Minimum count		2

Note The trigger will be ignored if the count has not been written right after the mode specification and if only one byte has been written in the read/write two-byte mode.

Figure 5-24. Mode 3 Operation Example



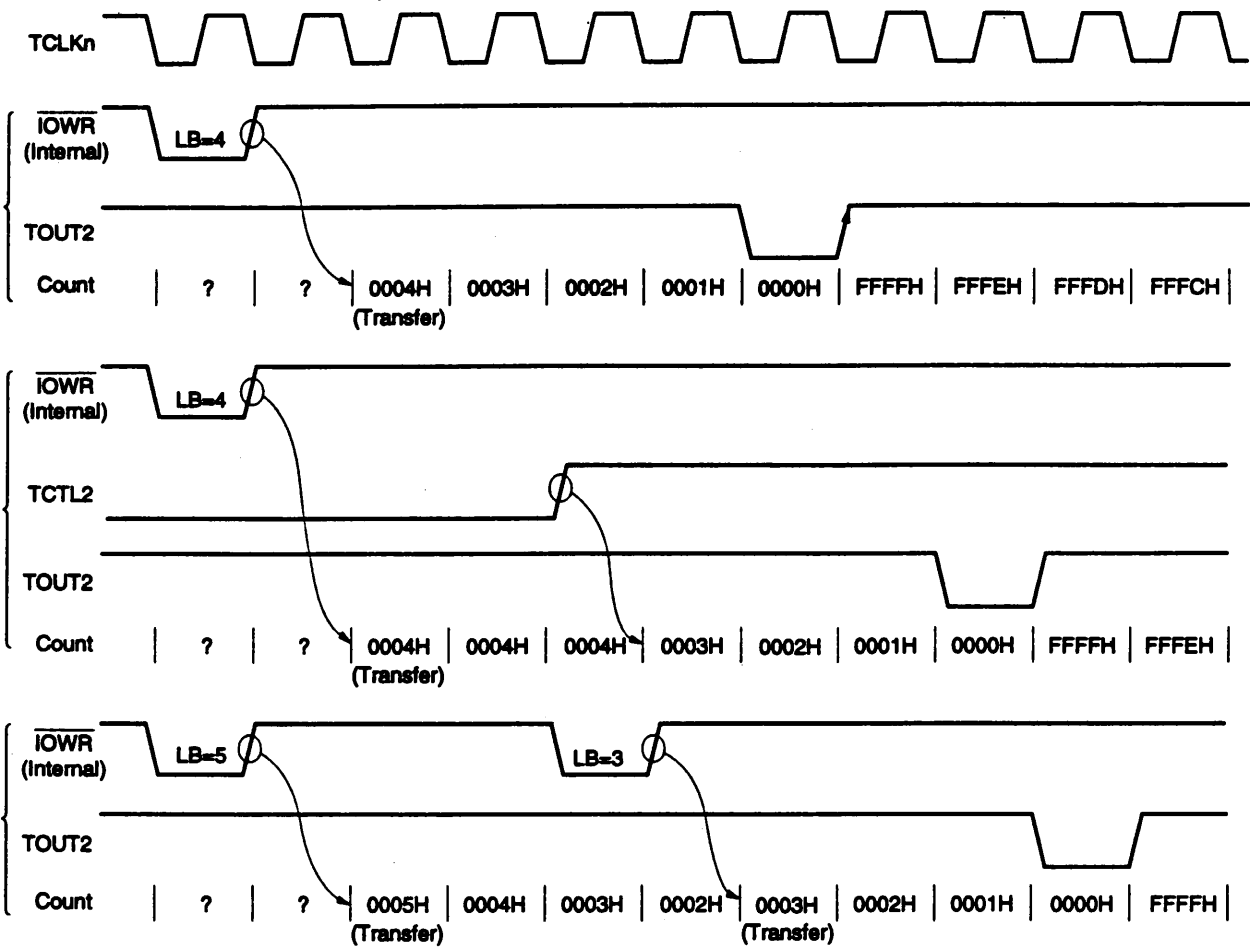
(5) Mode 4: Software-Triggered Strobe

When the specified count has completed, the TOUT pin will go low for 1TCLKn pulse. The function takes place only once per one count transfer.

Table 5-12. Mode 4 Operation

Item		Description
TOUT initial condition		High level
TCTL input	High level	Count state
	Low level	The count prohibited state.
Writing the count		When the count is written, it will be transferred at the next TCLKn pulse. When read/write is in the 2-byte mode, the above operation will take place when the second byte is written.
Count transfer and count		Transfer takes place at the TCLKn pulse after the count is written. If TCTL is high, decrementing will begin at the next TCLKn pulse. If TCTL is low, decrementing will begin at the TCLKn pulse after TCTL goes high.
Count zero		The TOUT pin is low during 1TCLKn and then returns to high. Without stopping the count operation, if binary, the count down will take place to FFFFH, if BCD, it will take place to 9999.
Minimum count		1

Figure 5-25. Mode 4 Operation Example



(6) Mode 5: Hardware-Triggered Strobe (Retriggerable)

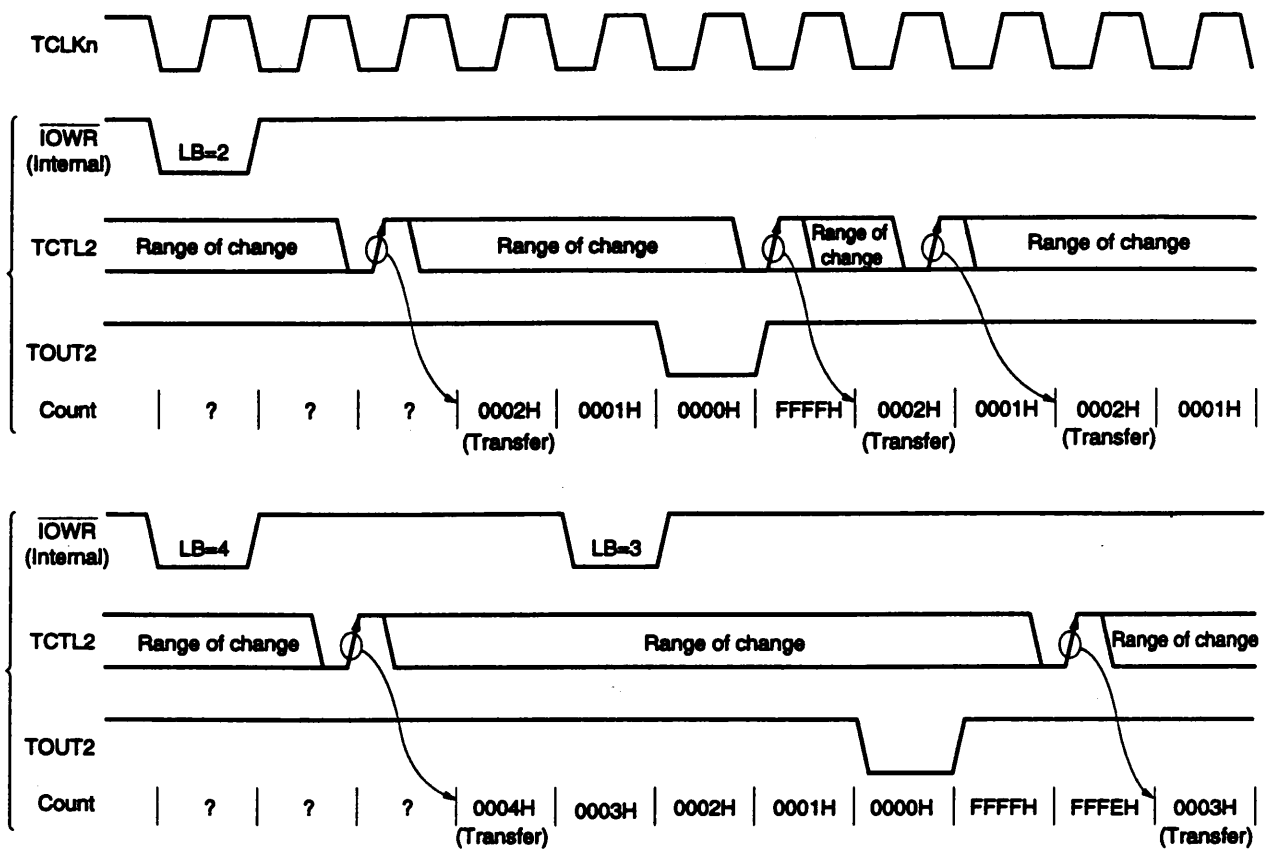
The operation of mode 5 is the same as that of mode 4, but this mode can be started up by TCTL input and is retriggerable. This function operates once per one count transfer.

Table 5-13. Mode 5 Operation

Item		Description
TOUT initial condition		High level
TCTL Input	Trigger ^{Note}	Count transfer takes place at the TCLKn pulse after the trigger.
Writing the count		Writes without influencing the current operation.
Count transfer and count		When there is a trigger, transfer takes place at the next TCLKn pulse. Decrementing starts at the TCLKn pulse after the transfer. Thus, if the count is N, the TOUT pin is not low during the period from the trigger to N+1TCLKn pulses.
Count zero		The TOUT pin is low during 1TCLKn and then returns to high. Without stopping the count operation, if binary, the count down will take place to FFFFH, if BCD, it will take place to 9999.
Minimum count		1

Note The trigger will be ignored if the count has not been written right after the mode specification and if only one byte has been written in the read/write two-byte mode.

Figure 5-26. Mode 5 Operation Example



5.7.7 Precautions

The following precautions apply to switching the count mode.

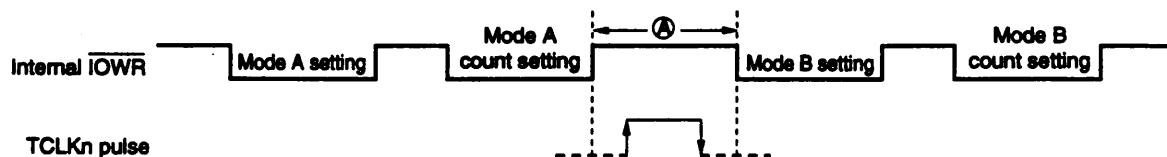
The TCLKn pulse is required to switch a counter set to a desired mode to another mode (mode 2 or 3 only).

Input the TCLKn pulse for one pulse or more (a rising pulse that will fall next) between the count setting of the prior mode and that of the new mode setting.

If this pulse is not entered, the operation of the new mode may make an error in the count.

Switch the mode by using one of the following methods:

- Use a cyclical clock that will enter a minimum of one pulse in the interval indicated by (A).
- Delay the new mode setting so that clock used will have a minimum of one rise and fall.



Remark Mode A: an optional mode.

Mode B: mode 2 or mode 3

5.8 Serial Control Unit (SCU)

The V40HL and V50HL systems can easily conduct serial communications (start-stop synchronous) using the SCU. The command system is similar to that of the μ PD71051 (except for lack of synchronous communication protocols). However, what was the control word register in the μ PD71051 has been divided up into two registers in the V40HL and V50HL: the serial command register (SCM) and the serial mode register (SMD). These registers make the V40HL and V50HL easier to use.

The SCU has only three pins, RxD input (serial receive), TxD output (serial transmit), and $\overline{\text{SRDY}}$ (data transmit ready). Therefore, for handling functions such as RS-232C, more peripheral devices are necessary.

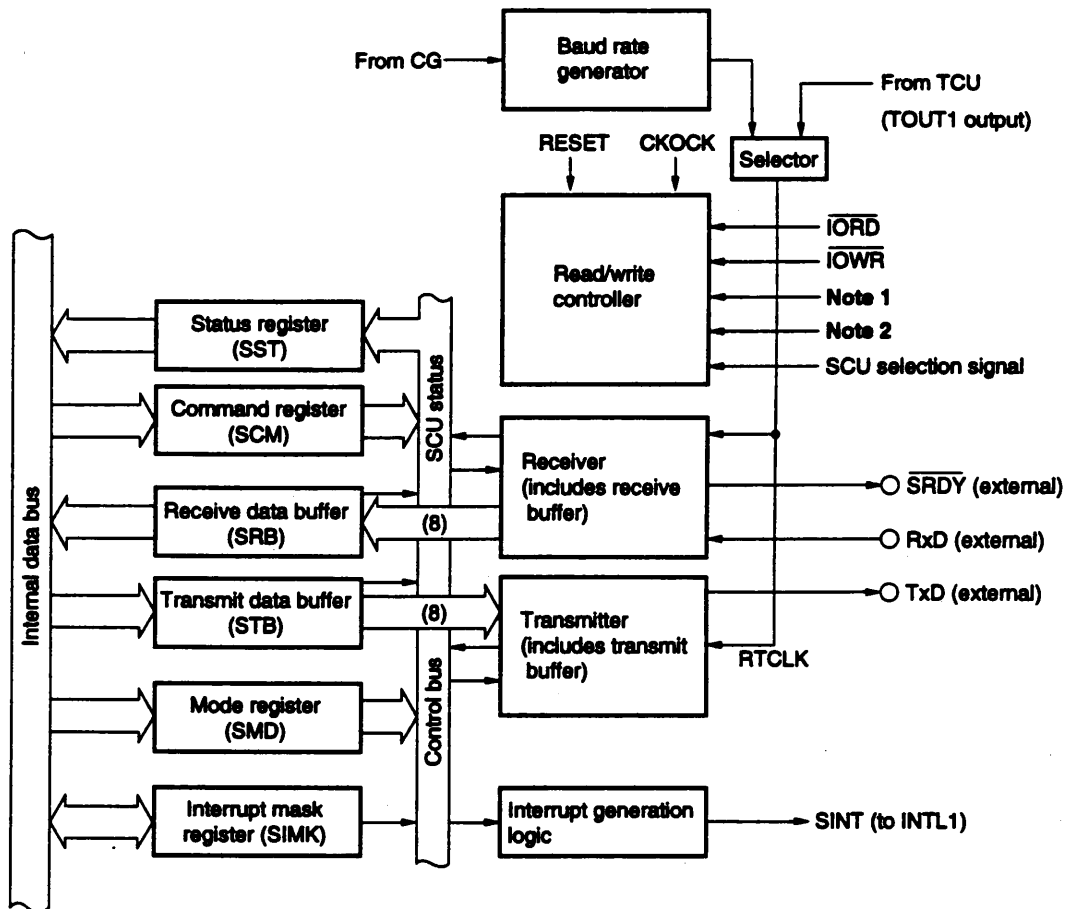
5.8.1 Features

- Internal dedicated baud rate generator (uses internal clock)
- Baud rate generator output or timer output can be selected as the transmit/receive clock
- Asynchronous serial communications
- Clock rate and baud rate divisor: 16 and 64
- Baud rate of DC-500 Kbits per second
- 7- and 8-bit character lengths
- 1- and 2-bit transmission stop bit lengths
- Break transmission
- Automatic break detection
- Full-duplex double-buffered communication
- Parity add/check
- Parity, overrun, and framing error detection
- Maskable interrupt generation

5.8.2 Differences with the V40 and V50

In order to increase the general-purpose characteristics of the TCU, it has an internal dedicated baud rate generator. In addition, to maintain compatibility with the V40 and V50, the transmit/receive clock can be selected from the internal timer output and the baud rate generator output so that the baud rate can be defined by TCLKn input.

5.8.3 Internal block diagram



Notes 1. A0 (IOAG = 1) or A1 (IOAG = 0)

2. A1 (IOAG = 1) or A2 (IOAG = 0)

5.8.4 Addressing

The SCU internal registers are mapped to addresses determined by the OPHA, SULA and SCTL IOAG bits that are defined in the system I/O area. (For the methods of pointing to an address, refer to section 4.1.3 System I/O area.

The BRC address is fixed at FFE9H in the system I/O area and cannot be relocated by the content of OPHA or SULA.

Table 5-14. SCU Internal Register and Command Addresses

<div></div>	High-Order Addresses	Low-Order Addresses						Registers/ Commands		Operation
When IOAG = 1	A15 - A8 (OPHA)	A7 A6 A5 A4 A3 A2 (SULA)	0 0		SRB		Read			
							STB		Write	
			0 1		SST				Read	
							SCM		Write	
			1 0		SMD				Write	
							1 1		SIMK	
When IOAG = 0	A15 - A8 (OPHA)	A7 A6 A5 A4 A3 (SULA)	0 0		A0					
							STB		Write	
			0 1						SST	
							SCM			
			1 0						SMD	
							1 1			
—	1 1 1 1 1 1 1 1	1 1 1 0 1 0 0 1	BRC		Read/write					

Remark The BRC address is FFE9H regardless of the IOAG bit and the content of OPHA and SULA.

5.8.5 Initialization

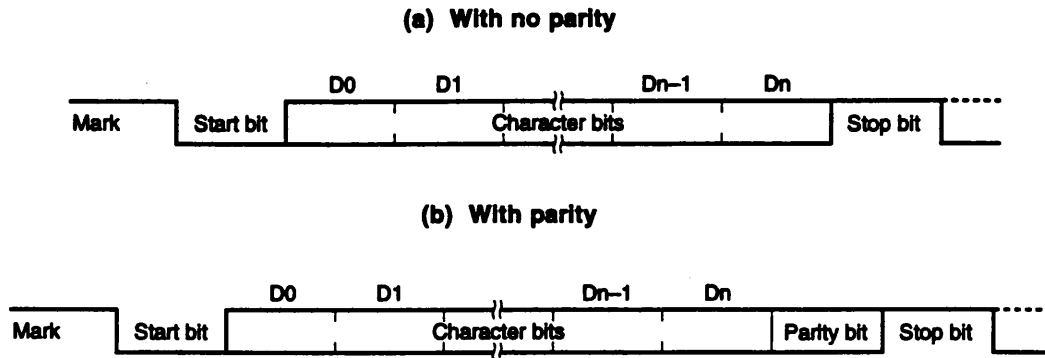
SCU is initialized to the following conditions when the power is turned on or when there is a reset signal:

- Baud rate factor : x64
- Character length: 7 bits
- Parity bit : none
- Stop bit : 1 bit
- Transmit/receive prohibited
- No break detection
- No errors
- Buffer not ready

5.8.6 Serial data format

Figure 5-27 shows the serial data format handled by SCU. Data the CPU receives from SCU and passes to SCU is that portion called character bits. The start bit, parity bit and stop bit which are on either side of the character bits, are control data needed to conduct serial communications. SCU automatically appends (for transmission) and deletes (for reception) this data.

Figure 5-27. Serial Data Format



n: 6 or 7

Stop bit: 1 bit or 2 bits

Each data state will be described below.

(1) Mark (high level)

In a state in which no data transmission is taking place, the TxD pin is at the high level (mark state).

(2) Start bit (low level)

This is a one-bit-length, low-level bit that indicates the start of serial data.

(3) Character bits

For both transmission and reception, this is the portion that is actually handled as data. It is defined as 7 or 8 bits.

(4) Parity bit

When parity is enabled, among the bits that match the character bits and the parity bit, the parity bit is appended or checked so that the number of certain bits is even by adding one (even parity) or odd by adding one (odd parity).

Example	Characters (8 bits)	Parity bit
Even parity	10100101	0
Odd parity	10100101	1

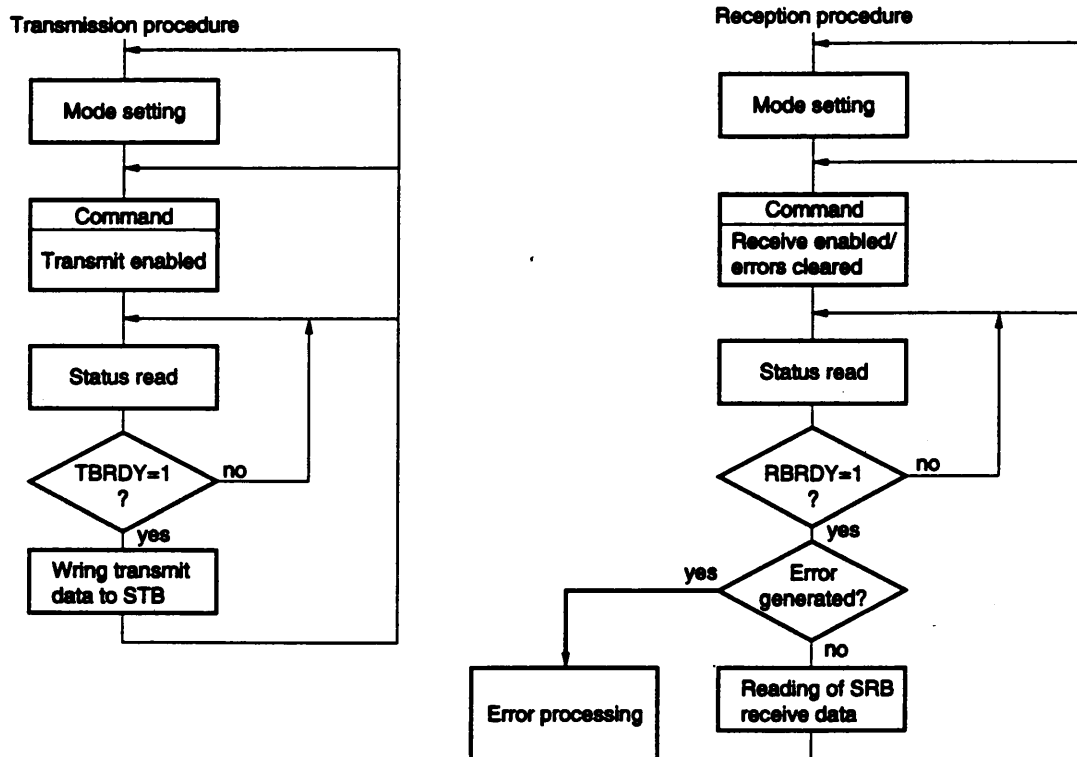
(5) Stop bit

The stop bit is in "1" status of the 1- or 2-bit length that indicates the end of serial data. The serial mode register (SMD) is used to specify 1-bit length or 2-bit length.

5.8.7 Operating procedures

Figure 5-28 shows the operating procedures for conduction serial communication using the SCU.

Figure 5-28. SCU Operating Procedures



To conduct a transmission, after being transmit enabled, transmission data is written to STB after the status is used to confirm that the transmit data buffer (STB) is empty.

To conduct a reception, reception is enabled and at the same time the error flags in the status are cleared. Next, the status is used to confirm that receive data is stored in the receive data buffer (SRB). Then the data is read from SRB.

In Figure 5-28, the status is read and transmission and reception take place while the TBRDY and RBDY flags are being looked at. However, this can also take place using interrupts.

5.8.8 Registers and commands

The SCU register reading or writing, and command issuance take place using I/O instructions for addresses that are defined in the system I/O area. Register commands are selected using A1 and A0 (IOAG = 1) or A2 and A1 (IOAG = 0), respectively.

Table 5-15. SCU Register and Command Addresses

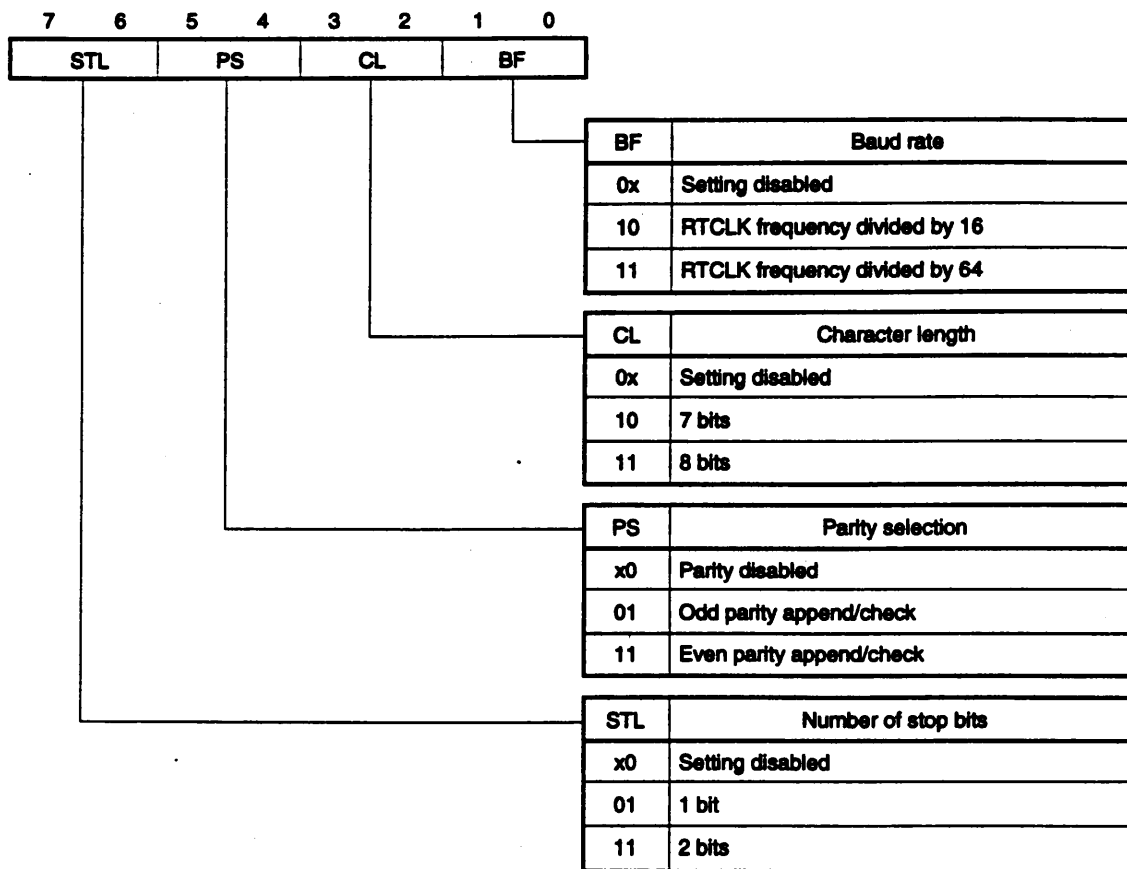
When IOAG = 0		When IOAG = 1		Registers and Commands	Operation
A2	A1	A1	A0		
0	0	0	0	SRB	Read
				STB	Write
0	1	0	1	SST	Read
				SCM	Write
1	0	1	0	SMD	Write
1	1	1	1	SIMK	Read/write
Note				BRC	Read/write

Note The BRC address is fixed at FFE9H.

- Serial receive data buffer (SRB) : The CPU reads receive data from this buffer.
- Serial transmit data buffer (STB) : The CPU writes transmit data to this buffer.
- Serial status register (SST) : This register indicates the communication status. It has SRB and STB data and reception error data.
- Serial command register (SCM) : This register controls the enabling and disabling of transmission and reception, the clearing of error flags, break transmissions and the masking of $\overline{\text{SRDY}}$ pin.
- Serial mode register (SMD) : This register makes the setting of the baud rate generator, character length, parity, and the number of stop bits. When the character length is seven bits, the lower seven bits of SRB and STB become valid.
- Serial interrupt masking register (SIMK): This register controls the masking of the interrupt requests generated by SCU. When masked, interrupts will no longer be generated.
- Baud rate counter (BRC) : The is an 8-bit frequency divider counter for the SCU dedicated baud rate generator.

(1) Serial Mode Register (SMD)

Figure 5-29. Serial Mode Register (SMD)



Remark x: don't care

(a) STL Bit

This bit sets the number of stop bits to one or two during transmission.

(b) PS Bit

This bit defines the parity bit. If parity is disabled, no parity append will take place during transmission nor parity check during reception. If parity is enabled, either odd parity or even parity must be selected.

(c) CL Bit

This bit defines the number of bits in one character. The selection is between seven bits or eight bits. If seven bits are selected, SCU will receive the low-order seven bits of the 8-bit data written by the CPU. If the data that SCU outputs to the CPU, the high-order 1 bit is set to zero.

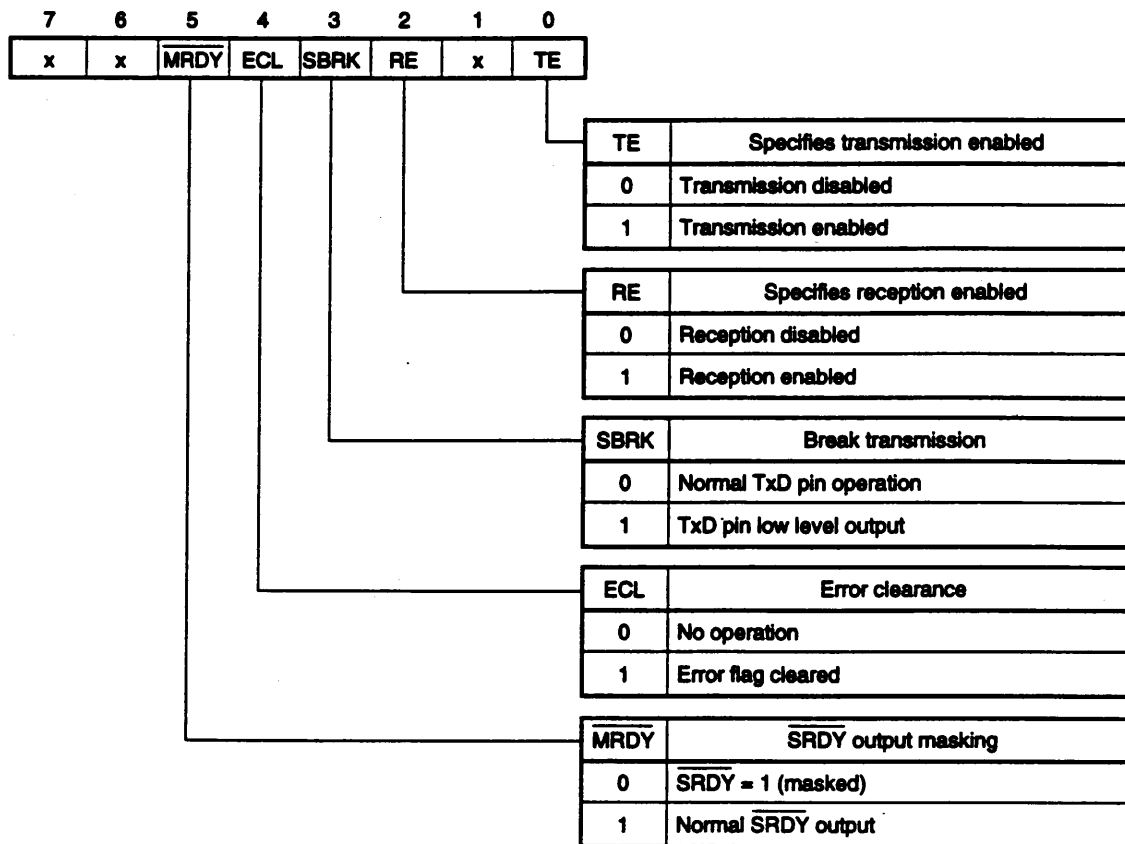
(d) BF Bit

This bit determines the relationship between the RTCLK frequency and the baud rate. It determines if the baud rate for the transmit/receive clock will be divided by 16 or 64.

For information on the setting the baud rate, see section 5.8.9 Transmission and reception baud rates.

(2) Serial Command Register (SCM)

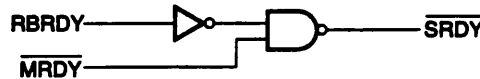
Figure 5-30. Serial Command Register (SCM)



Remark x: don't care.

(a) $\overline{\text{MRDY}}$ Bit

This bit controls the masking of the $\overline{\text{SRDY}}$ pin output. The $\overline{\text{SRDY}}$ output is as shown in Figure 5-31. When $\overline{\text{MRDY}} = 0$, it always outputs the high level and disables data transmission from the transmitting side. When $\overline{\text{MRDY}} = 1$, it outputs the RBRDY signal. (Refer to the information on the serial status register (SST) in number (3).)

Figure 5-31. $\overline{\text{SRDY}}$ Signal**(b) ECL Bit**

If the ECL bit is set to 1, all of the error flags in the status register (PE, OVE, FE) will be cleared (0). In particular, to enable reception, it is required that ECL = 1 to clear the error flags.

(c) SBRK Bit

This bit is used for a break state transmission. In a break transmission, the low level is output to the TxD pin irrespective of the data. Also, the break transmission function is valid even in the transmit disabled state.

(d) RE Bit

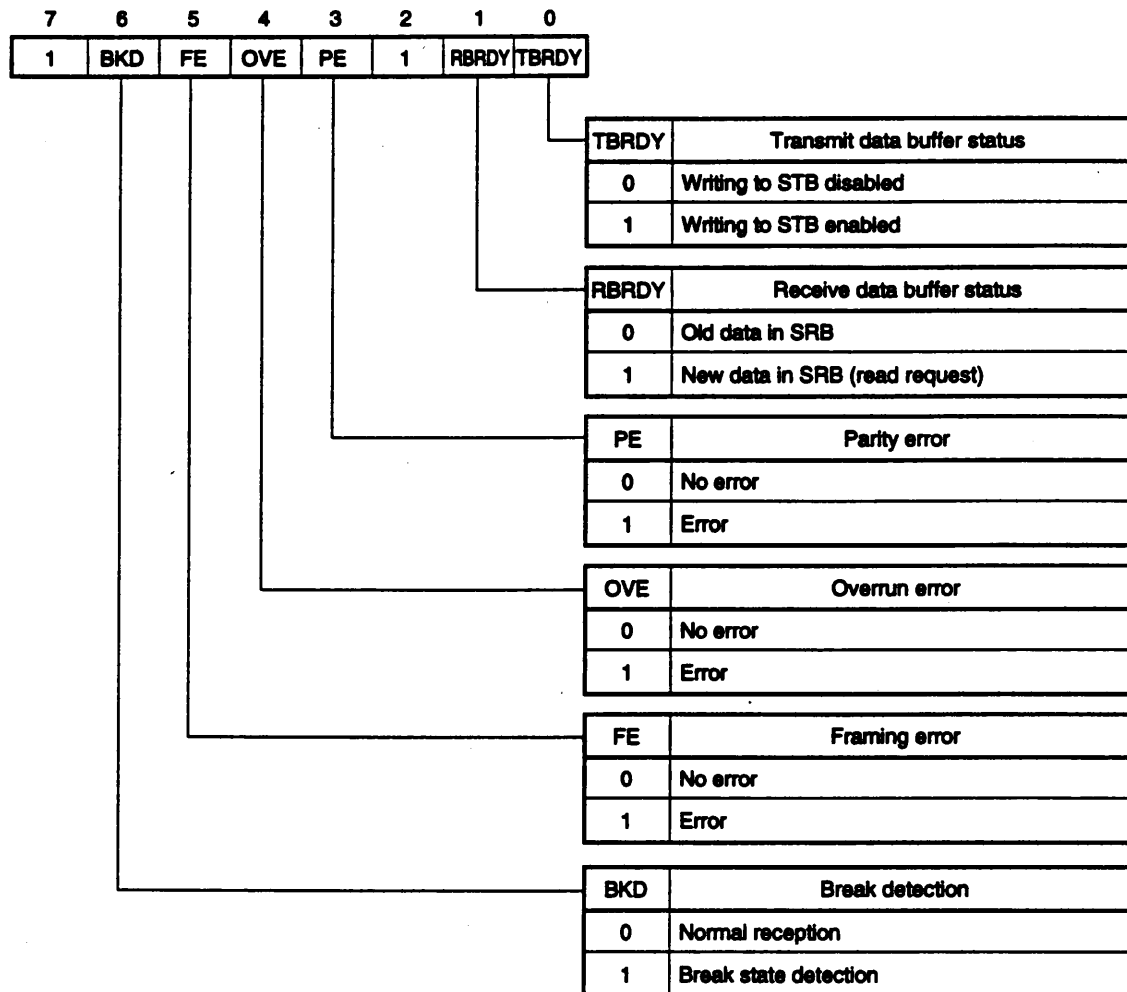
This bit controls the enabling and disabling of reception.

(e) TE Bit

This bit controls the enabling and disabling of transmission. When transmission is disabled, if data is in the transmit buffer at the point that TE is set to zero, that data will not be transmitted. It will be retained in the buffer. Subsequently, when the transmission is enabled again, the data being retained in the buffer will be transmitted. In the transmit disabled state, the TxD pin is high (mark state).

(3) Serial Status Register (SST)

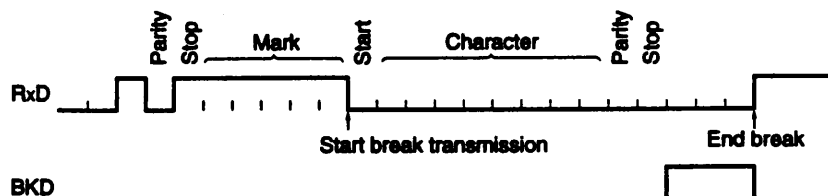
Figure 5-32. Serial Status Register (SST)



(a) BKD Bit

The bit sets to 1 when the break status is detected. That is, when the low level is input for all bits from the start bit to the stop bit, the break state is generated. Once the BKD bit is set (to 1), it will not be cleared (set to 0) until the RxD returns to high or the break is reset. Figure 5-33 shows the operation of the BKD bit as it relates to RxD.

Figure 5-33. Break Input and Detection



(b) FE Bit

This bit becomes 1 if the high level is not detected when a stop bit is required and indicates a framing error. In addition to being generated when there is reception in the break state, a framing error is also generated when the transmission clock and reception clock go out of synchronization and when data in a transmission path changes.

(c) OVE Bit

During data reception, this bit becomes 1 if the CPU is late in reading the receive data and generates an overrun error. When this happens, the old data in SRB will be lost due to the writing of the new data.

(d) PE Bit

When receiving data and parity is enabled, this bit will become 1 if an error is generated during a parity check.

(e) RBRDY Bit

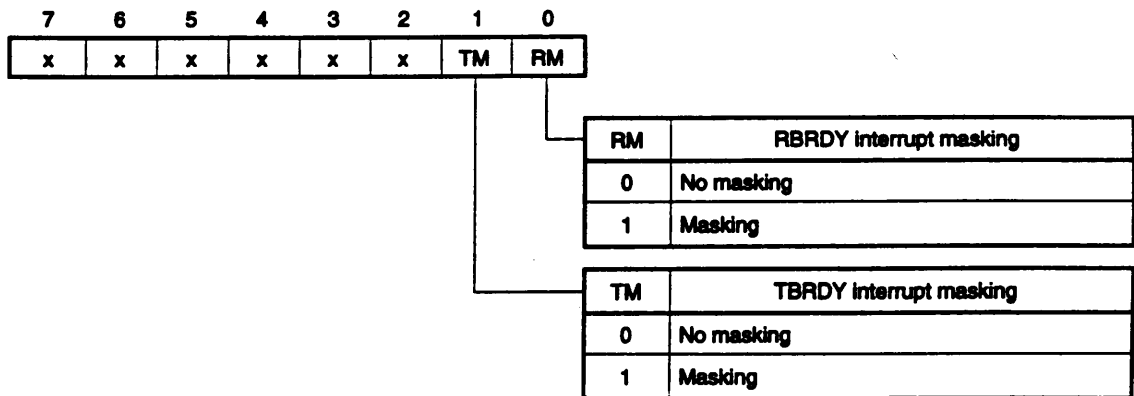
When a data of one character is received and that data is transferred to SRB, which is to say that when the CPU is enabled for reading receive data, this bit becomes 1. When the CPU reads SRB, the RBRDY bit is cleared (0).

(f) TBRDY Bit

This bit becomes 1 when STB is empty, that is, when the CPU is enabled to write data to STB. When the CPU writes data to STB, the TBRDY bit is cleared (set to 0). If transmit data is written when TBRDY = 0, the data within STB that has not been transmitted will be destroyed. In addition, in the transmit disabled state, the TBRDY bit will not become 1.

(4) Serial Interrupt Masking Register (SIMK)

Figure 5-34. Serial Interrupt Masking Register (SIMK)



Remark A small x refers to an optional 1 or 0.

This register independently masks the RBRDY bit and the TBRDY bit, which are the causes of interrupts requests from SCU.

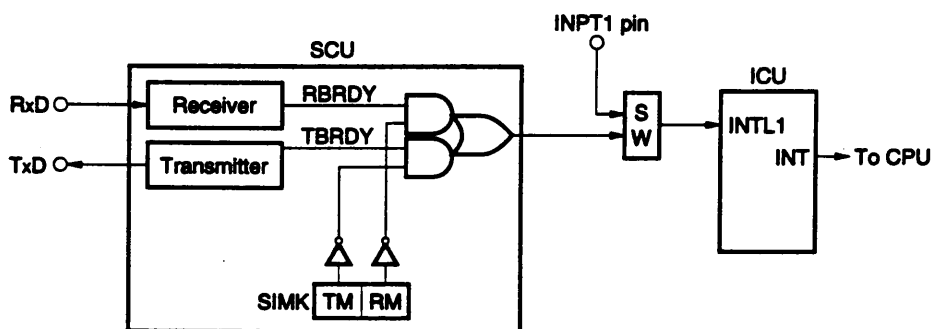
(a) TM Bit

If the TM bit is set to 1, even if TBRDY = 1, no interrupt will be generated.

(b) RM Bit

If the RM bit is set to 1, even if RBRDY = 1, no interrupt will be generated.

Figure 5-35. Relationship Between the TM Bit, RM bit and INTL1



5.8.9 Transmission and reception baud rates

The SCU transmit/receive baud rate is determined by the transmit/receive clock signal (RTCLK) and the BF bit of SMD.

The RTCLK signal can be selected from the output of the dedicated baud rate generator and the output of TOUT1 of the TCU. This is specified using the SC bit of SCTL in the system I/O area. (See 4.1.3 System I/O Area.)

When using the TOUT1 output, set the TCU TCT#1 operating mode to the mode 3 square wave generator. (See 5.7.6 Count Mode.)

(1) Baud Rate Counter (BRC)

The baud rate counter (BRC) is an 8-bit frequency divider counter for the SCU dedicated baud rate generator. It sets the frequency divisor of the internal clock (fixed at 1/2 of the oscillating frequency). BRC is assigned to address FFE9H of the system I/O area.

Figure 5-36. Baud Rate Counter (BRC)

7	6	5	4	3	2	1	0	
D7	D6	D5	D4	D3	D2	D1	D0	I/O address
								FFE9H

(2) Baud Rate Setting

The baud rate is determined by the following equation:

(a) When using the baud rate generator

$$\text{Baud rate} = \frac{\text{Oscillating frequency (Hz)}}{\text{BF} \times \text{BRC setting}} \times \frac{1}{2}$$

Remark The relationship between the BRC register and the BRC setting is shown in the table below.

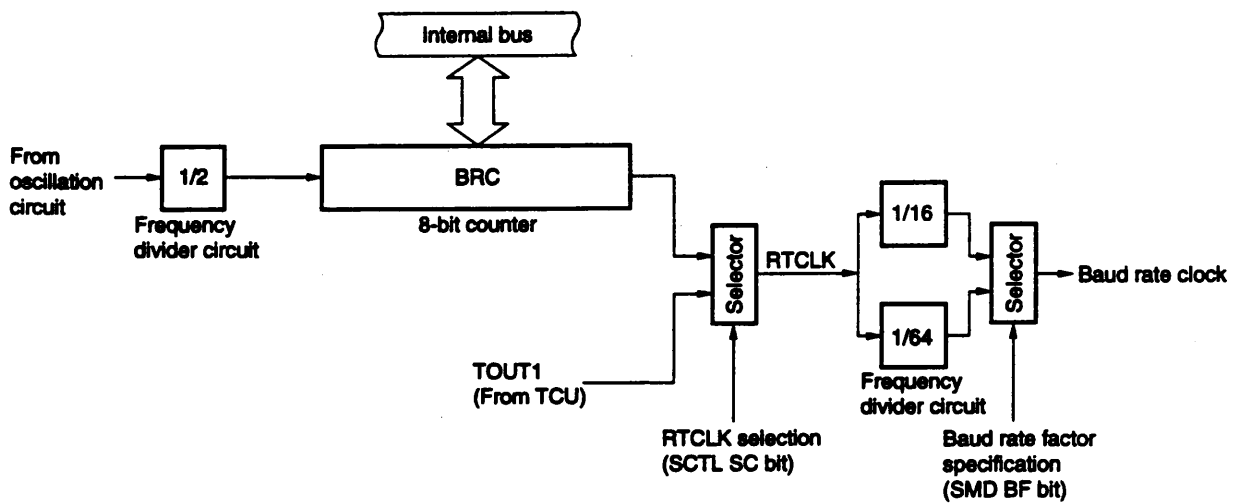
BRC Register	BRC Setting
00H	2
01H	2
02H	2
03H	3
04H	4
⋮	⋮
FFH	255

(b) When Using TCU

$$\text{Baud rate} = \frac{\text{TCLK input frequency (Hz)}}{\text{BF} \times \text{TCT\#1 setting}}$$

Remark The baud rate factor (BF) is specified using the SMD BF bit and is either 16 or 64.

Figure 5-37. Baud Rate Clock Generation Diagram



(3) Example of Baud Rate Settings

Tables 5-16, 1 of 2 and 2 of 2, show examples of baud rate settings when the baud rate generator is used and when the TOUT1 output is used.

Table 5-16. Baud Rate Settings (1 of 2)

(a) Baud Rate Generator

Oscillating Frequency (X1, X2)	24.576 MHz		29.4912 MHz	
Baud rate factor	16	64	16	64
Baud rate	BRC count			
1200	—	160	—	192
2400	—	80	—	96
4800	160	40	192	48
9600	80	20	96	24
19200	40	10	48	12
38400	20	5	24	6

Table 5-16. Baud Rate Settings (2 of 2)

(b) TOUT1 Output

TCT#1 Input Clock (MHz)	4.9152 ^{Note}		7.3728 ^{Note}	
Baud rate factor RTCLK/baud rate	16	64	16	64
Baud rate	TCT#1 count			
110	2793	698	4189	1047
150	2048	512	3072	768
300	1024	256	1536	384
600	512	128	768	192
1200	256	64	384	96
2400	128	32	192	48
4800	64	16	96	24
9600	32	8	48	12
19200	16	4	24	6
38400	8	—	12	—

Note External input clock

Remark TCT#1: set to mode 3

(4) Baud Rate Setting Precautions

SCU inputs the transmit/receive clock (RTCLK) either from the dedicated baud rate generator or from TCU. However, the maximum baud rate is determined by the internal clock, as indicated in Table 5-17.

Table 5-17. Internal Clock and Maximum Baud Rate

Internal Clock	Maximum Baud Rate
20 MHz	625 Kbps
16 MHz	500 Kbps
8 MHz	250 Kbps
4 MHz	125 Kbps
2 MHz	62.5 Kbps

*

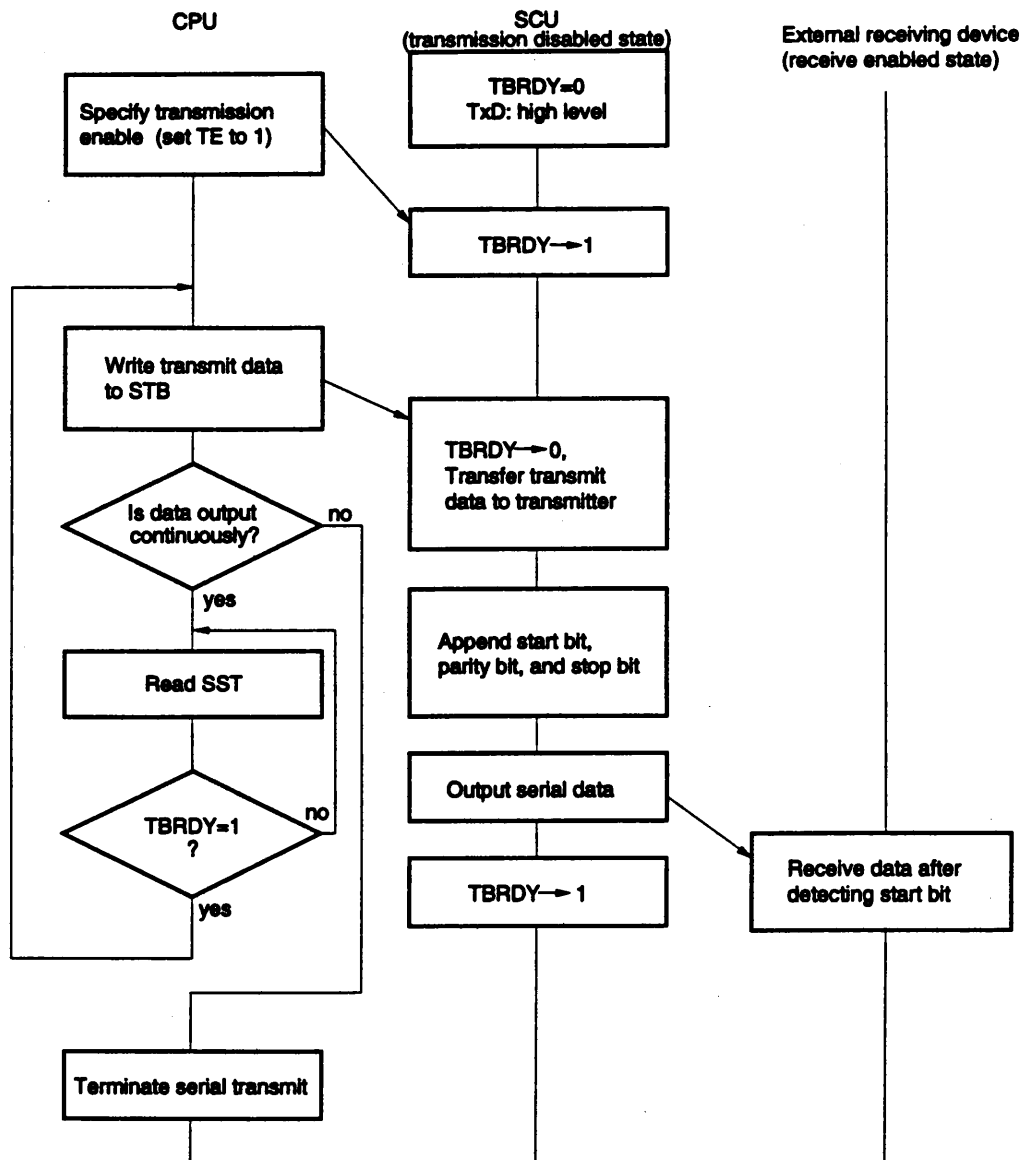
5.8.10 Serial transmission and reception

The basic flow of the serial data transmission and reception operations due to SCU will be described in this section.

(1) Serial Transmission Operation

- (a) In the transmit disabled state, the TxD pin is high (mark state), except during a break transmission, and the TBRDY bit is always zero in this situation.
- (b) When SCM generates the transmit enabled state, TBRDY will equal 1 and the writing of transmit data can take place.
- (c) When the CPU writes transmit data to STB, that data will first be transferred to the transmitter, a start bit, parity bit, and stop bit will be appended to it, and it will be transmitted from the TxD pin as serial data.
- (d) When data is to be transmitted continuously, the transmit data will be written after it has been confirmed that TBRDY = 1. When TBRDY = 0, the previous data is still in the transmit buffer. Therefore, the following data must not be written to it yet.

Figure 5-38. Serial Transmit Operation



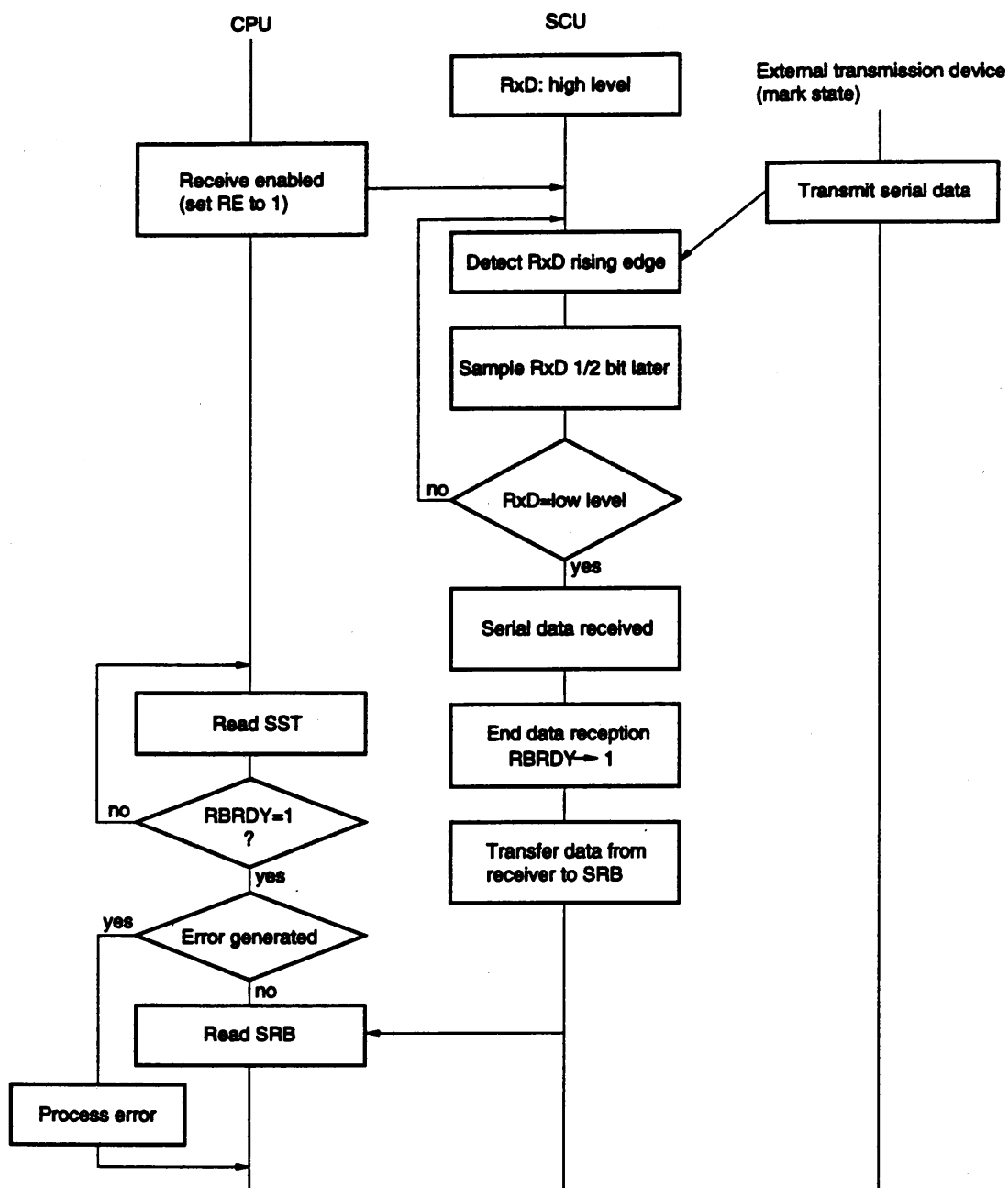
Remark TE is SCM bit 0, while TBRDY is SST bit 0.

(2) Serial Receive Operation

- (a) The RxD pin must be high when it is not receiving data.
- (b) If the low level is detected on the RxD pin in the receive enabled state, the receiver will sample the RxD input level at the location 1/2 bit from the fall to determine if this is a valid start bit. If the low level is detected at this time, it will be considered a valid start bit and reception will begin. Conversely, if the high level is detected, it will not be considered the start bit and the receiver will once again return to the low-level wait state.
- (c) After the start bit has been detected, character bits, parity bit, and stop bit will be sampled and data will be received.
- (d) When a data of one character has been received, the data will be sent from the receiver to SRB. Then RBRDY becomes one, the CPU will be requested to read the data.
- (e) If the CPU does not read data in the state in which RBRDY equals one, the next data will be sent to SRB and the prior data will be deleted. When this happens, OVE will equal 1 to indicate that an overrun error has occurred.
- (f) When parity is valid, a parity check will take place. If an error is detected, PE will equal one.
- (g) If the high level is detected when there should be a stop bit, there will be a wait for the start bit. But, if the high level is not detected, FE will equal 1 to indicate that a framing error has occurred.

No matter what kind of error occurs, the reception operation will not end while data is being received. Therefore, when an error occurs, it must be processed by the software.

Figure 5-39. Serial Receive Operation



Remark RE is SCM bit 1, while RBDY is SST bit 1.

5.8.11 Precautions

When using ICU in the edge trigger mode, be careful of the following concerning the SCU transmit/receive interrupt.

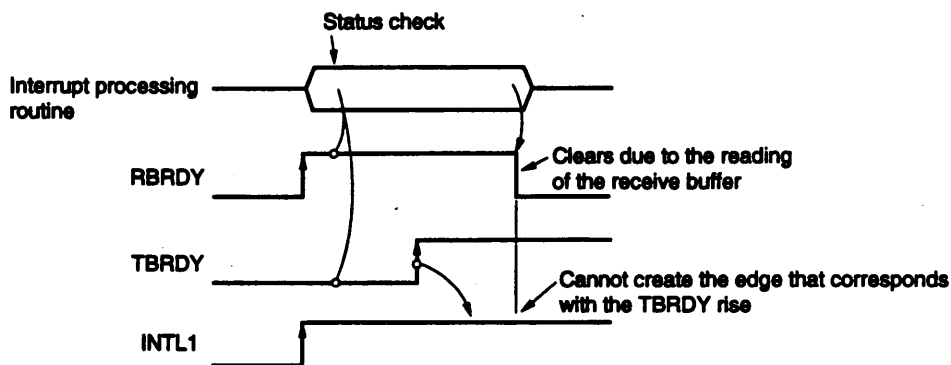
SCU ORs the RBRDY signal which indicates a completed reception, and the TBRDY signal which indicates a completed transmission, and inputs the result to ICU. (See Figure 5-35 Relationship Between the TM and RM bits and INTL1.) Therefore, when using a transmit or receive interrupt under the following conditions and timing, at times the edge of an interrupt generated after RBRDY and TBRDY became active will not be detected and interrupts will not be accepted.

Conditions: ICU = edge trigger mode

SIMK RM bit = 0 (RBRDY not masked)

SIMK TM bit = 0 (TBRDY not masked)

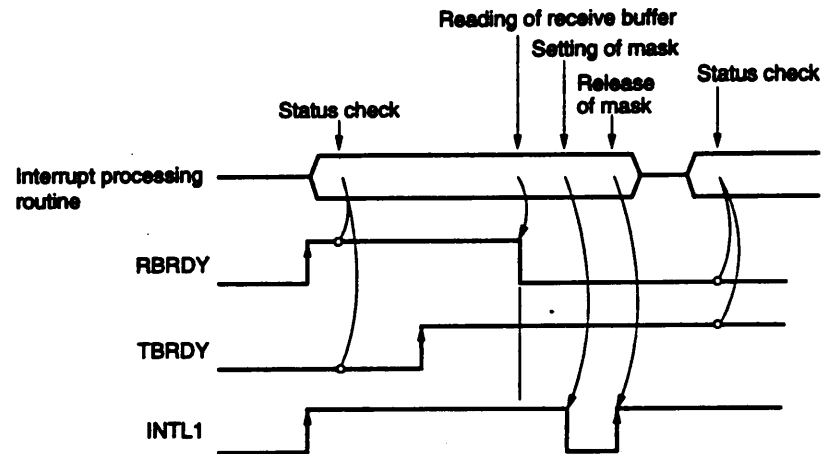
Timing:



To resolve the above condition, use one of the following methods:

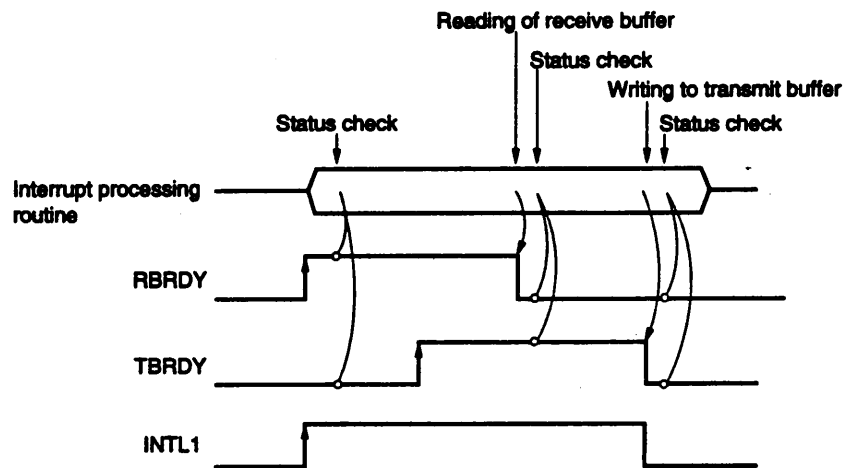
(1) Resolution through edge generation

Process transmission or reception in the serial transmit/receive interrupt processing routine and clear the RBRDY or TBRDY signal that caused the interrupt. Next, set SIMK to 1 to mask the interrupt request and then clear it to zero to release the mask.



(2) Resolution using the status check

Perform a status check at the end of the processing of the interrupt request first detected to confirm that neither a transmission nor a reception interrupt request has been generated.



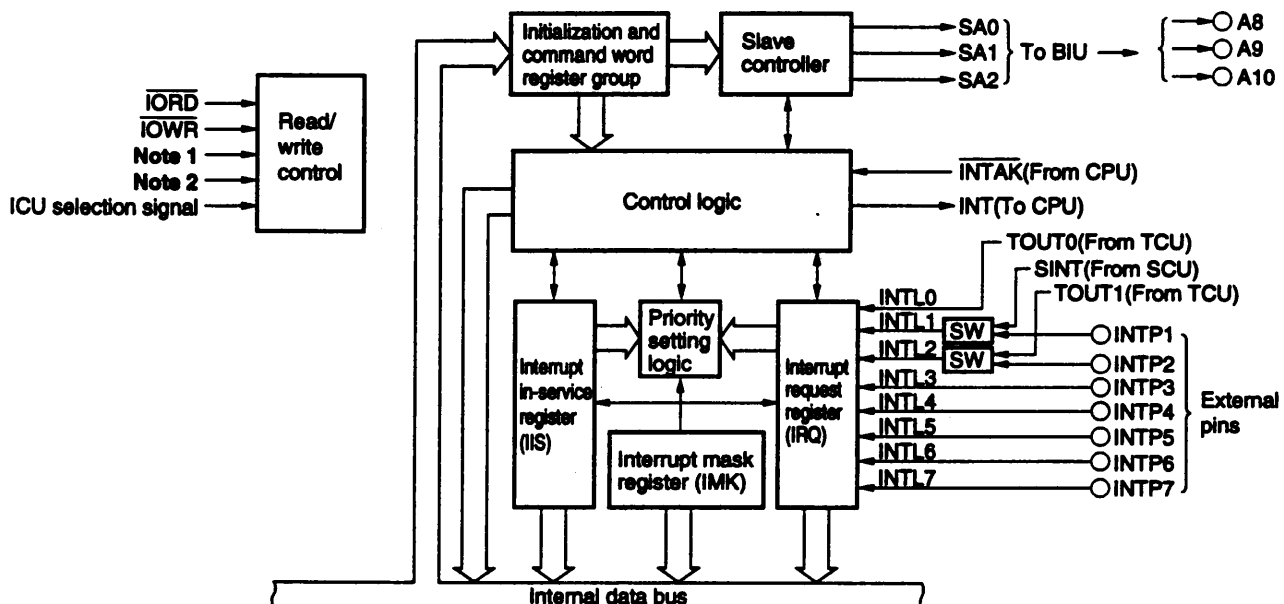
5.9 Interrupt Control Unit (ICU)

The interrupt control unit (ICU) arbitrates up to eight interrupt requests generated within and without the V40HL and V50HL. One among the interrupts is sent to the CPU. The ICU functions are those in which some of the unnecessary functions of the μ PD71059 have been eliminated (the CALL mode and slave functions for cascade connection when using the μ PD8085A).

5.9.1 Features

- Eight interrupt request inputs
- Able to cascade-connect the μ PD71059.
- Edge or level trigger request input (The input from internally connected TCU is edge-triggered only.)
- Interrupt requests are individually maskable.
- Programmable interrupt request priorities
- Pollable operations

5.9.2 Internal block diagram



- Notes**
1. A0 (when IOAG = 1) or A1 (when IOAG = 0)
 2. A1 (when IOAG = 1) or A2 (when IOAG = 0)

5.9.3 Addressing

The ICU internal registers are mapped to the addresses determined by the IOAG bit of OPHA, IULA and SCTL, which are defined in the system I/O area. (For the address specification method, see section 4.1.3 System I/O Area.) For A2 when IOAG = 0 or A1 when IOAG = 1, in either case, IOAG may either equal 1 or 0.

Table 5-18. ICU Internal Register and Command Addresses

	High-order Addresses	Low-order Addresses				Registers/ Commands	Operation
When IOAG = 1	A15 - A8 (OPHA)	A7 A6 A5 A4 A3 A2	(IULA)	x	0	IRQ/IIS/IPOL	Read
						IIW1/IPFW/IMDW	Write
				x	1	IMKW	Read/write
						IIW2/IIW3/IIW4	Write
When IOAG = 0	A15 - A8 (OPHA)	A7 A6 A5 A4 A3	(IULA)	x	0	A0 IRQ/IIS/IPOL	Read
						A0 IIW1/IPFW/IMDW	Write
				x	1	IMKW	Read/write
						A0 IIW2/IIW3/IIW4	Write

Remark x: don't care

Because there is only one signal for a multiple of registers, ICU requires a fixed sequence for reading from and writing to registers. For example, when initializing, ICU selects one of the four initialization sequences to match the objective and writes to the registers from interrupt initialization word 1 register to interrupt initialization word 4 register (IIW1 to IIW4).

As write enabled registers, there are the interrupt mask word register (IMKW), interrupt priority/finish word register (IPFW), and the interrupt mode word register (IMDW). IMKW can be written to as A0 = 1 (when IOAG = 1) or as A1 = 1 (when IOAG = 0) after the initialization sequence. However, because both IPFW and IMDW are in a state in which A0 = 0 (when IOAG = 1) or A1 = 0 (when IOAG = 0), they are distinguished by the values of bit 3 and bit 4 of the data to be written to them.

Among the registers, there are three that can be read: interrupt request register (IRQ), interrupt in-service register (IIS), and IMKW. With A0 = 1 (when IOAG = 1) or A1 = 1 (when IOAG = 0), IMKW can be read at any time. However, both IRQ and IIS are A0 = 0 (when IOAG = 1) or A1 = 0 (IOAG = 0). Thus, before they are read, IMDW must be used to specify in advance which will be read.

The ICU register and command addresses are listed in Table 5-19.

Table 5-19. ICU Register and Command Addresses

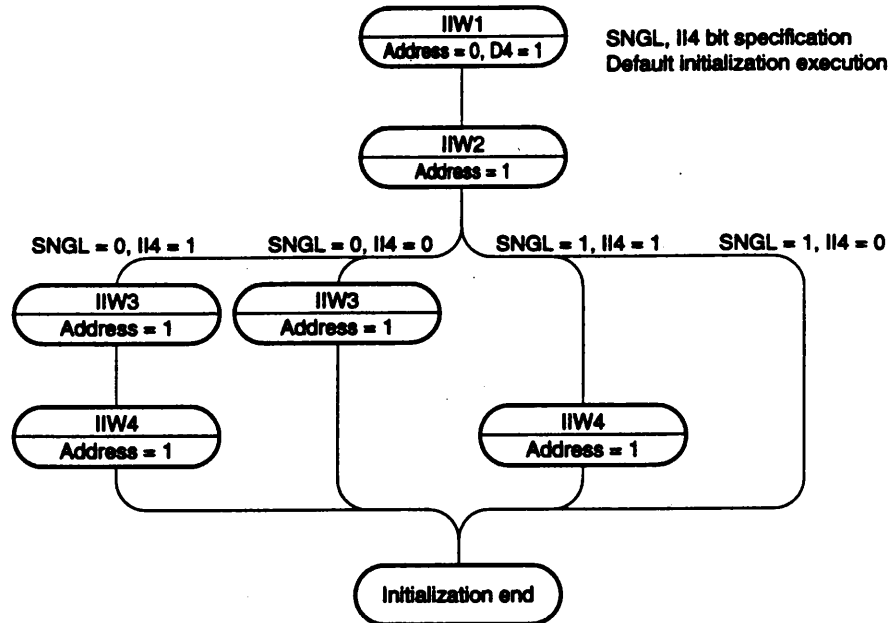
	IOAG = 0		IOAG = 1		Operation
	A1	Non-A1 condition	A0	Non-A0 condition	
Read	0	IRQ defined using IMDW	0	IRQ defined using IMDW	CPU ← IRQ data
		IIS defined using IMDW		IIS defined using IMDW	CPU ← IIS data
		Polling phase ^{Note}		Polling phase ^{Note}	CPU ← IPOL
	1	—	1	—	CPU ← IMKW
Write	0	Bit 4 = 1	0	Bit 4 = 1	CPU → IIW1
		Bits 3 and 4 = 00		Bits 3 and 4 = 00	CPU → IPFW
		Bits 3 and 4 = 10		Bits 3 and 4 = 10	CPU → IMDW
	1	Under initialization sequence	1	Under initialization sequence	CPU → IIW2
					CPU → IIW3
					CPU → IIW4
		After initialization		After initialization	CPU → IMKW

Note In the polling phase, the reading of IPOL takes priority over IRQ and IIS.

5.9.4 Initialization

ICU cannot be reset using the **RESET** signal. Therefore, ICU must be initialized when the power is turned on. Initialization takes place by writing to IIW1, IIW2, IIW3, and IIW4 according to the initialization sequence. However, as shown in Figure 5-40, there are four initialization sequences

Figure 5-40. Initialization Sequence



Remark The term address refers to A0 (IOAG = 1) or A1 (IOAG = 0).

Relative to A0 = 0 (IOAG = 1) or A1 = 0 (IOAG = 0), the initialization sequence starts at the point at which data indicating that bit 4 is 1 is written. The value written at that time will become IIW1. One initialization sequence of the four will be selected by means of the value of the SNGL bit and I4 bit within IIW1. (See Figure 5-41.) In addition, a default initialization^{Note} will be executed when IIW1 is written to.

Note A default initialization will be executed as follows:

- <1> In the edge trigger mode, the IRQ will be cleared (0). (Fixed to the edge trigger mode for the the input from TCU connected internally.)
- <2> IIS and IMKW will be cleared to 0.
- <3> Channel 0 will have the highest priority.
- <4> The normal nesting mode will be defined.
- <5> IRQ will be read and placed in the register to be read.
- <6> IIW4 will be cleared (0).

5.9.5 Registers and commands

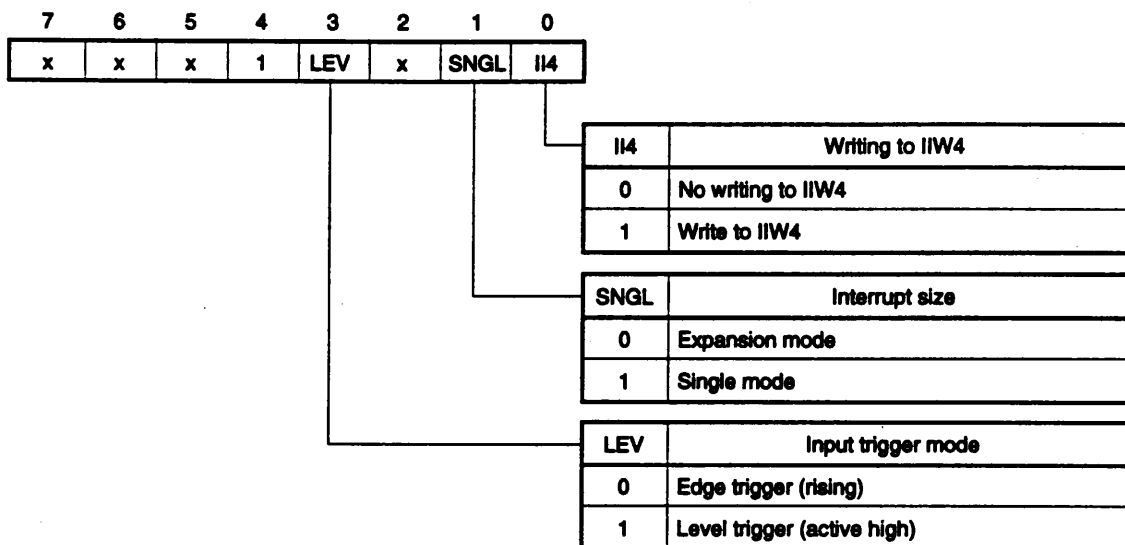
A general description of the register function to be described in this section are shown below in Table 5-20.

Table 5-20. General Description of the ICU Registers

Name	Function
IIW1	ICU initialization setting
IIW2	
IIW3	
IIW4	
IMKW	Defines interrupt request mask
IPFW	Defines and alters the interrupt request priority and the FI command (interrupt processing end statement)
IMDW	Defines the nesting mode, polling operation and registers to be read ((IRQ and IIS)
IRQ	Indicates the existence or non-existence of an interrupt request (reading enabled by an IMDW setting)
IIS	Indicates an in-service interrupt (reading enabled by the IMDW setting)

(1) Interrupt Initialization Word 1 Register (IIW1)

Figure 5-41. Interrupt Initialization Word 1 Register (IIW1)



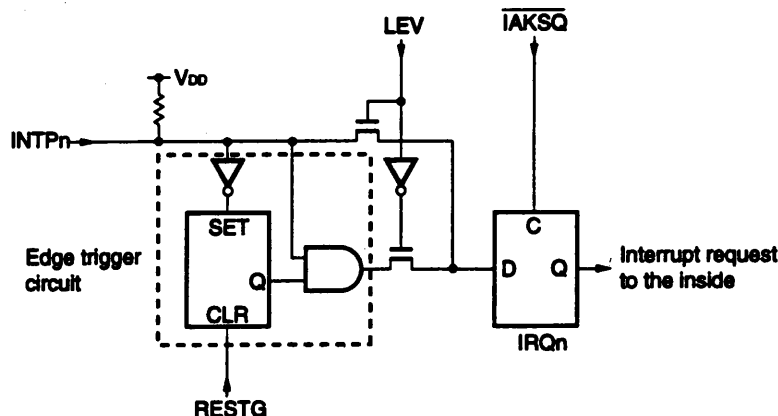
Remark x: don't care

(a) LEV Bit

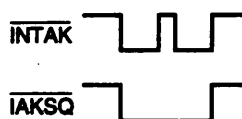
This bit sets the interrupt request trigger mode. In the case of the level trigger, if the INTP_n input is 1, it will be considered an interrupt request any number of times. However, for the edge trigger, if there is no rising edge, it will not be considered an interrupt request. In other words, in the case of an edge trigger, once an interrupt is acknowledged unless the INTP_n input is reset to 0 and then set to 1 again, it will not be considered an interrupt.

A request from TCU (INTL2 when connected to INTL0 and TCU) is always rising edge trigger.

Figure 5-42. Interrupt Request Input Circuit



Remarks 1. $\overline{\text{IAKSQ}}$ is the signal that goes low in the interrupt acknowledge sequence.



2. RESTG goes high when IIW1 is to be written to and when $\overline{\text{IAKSQ}}$ rises. When the flip-flop (F/F) is cleared, it goes low.

(b) SNGL Bit

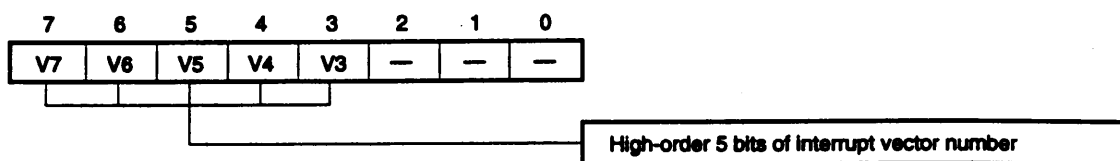
This bit defines the size of the interrupt (whether or not there is a cascade connection). That is, if only the internal ICU is used, it is set to 1 to specify the single mode. To cascade-connect an interrupt controller, such as the μ PD71059 to the INTP_n pin, set this bit to zero to place it in the expansion mode. If 0 is set to this bit, a write to IIW3 is executed during the initialization sequence.

(c) II4 Bit

This bit specifies whether or not to write to IIW4 in the initialize sequence. When not writing to IIW4 is specified, it will remain cleared to 00H due to the default initialization.

(2) Interrupt Initialization Word 2 Register (IIW2)

Figure 5-43. Interrupt Initialization Word 2 Register (IIW2)



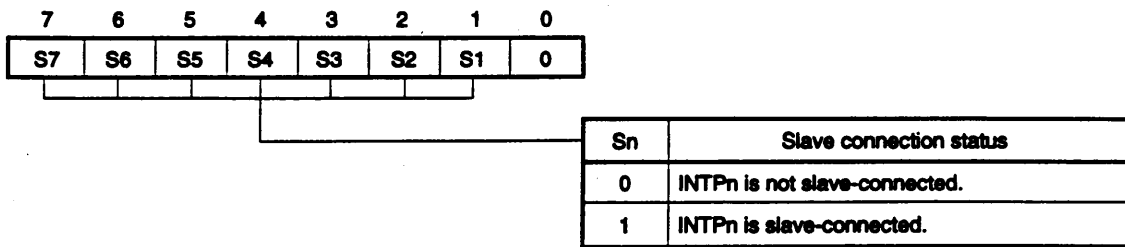
Place the high-order five bits of the 8-bit interrupt vector in V3 to V7. The remaining three low-order bits will be output based on the level at which the interrupt was acknowledged.

Figure 5-44. Interrupt Vector Number Generation

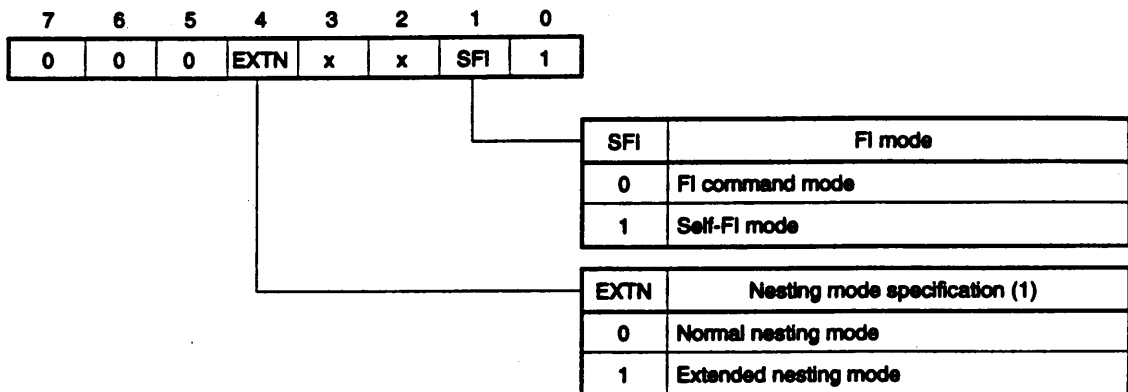
Interrupt level	7	6	5	4	3	2	1	0
INTP0	V7	V6	V5	V4	V3	0	0	0
INTP1	V7	V6	V5	V4	V3	0	0	1
INTP2	V7	V6	V5	V4	V3	0	1	0
INTP3	V7	V6	V5	V4	V3	0	1	1
INTP4	V7	V6	V5	V4	V3	1	0	0
INTP5	V7	V6	V5	V4	V3	1	0	1
INTP6	V7	V6	V5	V4	V3	1	1	0
INTP7	V7	V6	V5	V4	V3	1	1	1

(3) Interrupt Initialization Word 3 Register (IIW3)

Figure 5-45. Interrupt Initialization Word 3 Register (IIW3)



IIW3 becomes valid only when making cascade connections. In other words, the relevant Sn is set to 1 when INTPn connects μ PD71059 in cascade as the slave (an interrupt controller connected externally to expand the interrupt channels). For example, if a slave is connected to INTP5, S5 is set to one. When this happens, ICU will not output a vector, but the slave μ PD71059 must output a vector to D0 to D7 of the data bus in the second interrupt acknowledge cycle. The subsequent CPU operation is the same as the normal CPU processing. However, a slave interrupt controller cannot be connected to the INTP0 pin.

(4) Interrupt Initialization Word 4 Register (IIW4)**Figure 5-46. Interrupt Initialization Word 4 Register (IIW4)**

Remark x: don't care

(a) EXTN Bit

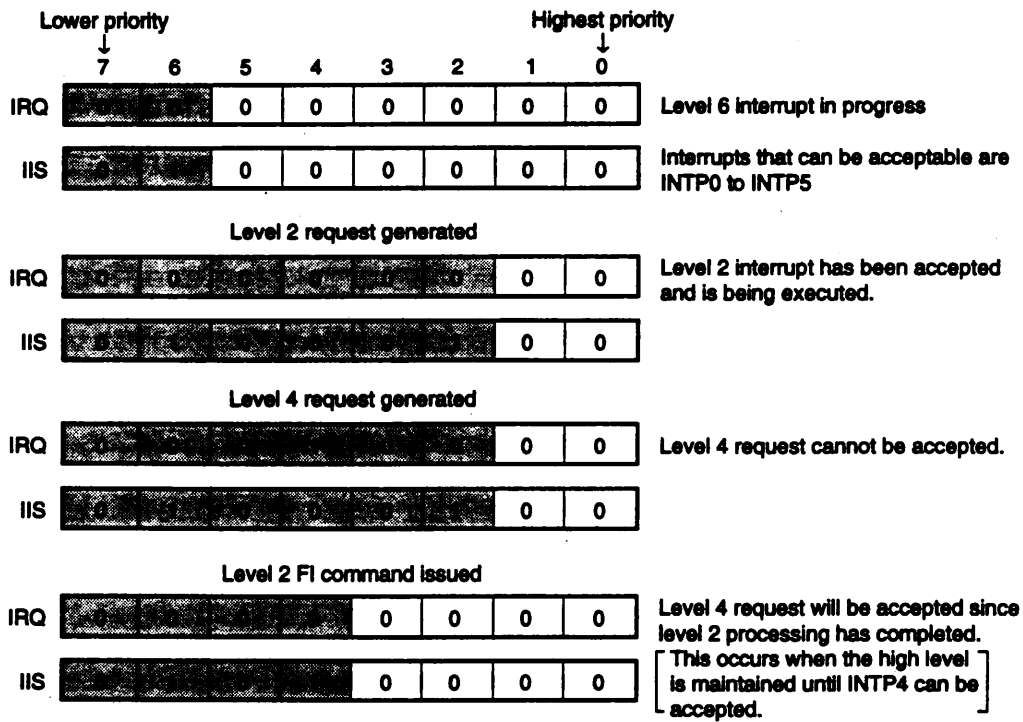
This bit sets the nesting mode. There are three types of nesting modes: normal nesting mode, extended nesting mode, and exception nesting mode. The exception nesting mode is set by IMDW. (See (7) Interrupt Mode Word Register (IMDW).)

(i) Normal Nesting Mode

In this mode, during the period that an interrupt is being executed (the relevant IIS bit is set to 1), only interrupts with a higher level of priority will be acknowledged. However, in the self-FI mode, an interrupt of any level will be acknowledged in the order of its priority because the relevant IIS bit will be cleared almost simultaneously with the interrupt acknowledgement.

IRQ is the register that indicates the level at which an interrupts request is generated, and IIS is the register that indicates the level of the interrupt that is currently being serviced. (Refer to Figure 5-54, Interrupt Request Register (IRQ) and Figure 5-55, Interrupt In-Service Register (IIS).)

Figure 5-47. Normal Nesting Mode



Remark The shaded area indicates a level that cannot be accepted.

(II) Expanded Nesting Mode

This mode has significance in the expansion mode, which will be described later. In the expansion nesting mode, only interrupt requests from the slave μ PD71059 specified by IIW3 will be acknowledged as interrupt request, even if it is at the same level. For example, even if the interrupt of a certain slave μ PD71059 is being executed and a higher priority interrupt comes from the same slave μ PD71059, the interrupt cannot be acknowledged in the normal nesting mode. This reason: the levels of priority defined for the slave μ PD71059 will lose their meaning.

Caution The issuance of an FI command in the expansion mode should take place under the following procedures. The FI command is a command issued by the CPU to inform ICU of the completion of an interrupt. (See (6) Interrupt Priority, Finish Word Register (IPFW).)

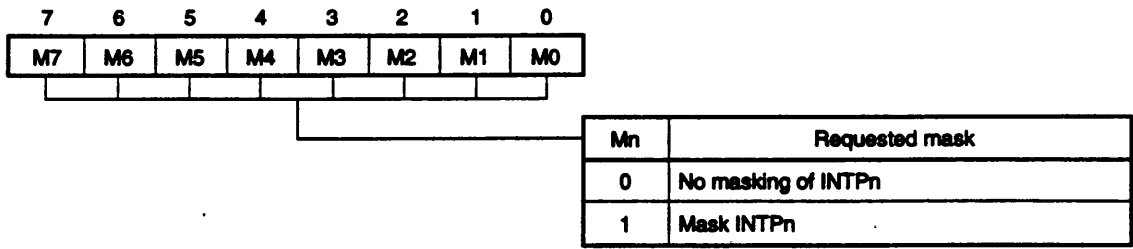
In the case of an interrupt from a slave μ PD71059, first, the CPU issues an FI command to the slave μ PD71059. Then the CPU checks, the IIS within the slave μ PD71059 to determine if any interrupts exist there that are still being serviced by the slave μ PD71059. If no such interrupts exist, the CPU issues an FI command to ICU as well. In the case of interrupts from channels other than those the slave μ PD71059 is connected to, the CPU issues an FI command only to ICU, which is the same manner as in the single mode.

(b) SFI Bit

This bit defines the FI command mode or self FI mode. These two modes differ in the IIS relevant bit clearing operations for informing ICU of the completion of interrupt service as follows. In the FI command mode, the relevant IIS bit is cleared when an FI command is issued. In the self-FI command mode, the relevant IIS bit is cleared automatically when the first interrupt acknowledge cycle is finished.

(5) Interrupt Mask Word Register (IMKW)

Figure 5-48. Interrupt Mask Word Register (IMKW)

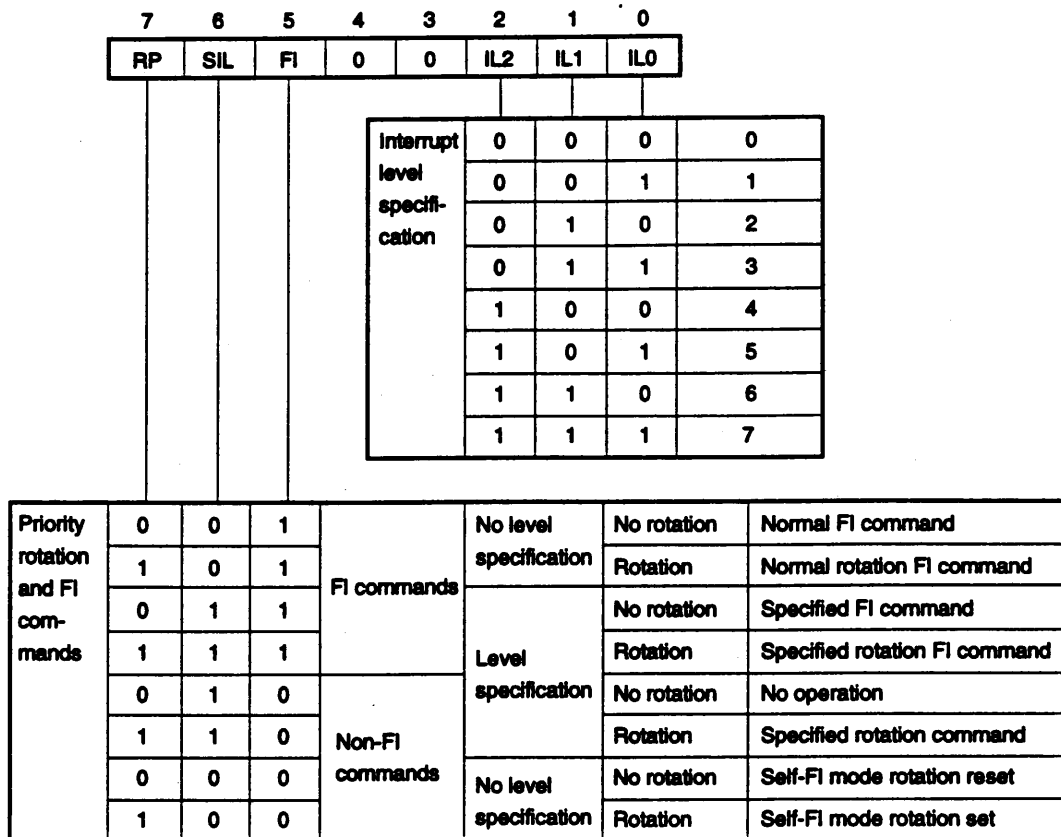


An interrupt request can be masked by defining IMKW. That is, even if IRQ is set by the generation of an interrupt request, if the relevant IMKW bit is set, ICU will not issue an interrupt request to the CPU. In addition, in the exceptional nesting mode, IIS will also be masked. (See Figure 5-52, which concerns the exceptional nesting mode.)

Be sure to use this register to mask external inputs that are used. (However, it is not necessary to mask INTP1 or INTP2 when the internal SCU or TCU output is selected as the interrupt source.)

(6) Interrupt Priority, Finish Word Register (IPFW)

Figure 5-49. Interrupt Priority, Finish Word Register (IPFW)

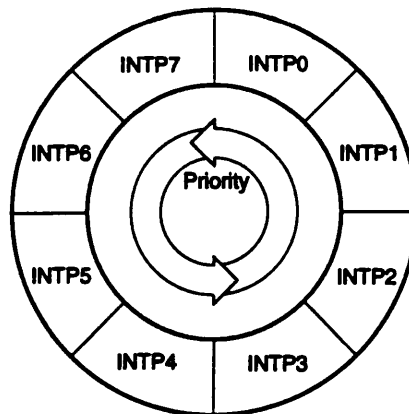


Setting and changing the interrupt priority and issuing the interrupt processing end statement (FI command issuance and clearing of the relevant IIS bit) take place simultaneously by writing to IPFW. Setting the FI bit to 1 selects FI commands that clear the IISn bit, and the remaining RP, SIL, and IL0 to IL2 bits specify the priority and interrupt level.

Setting the FI bit to 0 specifies either the specified rotation command that rotates based on bits IL0 to IL2 or a command that specifies the self-FI mode rotate set/reset.

The RP bit determines the rotation or non-rotation of the priority. When it is 1, rotation takes place. When there is a level specification, the interrupt levels specified by bits IL0 to IL2 become the lowest priorities. In addition, when no level is specified, the interrupt level at that time becomes the lowest priority. When initialization ends, the lowest interrupt level is INTP7 and the highest level is INTP0. As shown in Figure 5-50, when the interrupt priorities are rotated, if the lowest priority is INTPn, then the highest will be INTP(n+1). The interrupt priorities are as indicated in the ring shown below. (If n = 7, then n+1 = 0.)

Figure 5-50. INTL Request Priorities (Rotating Priority)



If the FI bit is set to 1 and the command described below is issued, if there is a level specification, the IISn bit of that level will be cleared. If there is no level specification, the highest priority IISn bit at that time will be cleared. Each individual command is described below.

(a) Normal FI Command

	7	6	5	4	3	2	1	0
IPFW =	1	0	1	0	0	—	—	—

- **Priority**
The priority will not change due to the issuance of this command.
- **Operational description**
This command will clear the IISn bit of the highest priority interrupt among those currently being serviced.

(b) Specified FI Command

	7	6	5	4	3	2	1	0
IPFW =	0	1	1	0	0	IL2	IL1	IL0

- **Priority**
The priority will not change due to the issuance of this command.
- **Operational description**
This command will clear the IISn bit at the level specified by bits IL0-IL2.

(c) Normal Rotation FI Command

	7	6	5	4	3	2	1	0
IPFW =	1	0	1	0	0	—	—	—

- **Priority**
The interrupt level that ended due to this command will become the lowest level priority.
- **Operational description**
This command will clear the IISn bit of the highest priority interrupt among those currently being serviced.

(d) Specified FI Command

	7	6	5	4	3	2	1	0
IPFW =	1	1	1	0	0	IL2	IL1	IL0

- **Priority**
The interrupt level specified by bits IL0 to IL2 will become the lowest priority.
- **Operational description**
This command will clear the IISn bit of the level specified by bits IL0 to IL2.

(e) Specified Rotation Command

	7	6	5	4	3	2	1	0
IPFW =	1	1	0	0	0	IL2	IL1	IL0

- **Priority**
The interrupt level specified by bits IL0 to IL2 are assigned the lowest level priority.
This command is used to change the priorities that have been set.
- **Operational description**
The IISn bit is not affected by the issuance of this command.

(f) Self-FI Mode

		7	6	5	4	3	2	1	0
Self-FI rotation appended	IPFW =	1	0	0	0	0	—	—	—

		7	6	5	4	3	2	1	0
Self-FI rotation not appended	IPFW =	0	0	0	0	0	—	—	—

If the aforementioned SSI bit of IIR4 is set to 1, the ICU will enter the self-FI mode.

- **Priority**

If a rotation is appended, the interrupt level acknowledged in that sequence will become the lowest priority when the interrupt acknowledge sequence ends.

If rotation is not appended, the priority will not change.

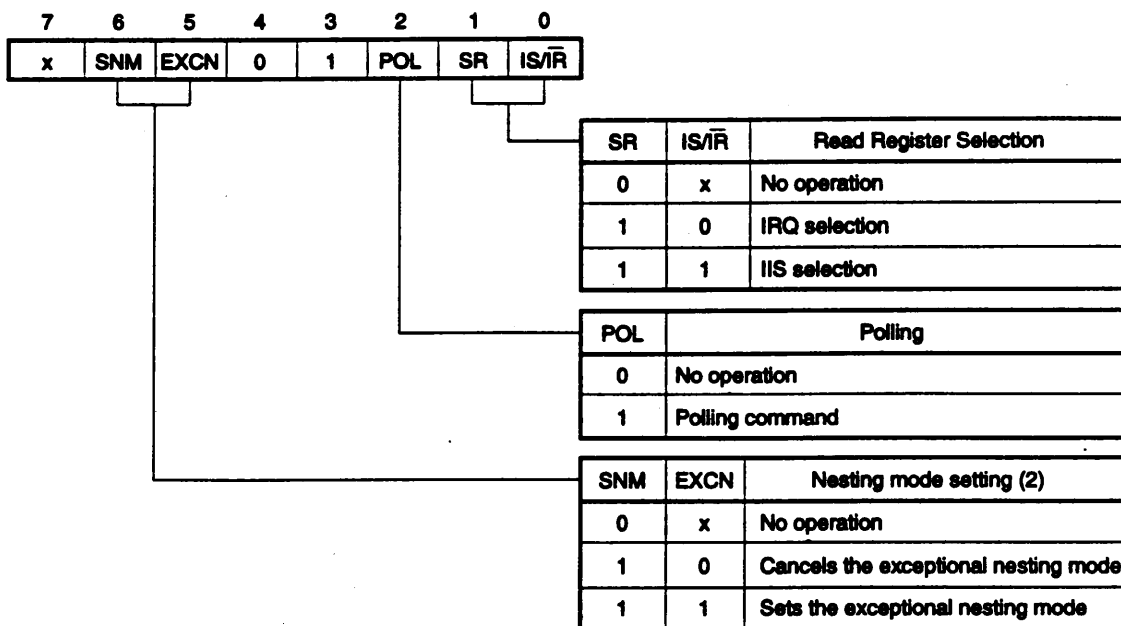
- **Operational description**

In the self-FI mode, the relevant IIS bit will set at the fall of the first acknowledge pulse. Visually, IIS will appear not to be set because it will reset immediately when there is a rise. For this reason, IPFW will not have to issue an FI command.

The issuance of an FI command by IPFW is required only when there is a priority rotation or non-rotation setting. In the self-FI mode, the relevant IIS bit will be cleared immediately. Thus, if the CPU is in the interrupt enabled condition, interrupts will be acknowledged continuously, one after the other. For this reason, it is important to pay attention to nesting management because stack overflows occur easily.

(7) Interrupt Mode Word Register (IMDW)

Figure 5-51. Interrupt Mode Word Register (IMDW)



Remark x: don't care

(a) SNM and EXCN Bits

The SNM and EXCN bits are used as a pair to set and cancel the exceptional nesting mode. When the SNM bit is zero, the nesting mode that is set by the initialization sequence becomes valid as is without making any settings.

- Exceptional Nesting Mode

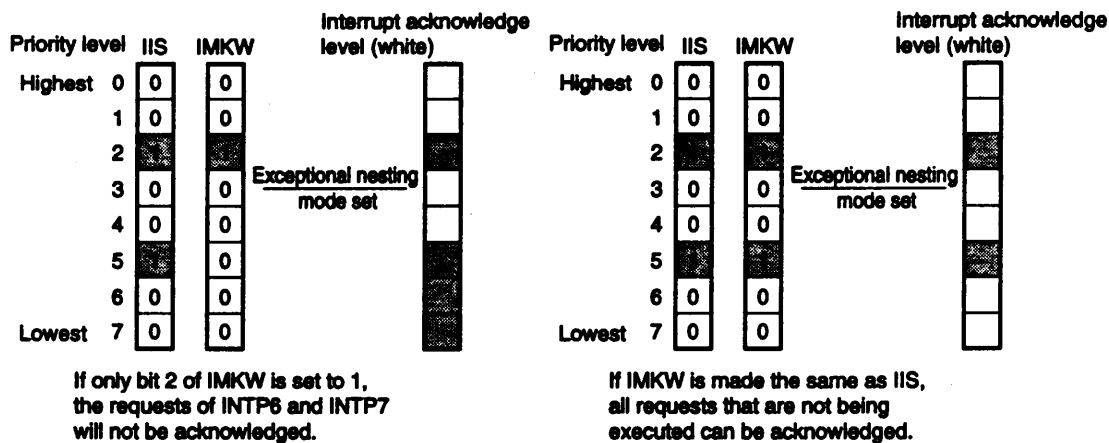
Among the interrupt nesting modes are the normal nesting mode and the extension nesting mode. (See number (4) above, which concerns interrupt initialization word 4 (IIW4).) In either case, an interrupt that is lower in priority than the one currently being serviced will not be acknowledged. It is for this reason that the exceptional nesting mode is used when there is a desire to have a low priority interrupt acknowledged.

In the exceptional nesting mode, the targets of the masking using IMKW are IRQ and IIS. That is, in order to have the level of the IIS bit specified by IMKW appear to be set to zero, interrupt requests at a level lower than that level will be acknowledged. However, since IRQ is also masked, another interrupt for that level is prohibited. An example of exceptional nesting mode setting procedures is shown below.

- (i) Read IIS.
- (ii) Write the value read to IMKW.
- (iii) Set the exceptional nesting mode.

In the above example, all of the interrupts not currently being serviced can be acknowledged. Figure 5-52 shows examples of interrupt acknowledge levels in the exceptional nesting mode.

Figure 5-52. Examples of the Exception Nesting Mode



Caution In the exceptional nesting mode, normal operation cannot take place by issuing a normal FI command because the IIS bits being serviced are masked. As a consequence, a specified FI command has to be issued. After the exceptional nesting mode has been canceled, the normal nesting mode can be issued.

(b) POL Bit

This bit is set to perform polling operations.

This bit is used when interrupt processing is to take place without the CPU issuing an interrupt sequence, or when there is a desire to obtain information concerning which request has to be serviced now.

When polling takes place, the CPU must execute the DI instruction to institute the interrupt prohibited state. Next, it will write an IMDW which has its POL bit set to 1 and issue a polling command. After this command has been issued, ICU will enter the polling phase during the period until the CPU reads from ICU. If the reading takes place during the polling phase such that A0 = 0 (IOAG = 1) and A1 = 0 (IOAG = 0), the polling data in the interrupt polling register (IPOL) will be read, not IRQ and IIS. Then, ICU will end the polling phase.

Caution If reading takes place during the polling phase such that A0 = 1 (IOAG = 1) and A1 = 1 (IOAG = 0), IMKW will be read, but ICU will end the polling phase at that point. Since this sometimes causes abnormal nesting operation, be sure to read polling data in the polling phase.

Figure 5-53. Interrupt Polling Register (IPOL)

7	6	5	4	3	2	1	0
INT	0	0	0	0	PL2	PL1	PL0

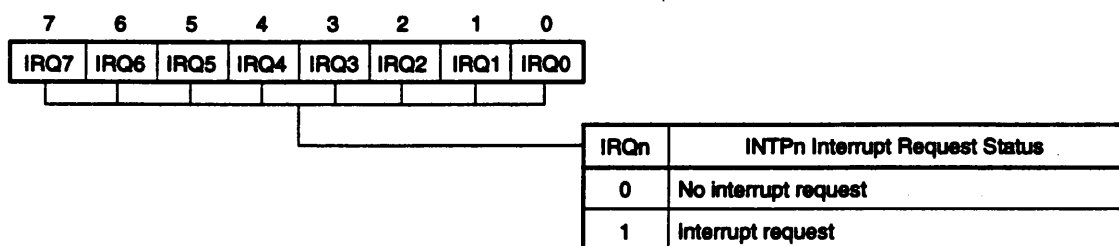
If the INT bit of the polling data that was read is 1, ICU will set the IIS bits that correspond to the ICU level indicated by PL0 to PL2, and this will be considered the servicing of that interrupt. As a consequence, the CPU must perform the processing required by the polling data that was read. In such a case, because the relevant bit of IIS will be set, the FI command must be issued at the end of the processing to clear the bit.

(c) SR and $\text{IS}/\overline{\text{IR}}$ Bits

The SR and $\text{IS}/\overline{\text{IR}}$ bits are used as a pair to specify the register to be read when the address is zero. When $\text{SR} = 0$, the content of the register will not be read. IRQ is specified if $\text{SR} = 1$ and $\text{IS}/\overline{\text{IR}} = 0$, and IIS is specified if $\text{SR} = 1$ and $\text{IS}/\overline{\text{IR}} = 1$.

(8) Interrupt Request Register (IRQ)

Figure 5-54. Interrupt Request Register (IRQ)

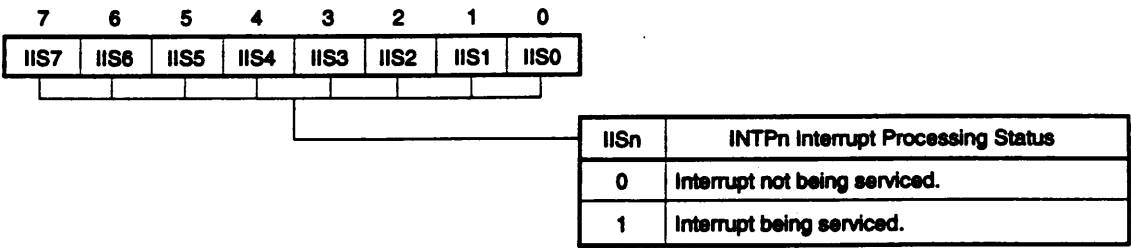


This register is read enabled by setting the IMDW SR bit and $\text{IS}/\overline{\text{IR}}$ bit ($\text{SR} = 1$, $\text{IS}/\overline{\text{IR}} = 0$). IRQ is composed of bits that indicate the interrupt status of each interrupt request from INTP0 to INTP7. If IRQ bit $n = 1$, it indicates that there are interrupts from INTPn.

The IRQ data will be sent to priority determination logic, but the interrupt of bits that are masked by IMKW will be ignored.

(9) Interrupt In-Service Register (IIS)

Figure 5-55. Interrupt In-Service Register (IIS)



This register is read enabled by setting the IMDW SR bit and IS/\overline{IR} bit ($SR = 1$, $IS/\overline{IR} = 1$).

IIS is composed of bits that indicate the CPU service condition for each interrupt request of INTP0 to INTP7. If IIS bit $n = 1$, it indicates that the interrupt routine for INTPn is being serviced.

The IIS data will be sent to priority determination logic. However, in the exceptional nesting mode, optional bits will be masked by IMKW.

5.9.6 Interrupt sequence

This subsection describes the interrupt sequence in the single mode and expansion mode.

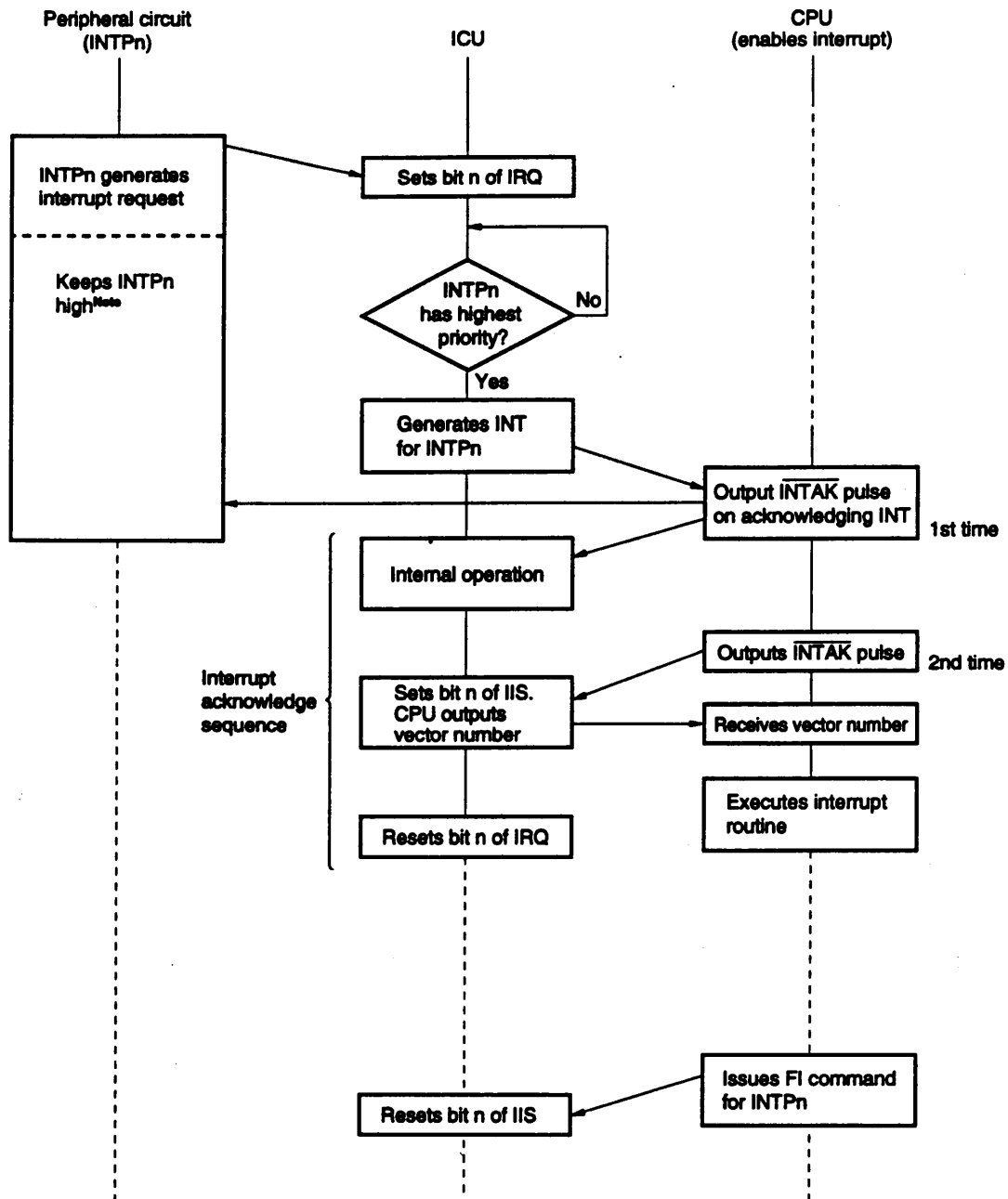
(1) Single mode

Figure 5-56 shows the interrupt sequence in the single mode. In this mode, the ICU acknowledges an interrupt request issued from a peripheral circuit and generates an interrupt request to the CPU. The CPU acknowledges this interrupt request and performs processing.

This interrupt sequence is described in detail below.

- <1> A peripheral circuit generates an interrupt request signal. This interrupt request signal (INTP_n) must remain active until the interrupt acknowledge signal ($\overline{\text{INTAK}}$) from the V40HL/V50HL becomes active, regardless of the level or edge trigger mode.
- <2> If interrupt request INTP_n is generated, the corresponding bit of the IRQ is set to 1. The ICU acknowledges only INTP_n having the highest priority and corresponding to the IRQ bits that have been set.
- <3> The ICU generates an interrupt request (INT) to the CPU when it acknowledges INTP_n.
- <4> When the CPU acknowledges the interrupt request from the ICU, it activates the first interrupt acknowledge pulse ($\overline{\text{INTAK}}$) once execution of the current instruction has completed. As described above, the INTP_n must remain active up to this point.
- <5> When the ICU receives the $\overline{\text{INTAK}}$ pulse from the CPU, it starts the interrupt acknowledge sequence.
- <6> The CPU activates the second $\overline{\text{INTAK}}$ pulse and requests the ICU for an interrupt vector number.
- <7> The ICU outputs an 8-bit interrupt vector to the lower 8 bits of an address in accordance with the second $\overline{\text{INTAK}}$ pulse. It also sets bit n of the IIS to 1, indicating that interrupt INTP_n is being processed. This completes the interrupt acknowledge sequence of the ICU. In the self-FI mode, however, bit n of the IIS is cleared at the completion of the interrupt acknowledge sequence.
- <8> The CPU receives the interrupt vector and executes interrupt servicing for INTP_n. When the interrupt routine starts, the maskable interrupts and BRK flag (single-step) interrupt are disabled (IE = 0, BRK = 0). At the end of the processing, an FI command is issued to inform the ICU of the end of the processing. This command, however, needs not to be issued in the self-FI mode.
- <9> The ICU receives the FI command, and clears bit n of the IIS. This completes the interrupt service for INTP_n. When the IIS is cleared, it is assumed that the processing of the acknowledged interrupt has ended. After that, a pending interrupt or newly generated interrupt, if any, will be enabled and can be executed.

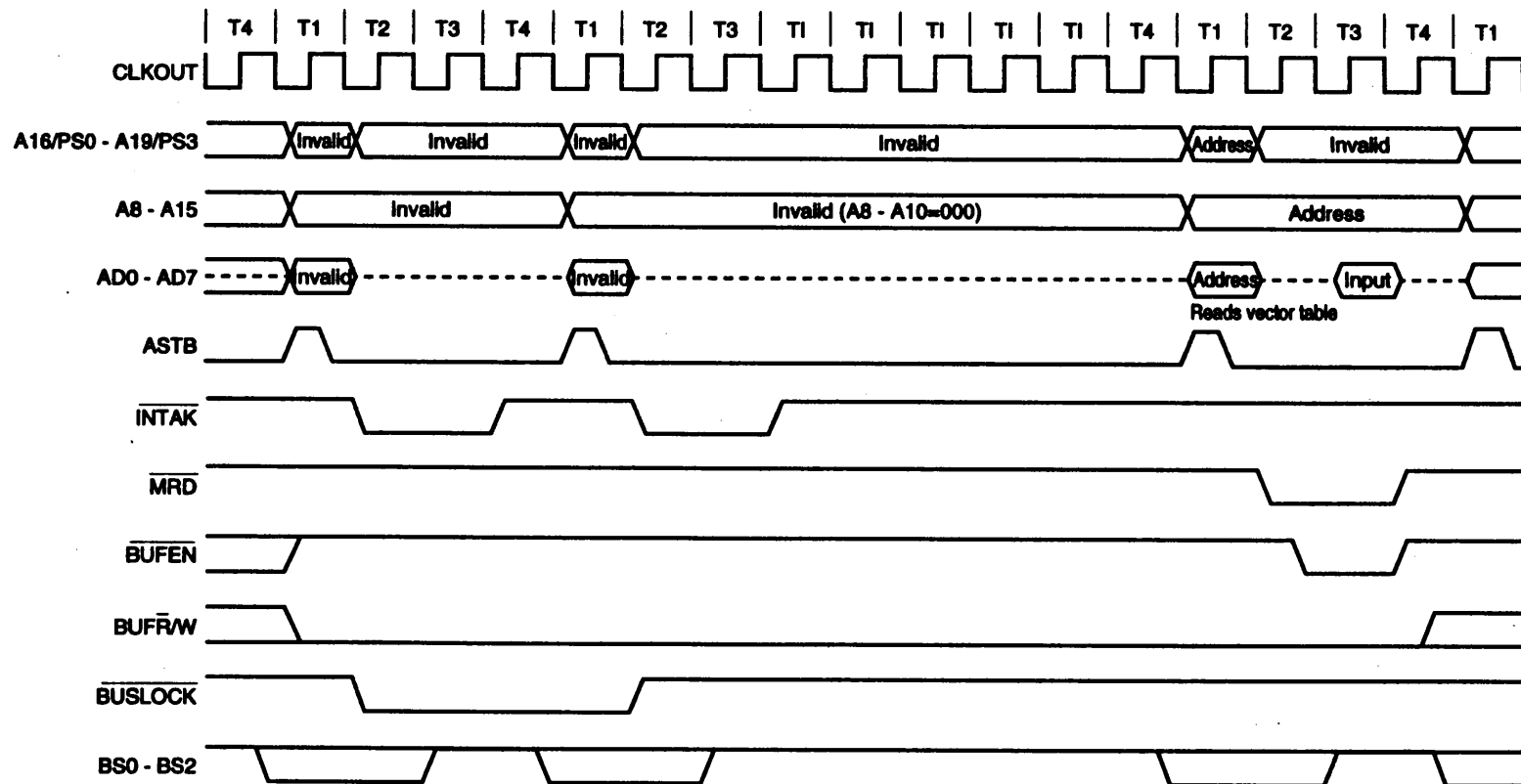
Figure 5-56. Interrupt Sequence



Note INTPn must remain high until the first $\overline{\text{INTAK}}$ pulse is input in response to the interrupt request.

Figure 5-57. Interrupt Acknowledge Timing Example (single mode) (1/2)

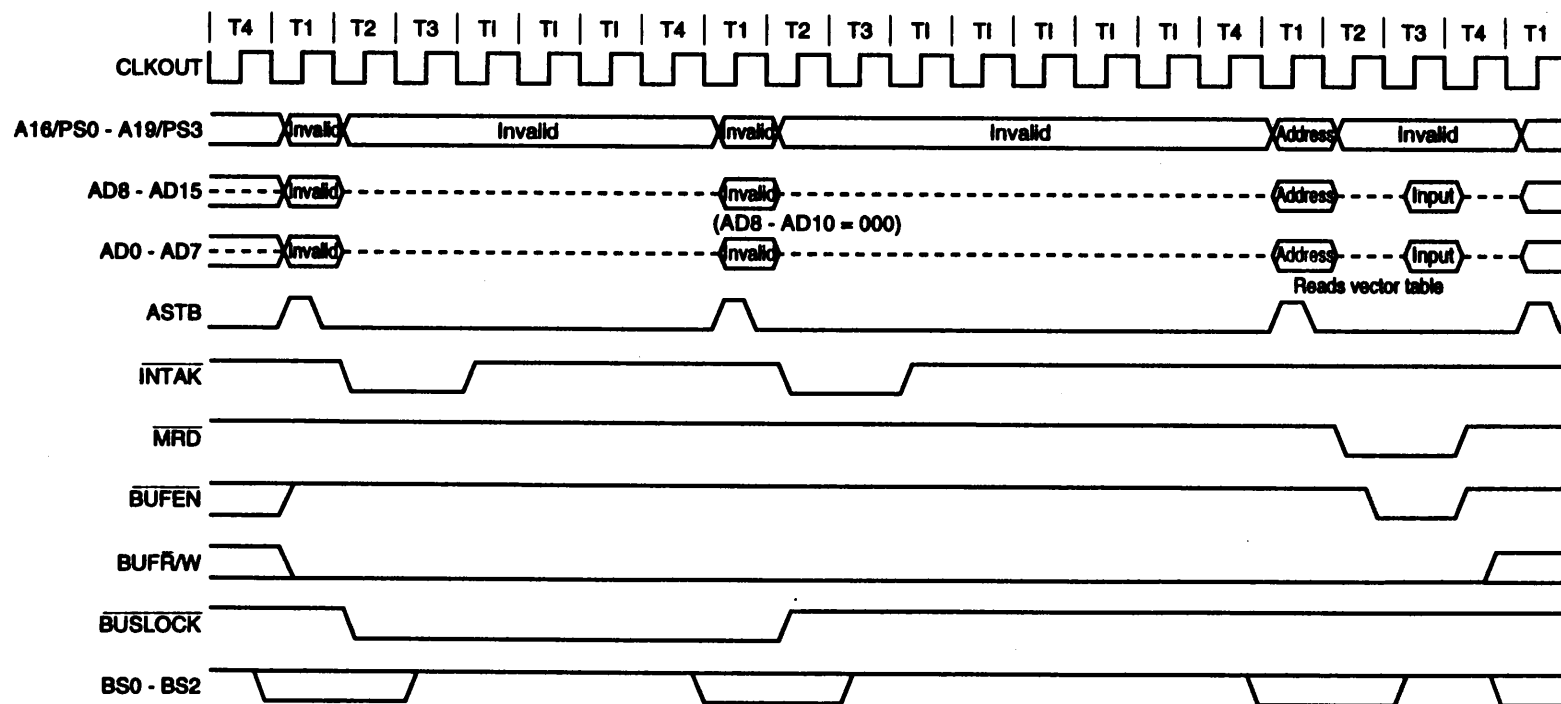
(a) V40HL



Remark The broken lines indicate the high-impedance state.

Figure 5-57. Interrupt Acknowledge Timing Example (single mode) (2/2)

(b) V50HL



Remark The broken lines indicate the high-impedance state.

(2) Expansion mode

The V40HL and V50HL can accept up to eight external interrupt requests. If more interrupt requests are necessary, the number of channels can be increased by connecting external interrupt controllers, such as the μ PD71059, in cascade.

Figure 5-58 shows an example of connecting two μ PD71059's in cascade. The INT output from a slave interrupt controller is connected to one of the INTP_n of the ICU. Note that the setting of registers/commands for the expansion mode on initialization is different from that for the single mode.

The ICU operates in two ways in the expansion mode. In one way, the interrupt request is input to the ICU without going through the slave, in which case the ICU operates in the same manner as in the single mode. The interrupt request can also be input to the ICU via the slave. In this case, the operation of the ICU is different from that in the single mode.

When the interrupt request is input to the ICU via the slave, the slave, not the ICU, gives a vector number to the CPU. The ICU determines which slave must output the vector number. This is done by using the slave address signal output to A8 through A10. (Refer to Figure 5-60, Interrupt Acknowledge Timing Example (Expansion Mode).)

Figure 5-59 illustrates how the request generated by the INTP_i of a μ PD71059 is acknowledged when the μ PD71059 is connected to the INTP_n pin of the V40HL or V50HL.

The major differences between the expanded mode and single mode are as follows:

- (a) A slave address is output to A8 through A10 in the first interrupt acknowledge cycle. This slave address is output according to the channel connected to the slave μ PD71059. Of course, the channel to which the slave is connected must be correctly set by the IIW3.

When the slave is connected to channel 5, for example, address 5H is output to A8 through A10.

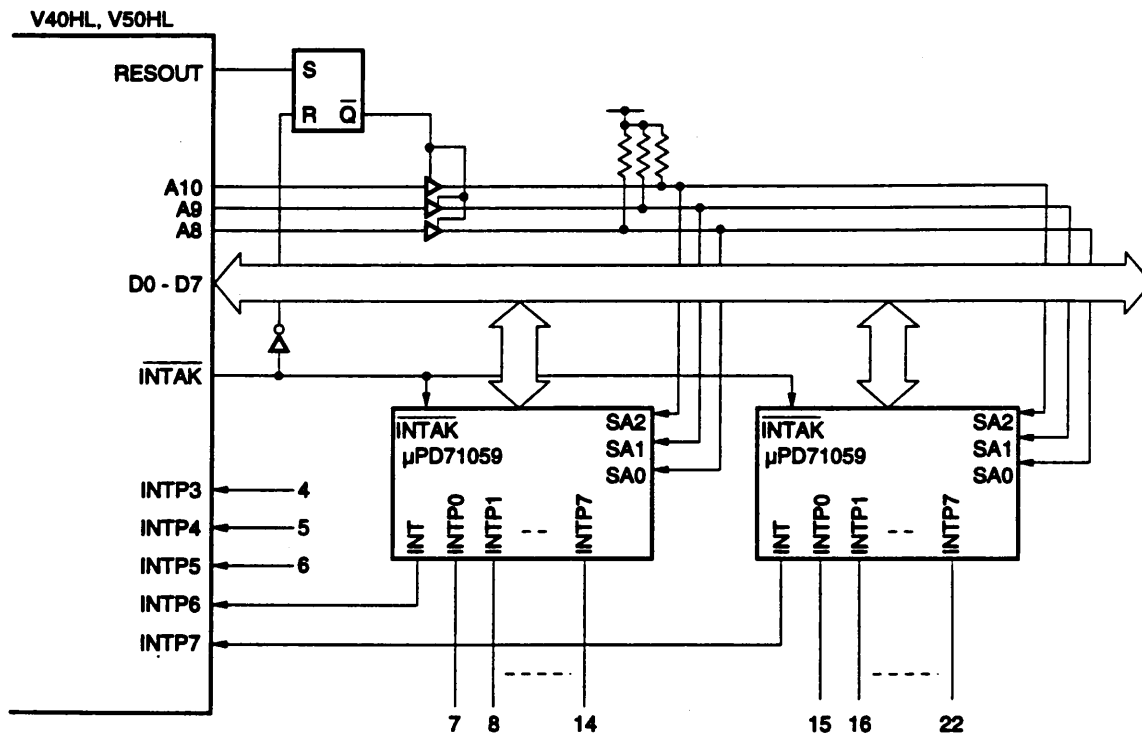
However, a slave interrupt controller cannot be connected to the INP0 pin.

Cascade Channel	A10	A9	A8
INTP1	0	0	1
INTP2	0	1	0
INTP3	0	1	1
INTP4	1	0	0
INTP5	1	0	1
INTP6	1	1	0
INTP7	1	1	1

- (b) The interrupt vector is output not by the ICU but by the slave μ PD71059 to the lower 8 bits of the data bus.

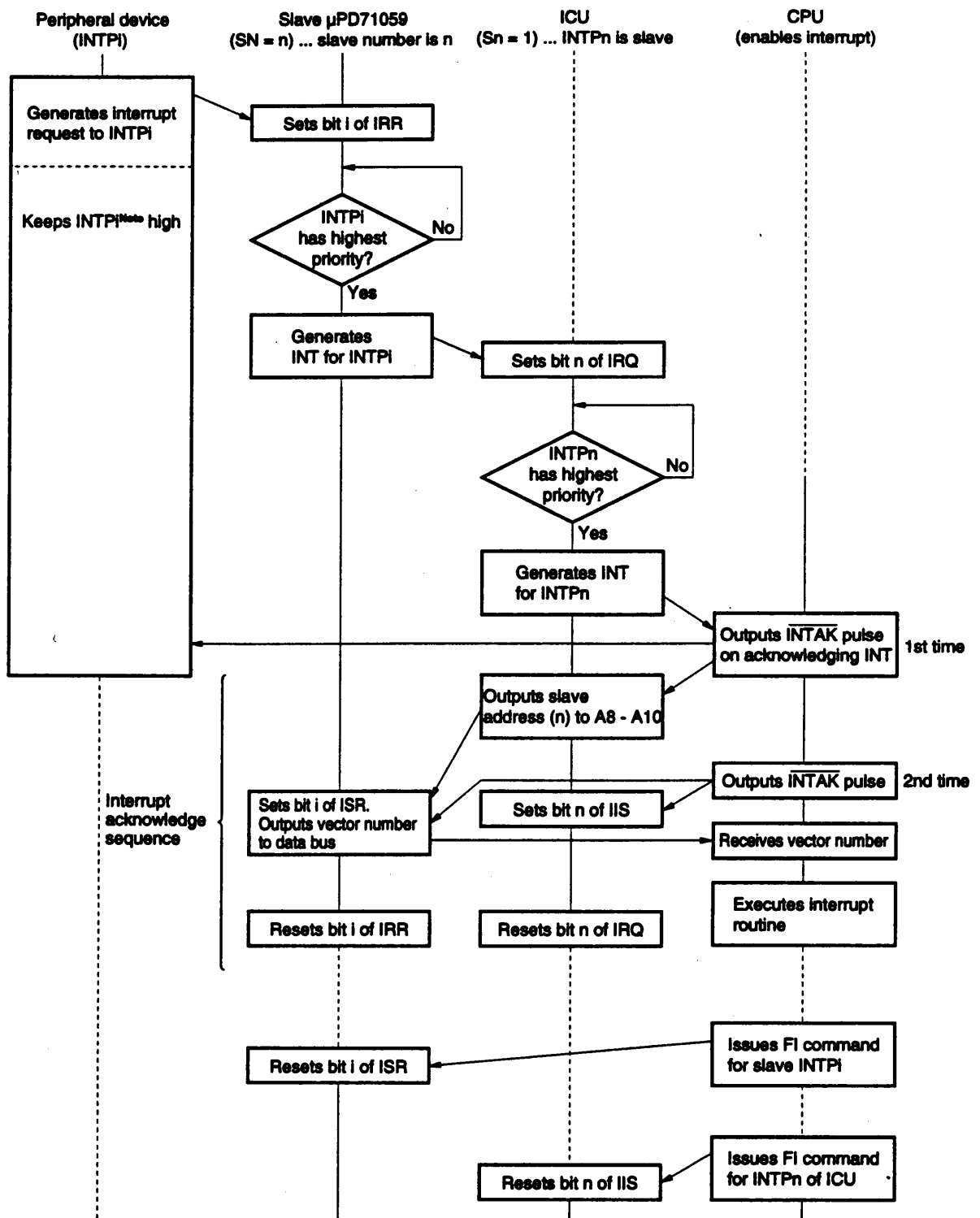
Caution The INTP_n interrupt sequence in the expansion mode for the channel to which the slave μ PD71059 is not connected is the same as in the single mode.

Figure 5-58. Example of Connecting Slave in Expansion Mode



Caution In this example, a 3-state buffer is necessary between the address bus (A8 through A10) and the SA0, SA1, and SA2 lines of the slave μ PD71059. This is because the I/O statuses of SA0, SA1, and SA2 of the μ PD71059 are undefined until the pins are initialized. Therefore, if these pins are directly connected to the address bus, the address information is affected in one way or another.

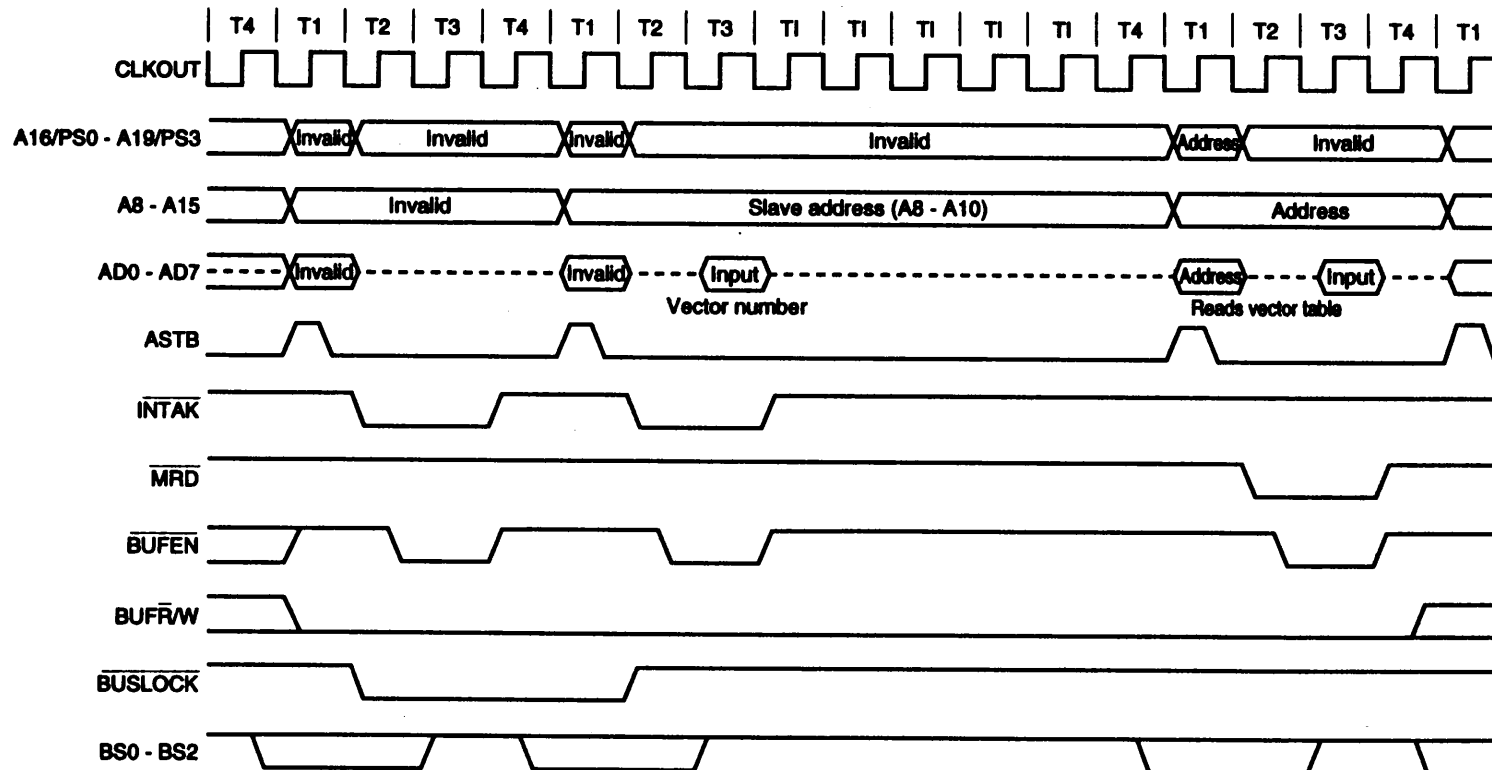
Figure 5-59. Interrupt Sequence In Expansion Mode



Note INTPn must be kept high until the first INTAK pulse is input in response to the interrupt request.

Figure 5-60. Interrupt Acknowledge Timing Example (expansion mode) (1/2)

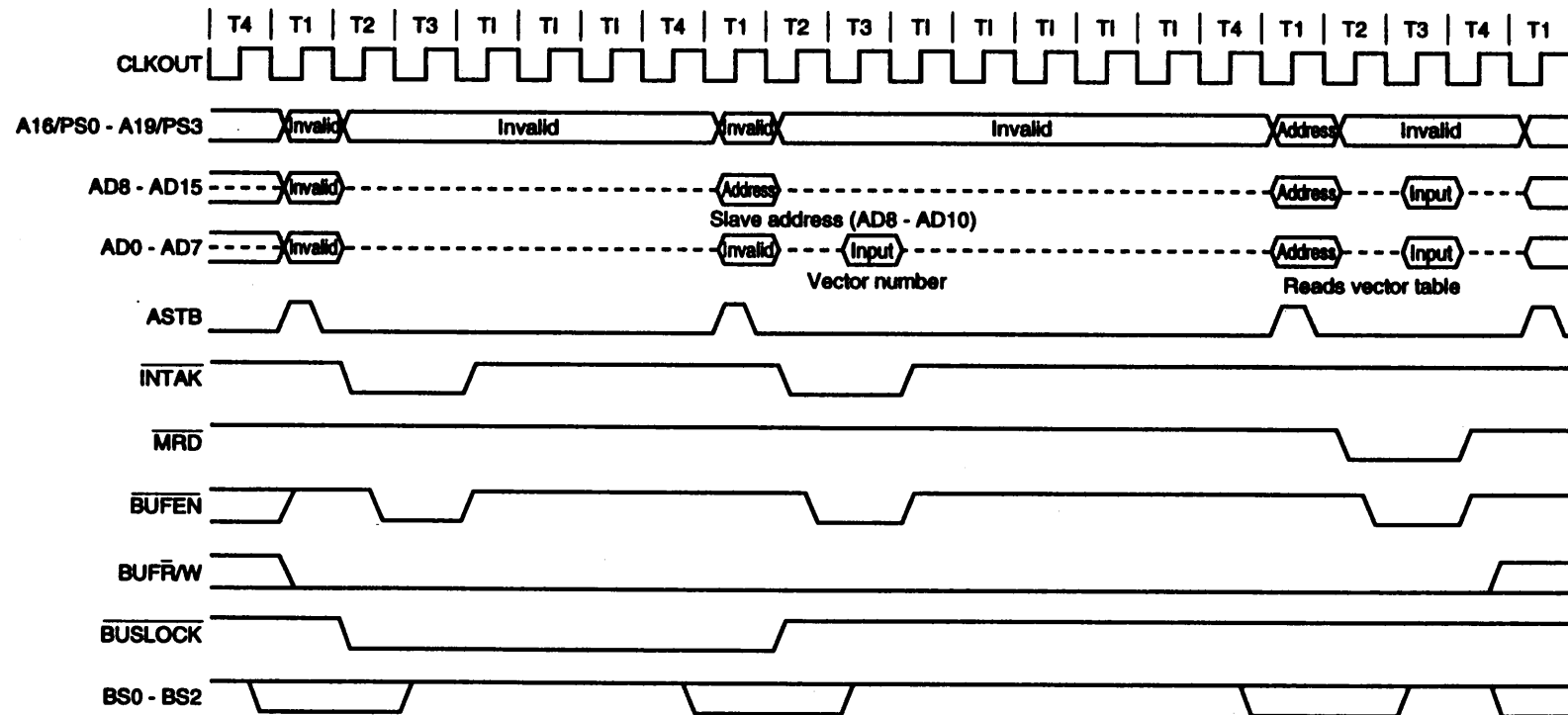
(a) V40HL



Remark The broken lines indicate the high-impedance state.

Figure 5-60. Interrupt Acknowledge Timing Example (Expanded Mode) (2/2)

(b) V50HL



Remark The broken lines indicate the high-impedance state.

5.9.7 Incomplete Interrupt

In this document, it has been described that INTP_n input should be retained until interrupt acknowledge pulse (INTAK) from CPU returns. This is because no interrupt can be acknowledged or incomplete interrupt is generated if INTP_n is changed before INTAK.

The incomplete interrupt means the state that IIS bit 7 is not set although the ICU operates as if level 7 interrupt is acknowledged. In the case that incomplete interrupt may occur, incomplete interrupt service routine should be prepared in the level 7 interrupt service routine.

To recognize the incomplete interrupt occurrence, check the setting of IIS bit 7.

In the incomplete interrupt service routine, if any other processing is not required especially, execute return instruction. However, FI command which is used to clear IIS bit should not be issued because IIS bit 7 is not set (it may cause disorder of interrupt nesting.)

Even when the level 7 interrupt is masked by IMKW, CPU executes incomplete interrupt.

Caution 1. In self-FI mode, level 7 interrupt and incomplete interrupt cannot be distinguished. In the case incomplete interrupt may occur, levels 0 to 6 should be used for interrupts, and level 7 for incomplete interrupt service only.

2. In the case of employing cascade connection for μ PD71059, etc., if incomplete interrupt is generated in the slave side, the master ICU may also generate incomplete interrupt. In this case, the ICU outputs the level 7 vector to master side.

At this time, be sure not to issue FI command to both master and slave because IIS and ISR bits of master and slave are not set.

★

5.10 DMA Controller Unit (DMAU)

DMAU has four DMA channels and functions (subset) of two types of LSI devices: μ PD71071 and μ PD71037.

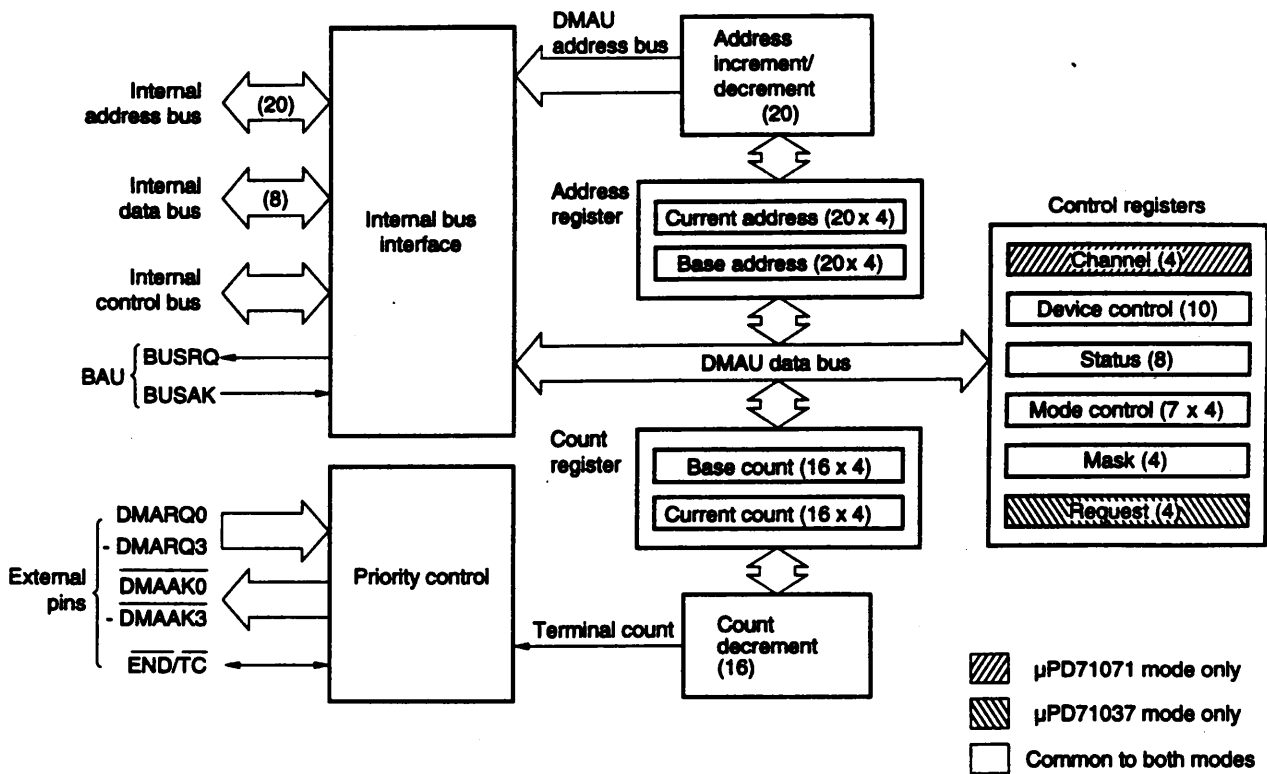
5.10.1 Features

- Two operation modes (μ PD71071 mode and μ PD71037 mode)
- 20-bit address register
- 16-bit count register
- Four independent DMA channels
- Four clocks/bus cycle
- Byte transfer or word transfer selectable
- Three types of transfer modes (can be set per channel)
Single transfer mode, demand transfer mode, and block transfer mode
- Two types of bus modes (common to all channels; bus release mode only in μ PD71037 mode)
Bus release mode
Bus hold mode
- DMA request of each channel can be masked
- Auto initialization function
- Increment or decrement of transfer address
- Two types of channel priority (fixed priority/rotating priority)
- \overline{TC} output at end of transfer
- Forced termination of service by \overline{END} input
- Cascade connection

5.10.2 Differences with the V40 and V50

- Addition of μ PD71037 operation mode

5.10.3 Internal block diagram



5.10.4 Differences between μ PD71071 mode and μ PD71071

The μ PD71071 mode of the DMAU has functions equivalent to those of the μ PD71071, except for the following functions:

Table 5-21. Differences between μ PD71071 Mode and μ PD71071

Function	μ PD71071 Mode	μ PD71071
Software request	None	Provided
Memory-memory transfer	None	Provided
DMARQ active level	High	High or low
DMAAK active level	Low	High or low
Bus cycle	4 clocks	4 or 3 clocks

5.10.5 Differences between μ PD71037 mode and μ PD71037

The μ PD71037 mode of the DMAU has functions equivalent to those of the μ PD71037, except for the following functions:

Table 5-22. Differences between μ PD71037 Mode and μ PD71037

Function	μ PD71037 Mode	μ PD71037
Memory-memory transfer	None	Provided
DMARQ active level	High	High or low
DMAAK active level	Low	High or low
Bus cycle	4 clocks	3 or 2 clocks

5.10.6 Differences between the μ PD71071 and μ PD71037 modes

All the internal registers, except the channel register and request register, are used in common in both the modes. The channel register is dedicated to the μ PD71071 mode, and the request register is dedicated to the μ PD71037 mode (refer to section 5.10.3 Internal block diagram). Even when the mode has been changed, the contents of each register are retained. The bits not used after the mode has been changed retain the status before the mode change.

The differences between the μ PD71071 and μ PD71037 modes are described below.

(1) Units of transfer

Byte or word transfer can be selected in the μ PD71071 mode. In the μ PD71037 mode, however, only byte transfer can be executed.

(2) Channel selection

Because the channel register is not used in the μ PD71037 mode, a channel must be specified when a command is issued. The channel is specified by using part of the data written to the mode control register. The channel can also be specified by using the address register or count register, by specifying an I/O address to be accessed.

(3) Accessing base/current register

The address register and count register of each channel consist of a base register and a current register. These registers are accessed differently depending on the mode, as follows:

(a) μ PD71071 mode

The current and/or base register can be selected by using the channel register.

- { Read : current register only
Write : both base register and current register
- Read/write: base register only

(b) μ PD71037 mode

- { Read : current register only
Write : both base register and current register

(4) Software DMA request

In the μ PD71037 mode, the request register which is not used in the μ PD71071 mode can be used to generate a DMA request via software.

(5) Bus mode

The μ PD71071 mode has a bus release mode and a bus hold mode.

In the μ PD71037 mode, the bus mode cannot be selected and operation is always performed in the bus release mode of the μ PD71071 mode.

(6) DMAU location address

In the μ PD71071 mode, the I/O addresses of DMAU are always located contiguously. In the μ PD71037 mode, however, location to contiguous addresses or to odd/even addresses can be selected by using SCTL in the system I/O area.

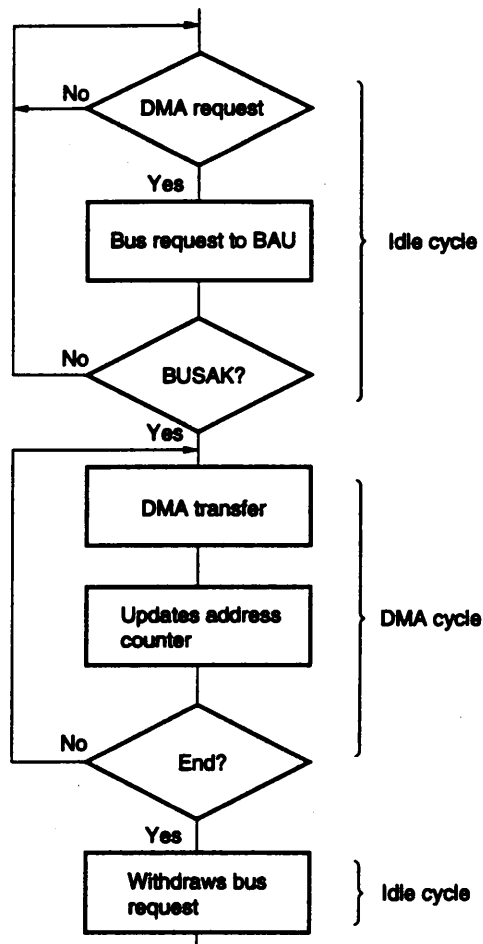
(7) Miscellaneous

The basic commands can be used in both the modes. However, some commands can be used only in the μ PD71071 mode and others can be used only in the μ PD71037 mode. The I/O address of each command is not compatible between the μ PD71071 mode and μ PD71037 mode. All the commands in the μ PD71037 mode must be executed by the IN/OUT instruction of the byte type.

5.10.7 Basic operation of DMAU

If external I/O devices issue requests for DMA service to the DMARQn pin, V40HL/V50HL acquires the bus control and then returns the DMAAKn signal to device that has issued the request with the highest priority. Then DMA transfer is executed by outputting an address to the memory, selecting an I/O device, and performing read/write control.

Figure 5-61. DMAU Operation Flow



The operation of the DMAU is performed in the DMA cycle and idle cycle.

Caution If a refresh request with a high priority is issued in the DMA cycle, the V40HL/V50HL may enter the bus wait status in which it returns the bus control to the REFU.

(1) Idle cycle

In the idle cycle, a bus master other than the DMAU holds the bus control. The DMAU does not accept a valid DMA request, or even if it has accepted the valid DMA request, it does not acquire the bus control.

In the idle cycle, the DMAU perform the following operations:

(a) Sampling of DMA request

The DMAU samples the statuses of the four DMARQn pins (whether DMARQ3 can be used is specified by OPCN) at every operating clock, and sets the result of the sampling to the status register.

(b) Requesting bus control

When a valid DMA request is issued, the DMAU requests the BAU for bus control by issuing an internal signal BUSRQ.

The DMAU continues sampling the DMA request until it obtains bus control (until the BUSAK (internal signal) from the BAU becomes active).

(c) Determining DMA channel

When the DMAU has acquired the bus control, it stops sampling of the DMA request, selects the DMA channel having the highest priority of the DMA channels that have issued the request, and asserts the corresponding DMAAKn signal active.

(d) Transfer of data with CPU (programming of DMAU)

The transfer address, the number of transfer times, and a DMA operation mode of the DMAU must be programmed by the CPU before the DMAU executes DMA transfer. In the idle cycle, the DMAU can read data from or write data to the CPU by executing the IN or OUT instruction to an address determined by OPHA and DULA in the system I/O area. At this time, a register in the DMAU and a command are selected by address signals A0 through A3.

(2) DMA cycle

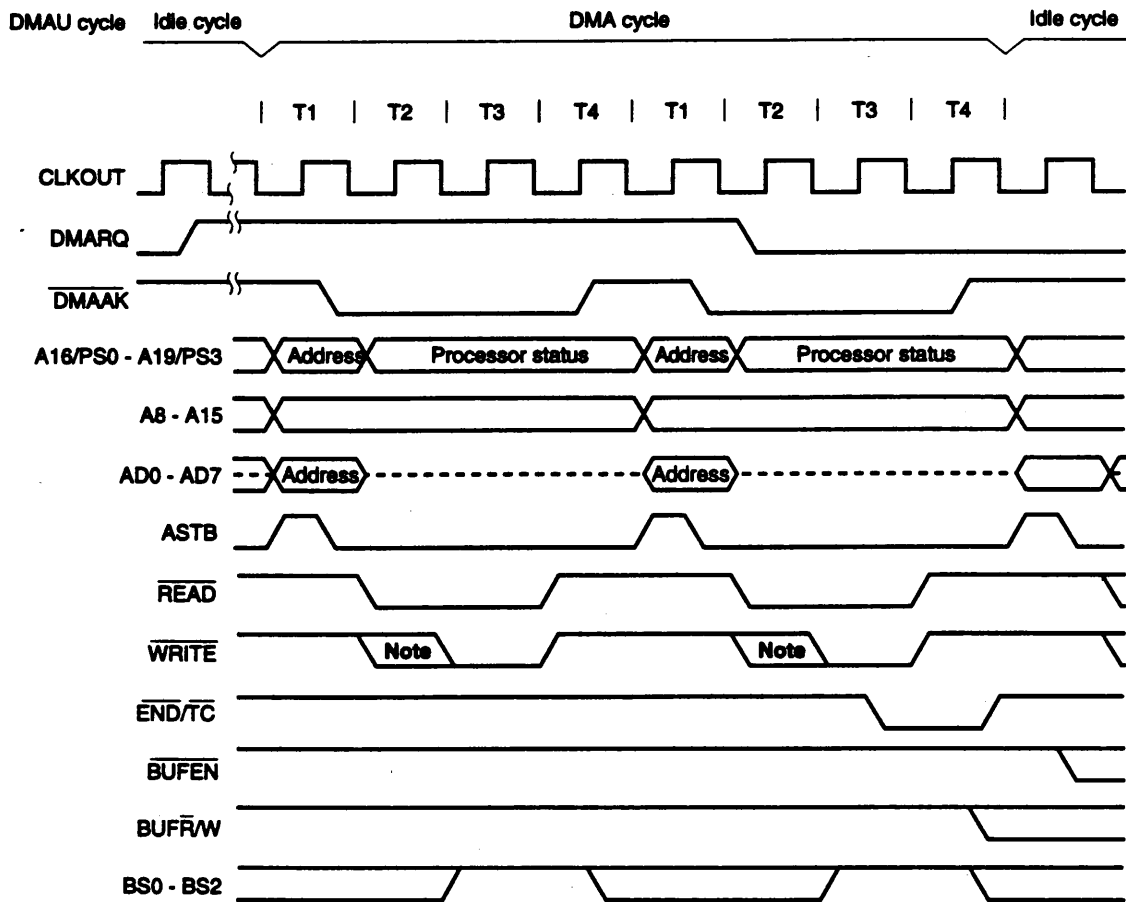
A DMA cycle consists of four clocks. If a wait state (TW) is inserted in the DMA cycle, the TW state is inserted following the T3 state. The \overline{TC} and \overline{DMAAK} signal continue the status in the T3 state. (However, in the DMA cycle with the DMAU connected in cascade to an external DMA controller, the wait state is controlled by the external DMA controller.)

Figure 5-62 shows an example of DMA timing. Take \overline{READ} and \overline{WRITE} in the figure as follows:

Signal Name in Diagram	I/O-to-Memory Transfer	Memory-to-I/O Transfer
\overline{READ}	\overline{IORD}	\overline{MRD}
\overline{WRITE}	\overline{MWR}	\overline{IOWR}

Figure 5-62. DMA Timing

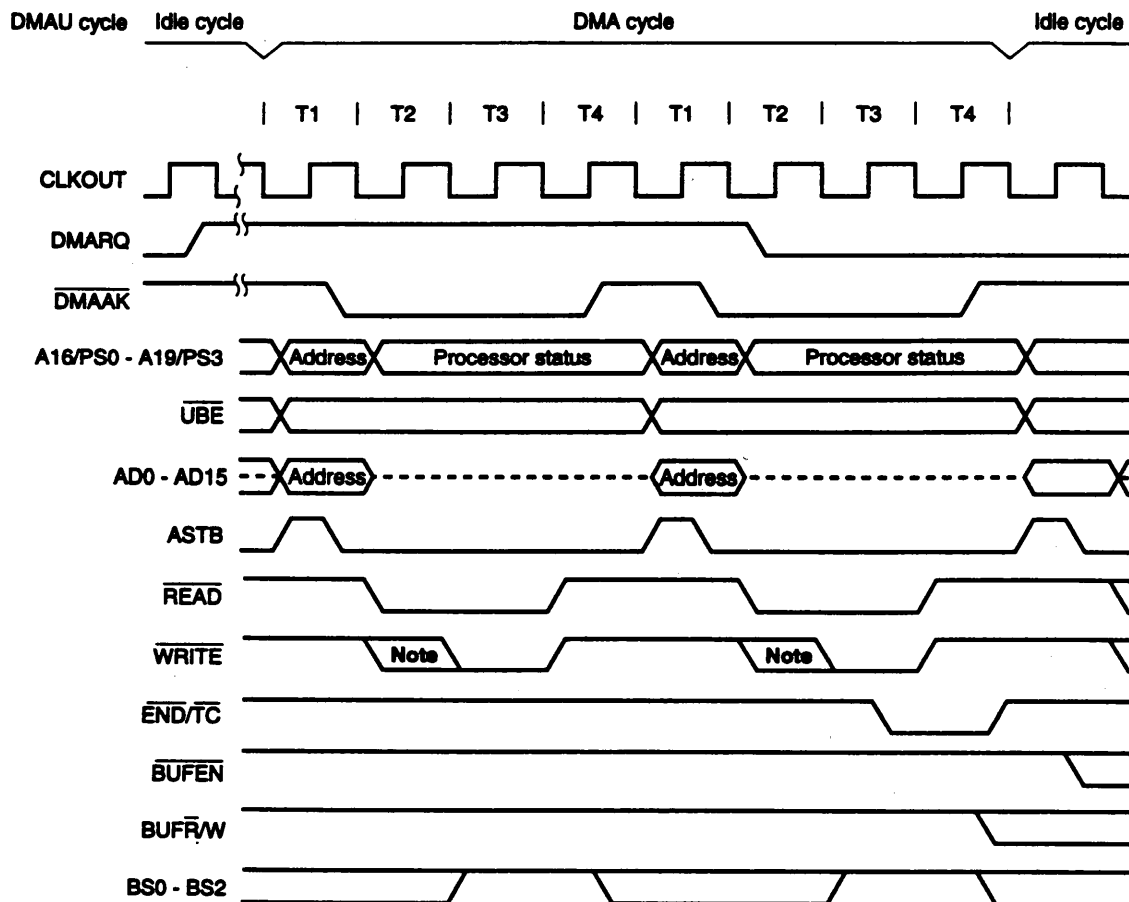
(a) V40HL



Note A low level is output in the expanded write mode.

Remark The broken lines indicate the high-impedance state.

(b) V50HL



Note A low level is output in the expanded write mode.

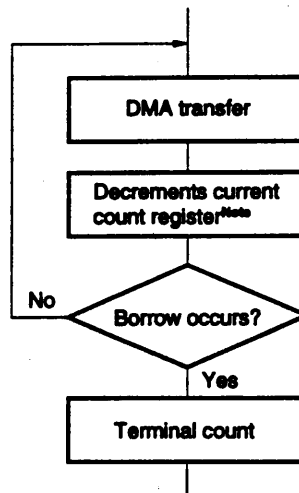
Remark The broken lines indicate the high-impedance state.

(3) Terminal count (TC)

One DMA service ends if terminal count occurs internally or if the $\overline{\text{END}}$ signal is input from an external source. (This service ends under other conditions depending on the operation mode.)

Figure 5-63 shows the relationship between the occurrence of the terminal count and the current count register. The contents of the current count register decrement each time a DMA transfer executes. If a borrow occurs from the current count register, a terminal count occurs, and a low-level signal is output to the $\overline{\text{TC}}$ pin. The contents of the current count register are evaluated after 1 byte of data has been transferred. Therefore, the number of times DMA transfer is executed is one more than the set value of the current count register.

Figure 5-63. Occurrence of Terminal Count (TC)



Note In μ PD71071 mode: DCC
In μ PD71037 mode: DRBCxx
DWBCxx

If the DMA service ends due to the $\overline{\text{END}}$ input or occurrence of the terminal count, the bits of the mask register corresponding to the channel that has ended the service are set (if auto initialization is not set), and the DMARQn input of that channel is masked.

5.10.8 μ PD71071 mode

The DMAU is set in the μ PD71071 mode when the DMAM bit of SCTL in the system I/O area is cleared to 0. The μ PD71071 mode is also set immediately after reset.

(1) Addressing

The internal registers of the DMAU are addressed by OPHA and DULA set in the system I/O area and address signals (A0 through A3). The DMAU has 24 registers as shown in Table 5-23. Of these, the DBA, DCA, DBC, DCC, and DMD registers are provided to each channel. It is therefore necessary to first specify the channel whose registers are to be accessed by using the DMA channel register (DCH).

Table 5-23. Registers of DMAU (μ PD71071 mode)

Register Name	Bit Size
DMA channel register (DCH)	5 bits
DMA base address register (DBA)	20 bits x 4
DMA current address register (DCA)	20 bits x 4
DMA base count register (DBC)	16 bits x 4
DMA current count register (DCC)	16 bits x 4
DMA mode control register (DMD)	7 bits x 4
DMA device control register (DDC)	5 bits
DMA status register (DST)	8 bits
DMA mask register (DMK)	4 bits

Table 5-24. Addresses of DMAU Internal Registers (μPD71071 mode)

Higher Address	Lower Address				Register	Operation	
A15 - A8 (OPHA)	A7 A6 A5 A4 (DULA)	0	0	0	0	DICM	Write
		0	0	0	1	DCH	Read/write
		0	0	1	0	DBC/DCC (lower byte)	Read/write
		0	0	1	1	DBC/DCC (higher byte)	Read/write
		0	1	0	0	DBA/DCA (lower byte)	Read/write
		0	1	0	1	DBA/DCA (middle byte)	Read/write
		0	1	1	0	DBA/DCA (higher byte)	Read/write
		0	1	1	1	Reserved	—
		1	0	0	0	DDC (lower byte)	Read/write
		1	0	0	1	DDC (higher byte)	Read/write
		1	0	1	0	DMD	Read/write
		1	0	1	1	DST	Read
		1	1	0	0	Reserved	—
		1	1	0	1	Reserved	—
		1	1	1	0	Reserved	—
		1	1	1	1	DMK	Read/write

Remark These addresses are not affected by the value of the IOAG bit of SCTL.

(2) Commands in μ PD71071 mode

Commands that read from or write to the registers of the DMAU are issued by executing the I/O instruction to an address set in the system I/O area and are selected by 4 bits of address signals, A0 through A3. Therefore, the address of a command is the values of A0 through A3.

Table 5-25. DMAU Command Address (μ PD71071 mode)

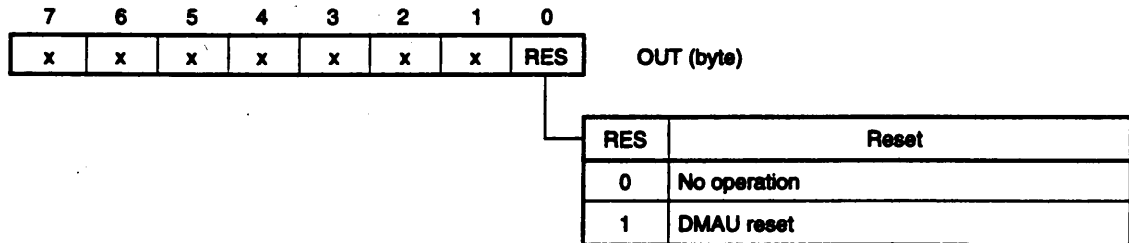
Address	Operation	Command
0H	W (B)	Initialize (DICM)
1H	R (B)	Channel register read (DCH)
	W (B)	Channel register write (DCH)
2H	R/W	Count register read/write (DBC/DCC)
3H	R/W	
4H	R/W	Address register read/write (DBA/DCA)
5H	R/W	
6H	R/W (B)	
8H	R/W	Device control register read/write (DDC)
9H	R/W	
0AH	R/W (B)	Mode control register read/write (DMD)
0BH	R (B)	Status register read (DST)
0FH	R/W (B)	Mask register read/write (DMK)

Caution Execute commands marked (B) using the byte IN/OUT instruction. Use only the address and operation combinations listed in this table.

(3) Initialization

As shown in Table 5-26, the DMAU is initialized by the reset signal ($\overline{\text{RESET}}$). It is also initialized in the same manner when the initialize command shown below is issued. In other words, the DMAU can be initialized by software as well as hardware.

Figure 5-64. DMA Initialize Command Register (DICM)



Remark x: don't care

If the RES bit is set to 1, the internal registers of the DMAU will be initialized as shown in Table 5-26. After initialization, the RES bit is automatically cleared to 0.

Table 5-26. Initializing Registers by Reset

Register Name	State after Initialization																
Address register	Not affected.																
Count register	Not affected.																
Channel register	<table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>-</td><td>-</td><td>-</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table> <div>(CH0 selected)</div>	7	6	5	4	3	2	1	0	-	-	-	0	0	0	0	1
7	6	5	4	3	2	1	0										
-	-	-	0	0	0	0	1										
Mode control register	All bits are cleared.																
Device control register	All bits are cleared.																
Status register	All bits are cleared.																
Mask register	All bits are set. (all channels are masked.)																

(4) Registers and commands

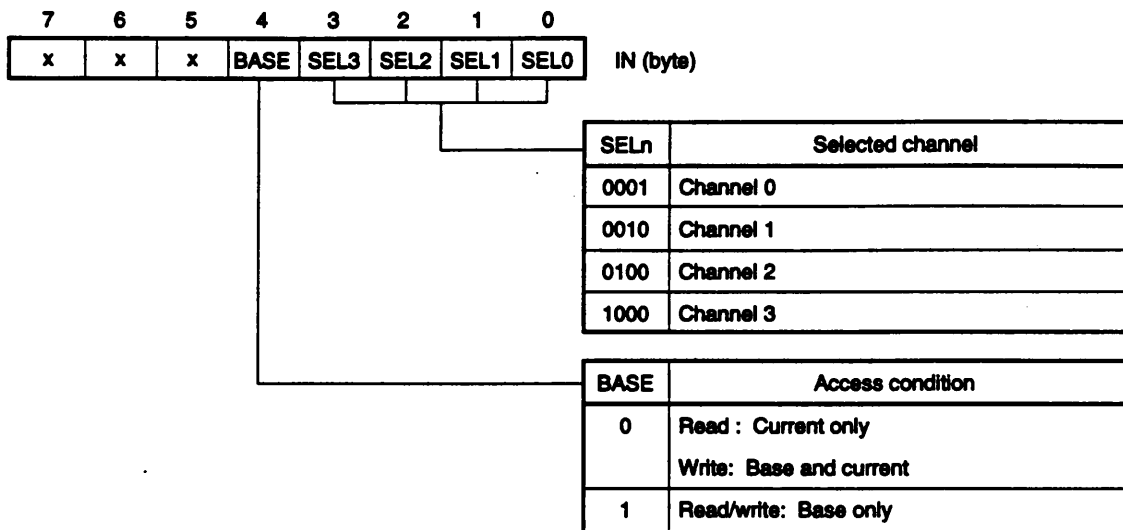
(a) DMA channel register (DCH)

This register selects one of the four DMA channels that is to be programmed by the CPU. Therefore, the channel whose registers are to be programmed is specified by using this register immediately before setting the address register, count register, and mode control register.

The format of this register differs depending on whether it is read or written.

(i) When DCH is read

Figure 5-65. DMA Channel Register Read Command



Remark x: don't care.

- **BASE bit**

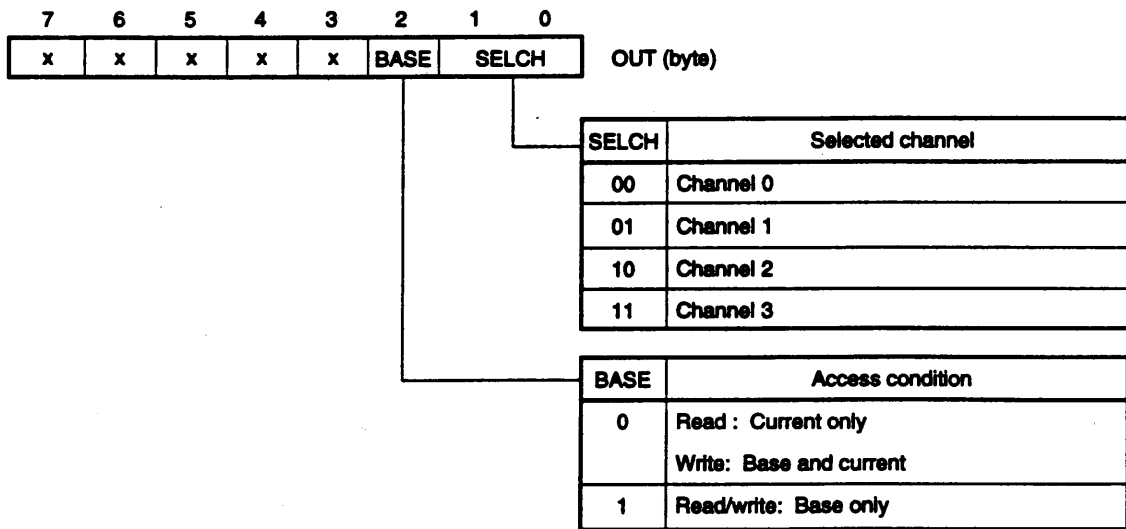
This bit indicates which of the address or count registers is accessed for read or write. If this bit is set, the base register can be read from or written to. If it is cleared, the current register can be read from and the base and current registers can be written to.

- **SEL bits**

These bits indicate which channel is currently selected.

(II) When DCH is written

Figure 5-66. DMA Channel Register Write Command



Remark x: don't care

- **BASE bit**

This bit indicates which of the address or count registers of the channel selected by the SELCH bits is accessed for reading or writing, in the same manner as when the DCH register is read. If this bit is set, the base register can be read from or written to. If it is cleared, the current register can be read from and the base and current registers can be written to.

- **SELCH bits**

These bits specify which channel is to be selected.

(b) DMA base/current count register (DBC/DCC)**Figure 5-67. DMA Count Register Read/Write Command**

7	6	5	4	3	2	1	0	
C7	C6	C5	C4	C3	C2	C1	C0	IN/OUT

7	6	5	4	3	2	1	0	
C15	C14	C13	C12	C11	C10	C9	C8	IN/OUT

The current register of each channel consists of a base count register and a current count register. Both the base count register and current count register are allocated to the same address. Which between the base count and current count registers is selected for read or write is specified by the channel register. The contents of the current count register decrements by one each time DMA transfer executes. The base count register holds the current set value until a new count value is set, and the current set value is transferred to the current count register when auto initialize is executed (refer to **section 5.10.11 Auto Initialization**). The count register can be read from or written to by using an IN/OUT instruction of the word type.

(c) DMA base/current address register (DBA/DCA)

Figure 5-68. DMA Address Register Read/Write Command

7	6	5	4	3	2	1	0	
A7	A6	A5	A4	A3	A2	A1	A0	IN/OUT

7	6	5	4	3	2	1	0	
A15	A14	A13	A12	A11	A10	A9	A8	IN/OUT

7	6	5	4	3	2	1	0	
x	x	x	x	A19	A18	A17	A16	IN/OUT (byte)

Remark x: don't care

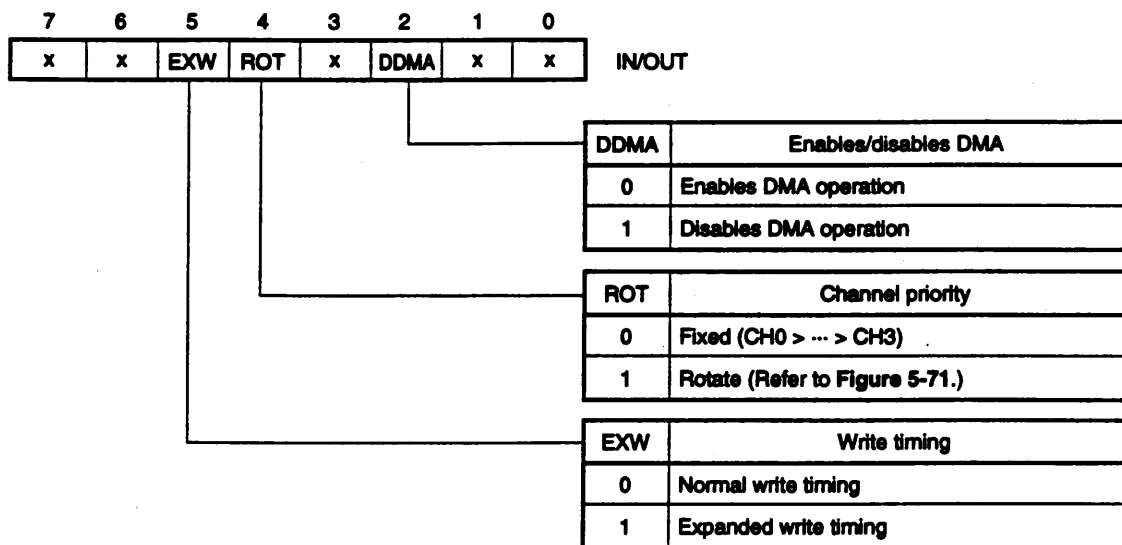
The address register of each channel consists of a base address register and a current address register. Both the base address register and current address register are allocated to the same address. Which between the base address and current address registers is selected for reading or writing is specified by the channel register.

The contents of the current address register are updated (± 2 for word transfer and ± 1 for byte transfer) each time DMA transfer executes. The base address register holds the current set value until a new address value is set, and the current set value is transferred to the current address register when auto initialization is executed.

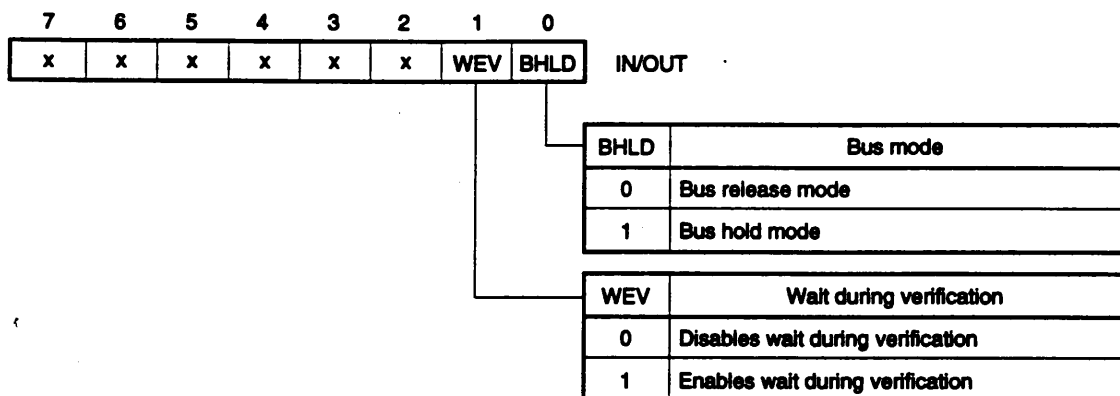
The lower 2 bytes (addresses 4H and 5H) of the address register can be read from or written to by using a word-length IN/OUT instruction. Use a byte-length IN/OUT instruction to read from or write to the highest 1 byte (address 6H).

(d) DMA device control register (DDC)

Figure 5-69. DMA Device Control Register Read/Write Command



Remark A small x refers to an optional 1 or 0.



Remark x: don't care

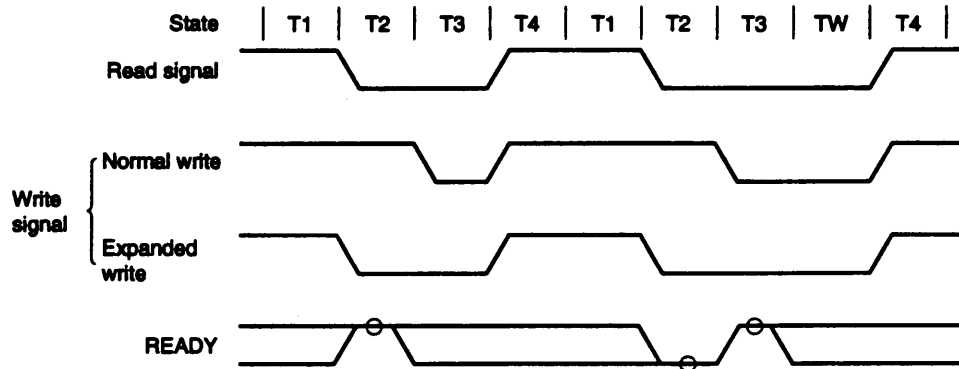
This register selects a mode common to all channels and performs control operations, such as enabling or disabling DMA operation. This register can be read from or written to by a word-length IN/OUT instruction.

(I) EXW bit

This bit sets the output timing of the write signal as shown in Figure 5-70.

- Normal write : Active in T3 and TW states
- Expanded write : Active in T2, T3, and TW states

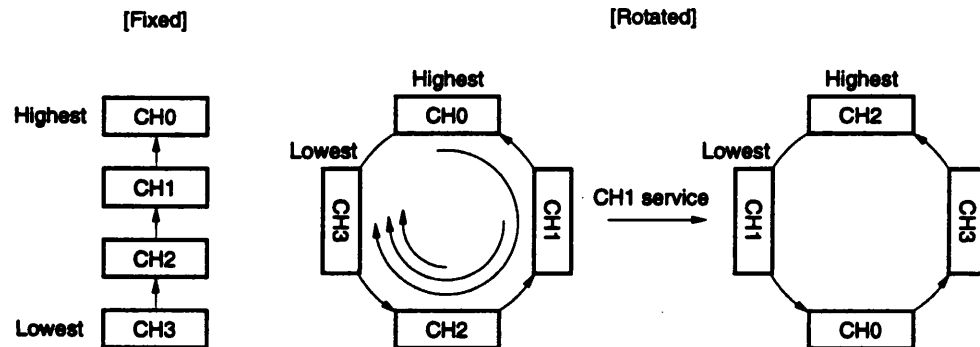
Figure 5-70. Expanded Write Timing



(II) ROT bit

This bit specifies rotating channel priority. When this bit is set, the rotating priority mode is specified. In the fixed priority mode, channel 0 always has the highest priority and channel 3 always has the lowest priority, as shown in Figure 5-71. In the rotating priority mode, however, the channel whose service has ended is given the lowest priority. This prevents a particular channel from monopolizing service.

Figure 5-71. DMA Priority



(III) DDMA bit

When this bit is set, the DMA operation is temporarily stopped. Internally, a bus request signal is not output to the BAU while DDMA = 1. If the DMA operation is enabled again later by clearing the DDMA bit to 0, the DMA operation can be resumed from where it was disabled. The DMAU can be programmed while the DMA operation is disabled.

(iv) WEV bit

This bit enables or disables the external $\overline{\text{READY}}$ signal and insertion of a wait state by the WCU during verification transfer.

(v) BHLD bit

This bit sets the bus mode for DMA transfer. The bus mode is used to hold or release the bus.

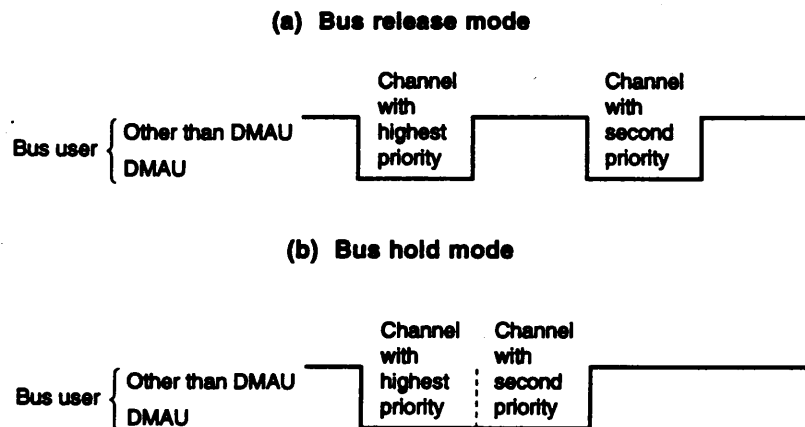
- **Bus release mode**

In the bus release mode, the bus control is returned to the CPU each time one service has ended. Consequently, even when two or more DMA requests are simultaneously issued, another bus cycle can be inserted in between the first service and the next service because the bus control is returned once. Because the bus control is returned each time one DMA cycle has ended, a quick response to a DMA request cannot be made.

- **Bus hold mode**

If another DMA request is issued after one service has ended in the bus hold mode, the bus master can hold the bus control and render the service. However, the same channel is not successively serviced. The bus hold mode enhances the transfer efficiency compared with the bus release mode.

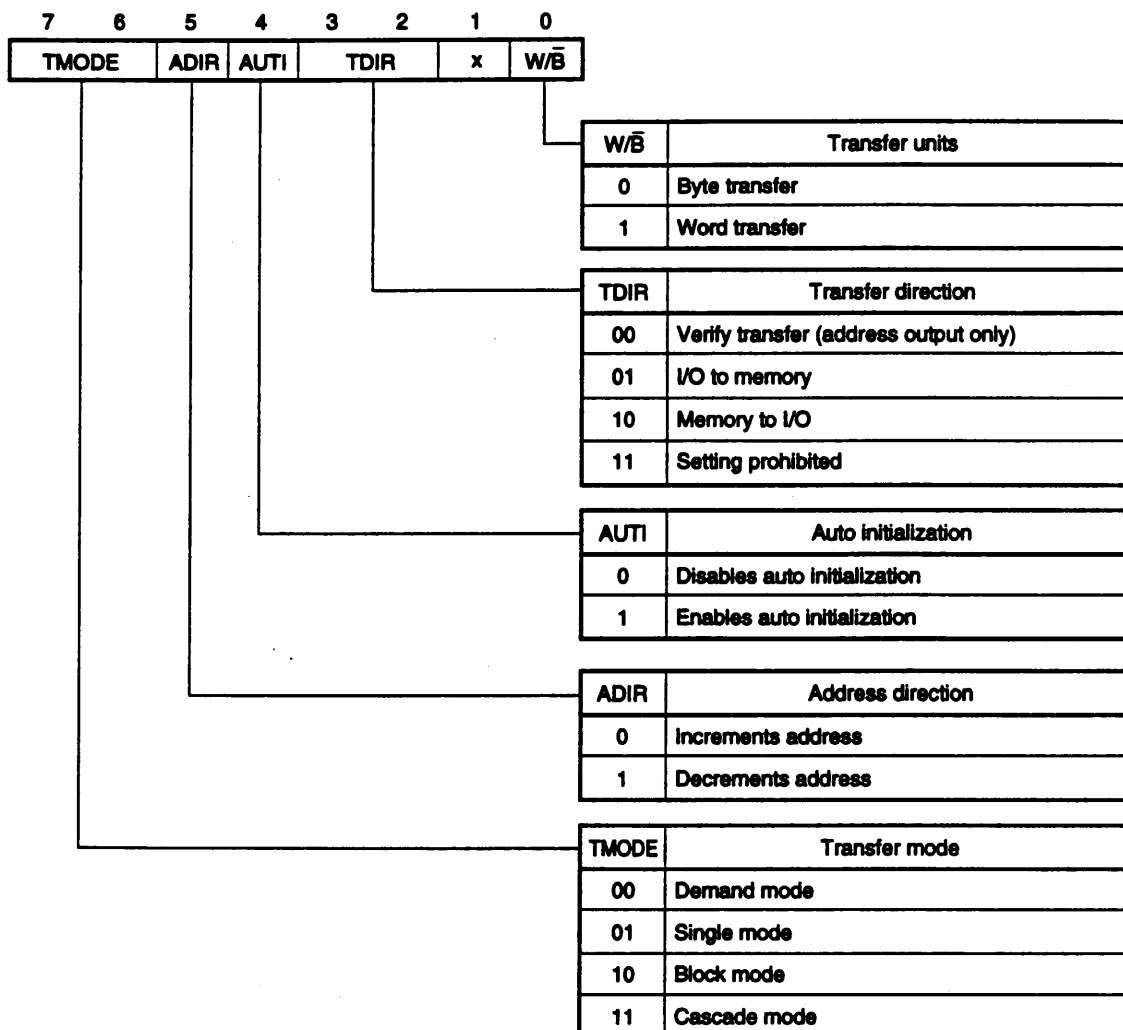
Figure 5-72. Difference In Bus Control Depending on Bus Mode



Remark It is assumed in the above figure that two requests are made.

(e) DMA mode control register (DMD)

Figure 5-73. DMA Mode Control Register Read/Write Command



Remark x: don't care

This register specifies the operation mode of each channel.

(I) TMODE bits

These bits specify the memory-to-I/O DMA transfer mode or the cascade mode when the DMA controller is cascade-connected externally.

For details, refer to sections 5.10.10 Transfer mode and 5.10.12 Cascade connection.

(II) ADIR bit

This bit specifies the direction in which the contents of the current address register are updated.

- Increments address
Word transfer: +2, byte transfer: +1
- Decrements address
Word transfer: -2, byte transfer: -1

(III) AUTI bit

By setting this bit, the auto initialization function is enabled. For details, refer to section 5.10.11 Auto Initialization.

(iv) TDIR bits

These bits specify the direction in which data is transferred between memory and I/O. The following three directions can be specified:

- Read transfer: memory to I/O
 $\overline{\text{MRD}}$ and $\overline{\text{IOWR}}$ are output at the same time.
- Write transfer: I/O to memory
 $\overline{\text{IORD}}$ and $\overline{\text{MWR}}$ are output at the same time.
- Verify transfer: Transfer is not performed.

Only an address is output and the read/write control signal is not output. This transfer is used for a verify operation which requires only the $\overline{\text{DMAAK}}$ signal (for CRC check of a floppy disk in a floppy disk controller (FDC)).

(v) $\text{W}/\overline{\text{B}}$ bit

This bit specifies whether DMA transfer is executed in word units or byte units. In word units, 2 bytes starting from an even address are treated as one word, and are transferred once by using the 16-bit data bus. If the transfer address is odd-numbered, the address is automatically decremented by one and word transfer begins. As a consequence, the last 1 byte cannot be transferred. Therefore, be sure to specify an even address as the transfer start address.

Table 5-27 shows how the contents of the current count register and current address register are updated when byte or word transfer is executed. Table 5-28 shows the statuses of the A0 and UBE signals during DMA transfer.

Table 5-27. Updating Current Registers

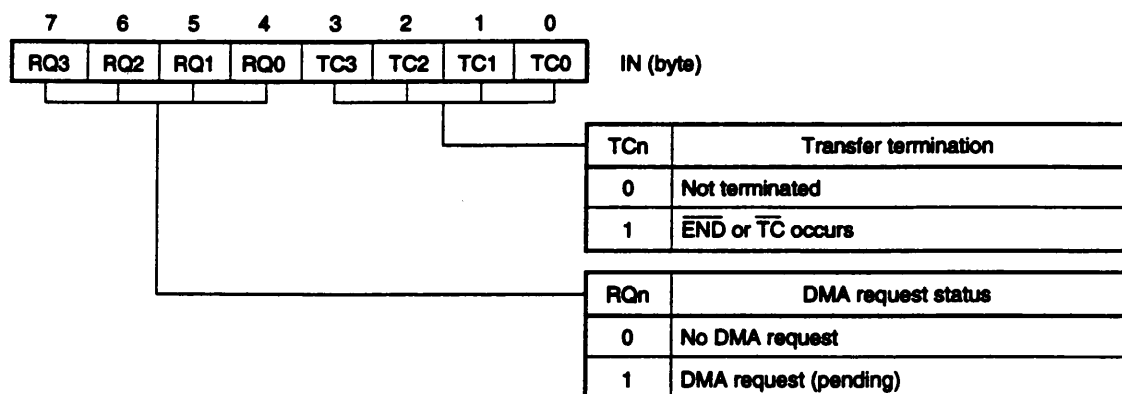
Register	Byte Transfer	Word Transfer
Current address	± 1	± 2
Current count	-1	-1

Table 5-28. A0 and $\overline{\text{UBE}}$ Signals during DMA Operation

Byte/Word	A0	$\overline{\text{UBE}}$	Data Bus Status
Byte	0	1	D0 to D7 valid
	1	0	D8 to D15 valid
Word	0	0	D0 to D15 valid

(f) DMA status register (DST)

Figure 5-74. DMA Status Register Read Command



This register maintains the status of DMA request of each channel and the status of the $\overline{\text{TC}}$ or $\overline{\text{END}}$ input.

(I) RQ0 through RQ3 bits

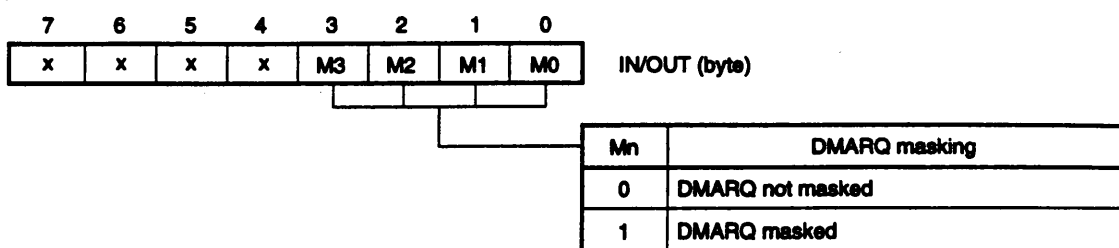
These bits indicate the DMA request status of each channel. They indicate the status of the DMARQ pin (1 if active) regardless of the mask status set by the mask register. Therefore, the DMA request that is masked and currently kept pending can be checked.

(II) TC0 through TC3 bits

These bits indicate the $\overline{\text{TC}}$ or $\overline{\text{END}}$ input generation status of each channel. These bits are automatically cleared each time this register is read.

(g) DMA mask register (DMK)

Figure 5-75. DMA Mask Register Read/Write Command



Remark x: don't care

The DMA request of any channel can be masked by setting this register. The mask bit of the corresponding channel is set when the terminal count occurs or when the DMA operation is forcibly terminated by the $\overline{\text{END}}$ input. However, the mask bit of the channel for which the auto initialization function is specified is not set.

Caution Mask the DMA request when the DMA operation is disabled.

(5) Cascade connection

Although the DMAU has four independent DMA channels, it may sometimes be necessary to increase the number of DMA channels. In this case, external DMA controllers such as the $\mu\text{PD71071}$ can be connected in cascade. (For details, refer to section 5.10.12 Cascade connection.)

5.10.9 μ PD71037 mode

The DMAU is set in the μ PD71037 mode when the DMAM bit of SCTL in the system I/O area is set to 1.

Note that the μ PD71071 mode is set immediately after reset.

(1) Addressing

The internal registers of the DMAU during the μ PD71037 mode are addressed by OPHA and DULA, which are set in the system I/O area, and by the address signals (A0 through A3 or A1 through A4). Table 5-29 shows the internal register addresses in the μ PD71037 mode.

In the μ PD71037 mode, the channel register (DCH) is not used because the channel to be set is specified when a command is issued.

Table 5-29. Addresses of DMAU's Internal Registers (μ PD71037 mode) (1/2)

	Higher Address	Lower Address				Register, Command	Operation
When IOAG = 1	A15 - A8 (OPHA)	A7 A6 A5 A4 (DULA)	0 0 0 0			Address register (channel 0)	Read/write
			0 0 1 0			Address register (channel 1)	Read/write
			0 1 0 0			Address register (channel 2)	Read/write
			0 1 1 0			Address register (channel 3)	Read/write
			0 0 0 1			Count register (channel 0)	Read/write
			0 0 1 1			Count register (channel 1)	Read/write
			0 1 0 1			Count register (channel 2)	Read/write
			0 1 1 1			Count register (channel 3)	Read/write
			1 0 0 0			Status register/command register	Read/write
			1 0 0 1			Request register	Write
			1 0 1 0			Single mask register	Write
			1 0 1 1			Mode register	Write
			1 1 0 0			Clear byte pointer F/F	Write
			1 1 0 1			Initialize	Write
			1 1 1 0			Clear mask register	Write
			1 1 1 1			All mask registers	Write
When IOAG = 0	A15 - A8 (OPHA)	A7 A6 A5 (DULA)	0 0 0 0	A0		Address register (channel 0)	Read/write
			0 0 1 0			Address register (channel 1)	Read/write
			0 1 0 0			Address register (channel 2)	Read/write
			0 1 1 0			Address register (channel 3)	Read/write
			0 0 0 1			Count register (channel 0)	Read/write
			0 0 1 1			Count register (channel 1)	Read/write
			0 1 0 1			Count register (channel 2)	Read/write
			0 1 1 1			Count register (channel 3)	Read/write
			1 0 0 0			Status register/command register	Read/write
			1 0 0 1			Request register	Write
			1 0 1 0			Single mask register	Write
			1 0 1 1			Mode register	Write
			1 1 0 0			Clear byte pointer F/F	Write
			1 1 0 1			Initialize	Write
			1 1 1 0			Clear mask register	Write
			1 1 1 1			All mask registers	Write

Table 5-29. Addresses of DMAU's Internal Registers (μ PD71037 mode) (2/2)

	Higher Address	Lower Address						Register, Command		Operation
—	1 1 1 1 1 1 1 1	1	1	1	0	0	0	0	BSEL	Read/write
	1 1 1 1 1 1 1 1	1	1	1	0	0	0	0	1	BADR
When IOAG = 1	A15 - A8 (OPHA)	A7	A6	A5	A4	A3	A2	BNKn (BSEL)	Bank register (channel n)	Read/write
When IOAG = 0	A15 - A8 (OPHA)	A7	A6	A5	A4	A3	BNKn (BSEL)	A0	Bank register (channel n)	Read/write

Remarks 1. The addresses of BSEL and BADR are FFE0H and FFE1H regardless of the SCTL IOAG bit, OPHA, and DULA.

2. n = 0 to 3

(2) Commands in μ PD71037 modeTable 5-30. Commands in μ PD71037 Mode

Address ^{Note 1}				Operation	F/F ^{Note 3}	Command Name ^{Note 4}
A3	A2	A1	A0			
(A4)	(A3)	(A2)	(A1)			
0 channel ^{Note 2 0}				Read	0	Read current address register (lower byte) (DRCAn)
					1	Read current address register (higher byte) (DRCAn)
				Write	0	Write base & current address register (lower byte) (DWBCAn)
					1	Write base & current address register (higher byte) (DWBCAn)
0 channel ^{Note 2 1}				Read	0	Read current count register (lower byte) (DRCCn)
					1	Read current count register (higher byte) (DRCCn)
				Write	0	Write base & current count register (lower byte) (DWBCCn)
					1	Write base & current count register (higher byte) (DWBCCn)
1	0	0	0	Read		Read status register (DRST)
				Write		Write command register (DWC)
1	0	0	1	Write		Write request register (DWRQ)
1	0	1	0	Write		Write single mask register (DWSM)
1	0	1	1	Write		Write mode register (DWM)
1	1	0	0	Write		Clear byte pointer F/F (DCBP)
1	1	0	1	Write		Initialize (DINT)
1	1	1	0	Write		Clear mask register (DCM)
1	1	1	1	Write		Write all mask register (DWAM)

Notes 1. Address in parentheses apply when IOAG = 0

2. If these bits = 00, channel 0 is selected.

If these bits = 01, channel 1 is selected.

If these bits = 10, channel 2 is selected.

If these bits = 11, channel 3 is selected.

3. F/F is a byte pointer flip-flop. This flip-flop specifies the higher or lower byte when the address register and count register are accessed. This flip-flop is cleared to 0 at reset or when the clear byte pointer F/F command is executed, and inverted each time the address register and count register have been read or written. Therefore, read or write the lower and higher bytes of the address register and count register in succession.

4. n = 0 to 3 (channel number)

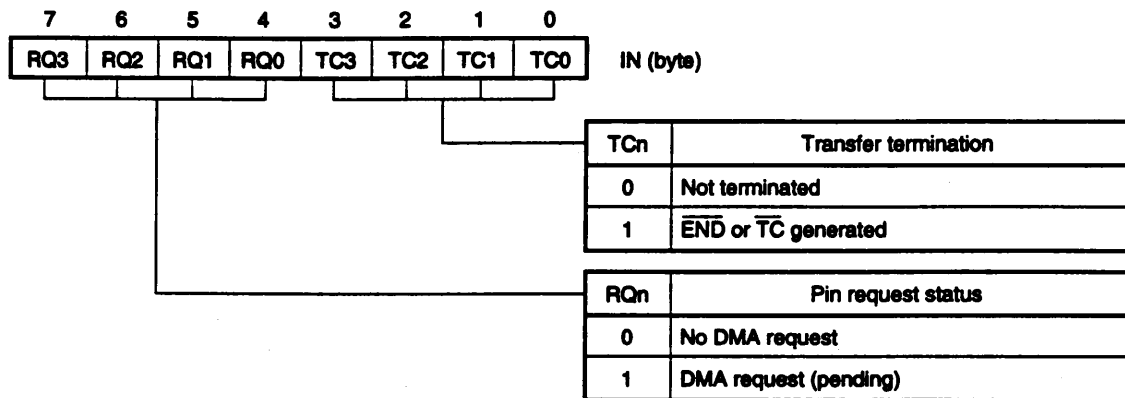
(3) Details of the μ PD71037 mode command

The detailed format of each control command is shown below. The address is indicated by the A0 through A3 signals when the IOAG bit of SCTL is 1 and by the A1 through A4 signals when the IOAG bit is 0.

(a) Read status register (DRST)

This register maintains information on each channel on the DMA request, terminal count generation and $\overline{\text{END}}$ signal input.

Figure 5-76. Read Status Register (DRST)



(i) RQ0 through RQ3 bits

These bits indicate the DMA request status (DMARQ pin input). They are set as long as the DMARQ input of the same channel is active, even if the corresponding channel is masked. By sampling these bits, a DMA request that is masked and kept pending can be detected.

(ii) TC0 through TC3 bits

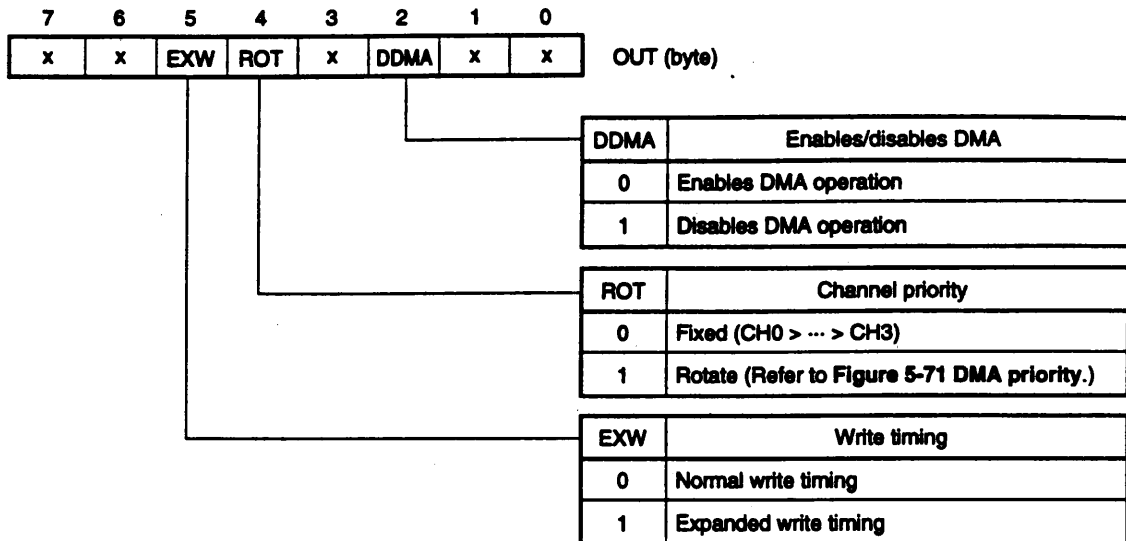
These bits indicate the $\overline{\text{END}}$ input or terminal count generation status of each channel.

If a terminal count occurs within a channel, or if the $\overline{\text{END}}$ signal is input to a channel from an external source, the bit of the relevant channel will be set. Each time the contents of DRST are read, these bits will be reset.

(b) Write command register (DWC)

This register determines DMA transfer enable/disable, the priority and write timing.

Figure 5-77. Write Command Register (DWC)

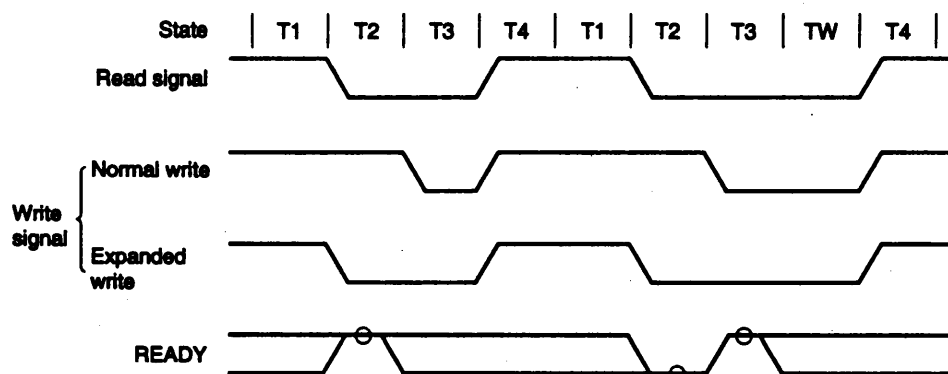


Remark x: don't care

(i) EXW bit

When this bit is set, the DMAU outputs the write signal at the same timing as the read signal (expanded write). Figure 5-78 shows the expanded write timing.

Figure 5-78. Expanded Write Timing



(II) ROT bit

When this bit is set, the priority of the DMA channel is determined in a rotating nest mode.

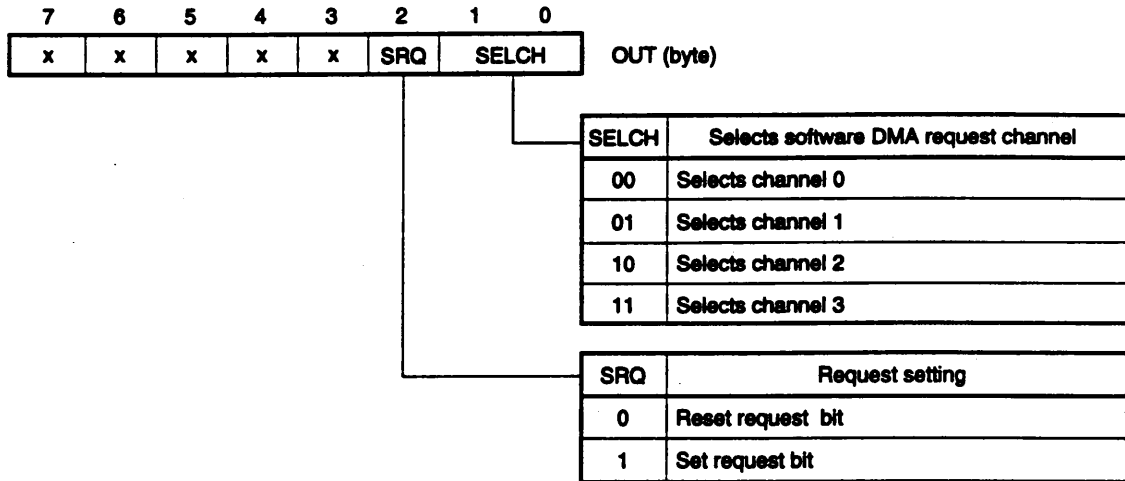
(III) DDMA bit

This bit disables the DMA operation. While this bit is set, the DMAU will not output the HLDRQ signal to the host CPU even if it has received the valid DMA request. The DDMA bit prevents wrong DMA transfer from being executed while the program of the DMAU is being executed.

(c) Write request register (DWRQ)

This register specifies control of software DMA request.

Figure 5-79. Write Request Register (DWRQ)



Remark x: don't care

(i) SRQ bit

This bit controls the software DMA request of the channel selected by bits 0 and 1. When this bit is 1, the request bit of the specified channel is set; when it is 0, the request bit is reset.

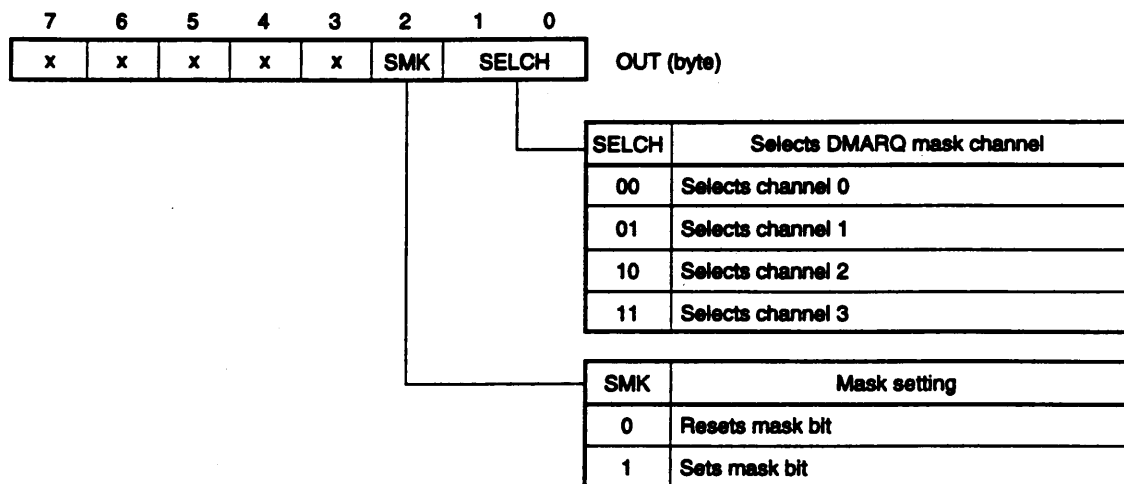
(ii) SELCH bit

This bit specifies a channel subject to software DMA request control.

(d) Write single mask register (DWSM)

This register controls set or reset of the mask bit of each channel.

Figure 5-80. Write Single Mask Register (DWSM)



Remark x: don't care

(I) SMK bit

This bit masks the DMA request (DMARQ pin input) of the channel specified by bits 0 and 1. When this bit is 1, the DMA request is masked; when it is 0, the DMA request is unmasked.

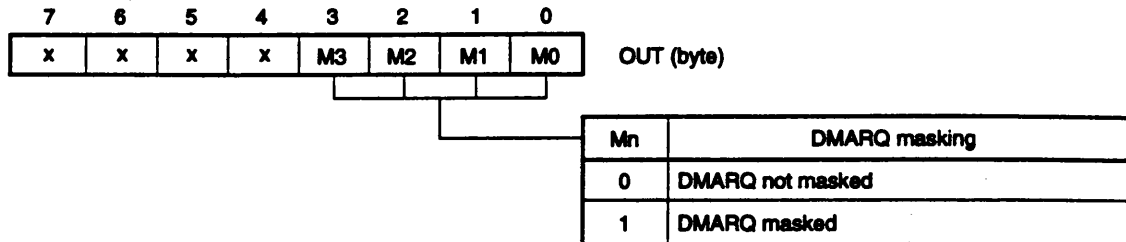
(II) SELCH bit

These bits select a channel subject to DMA request masking.

(e) Write all mask register (DWAM)

This register specifies set or reset of the mask bits of all channels.

Figure 5-81. Write All Mask Register (DWAM)



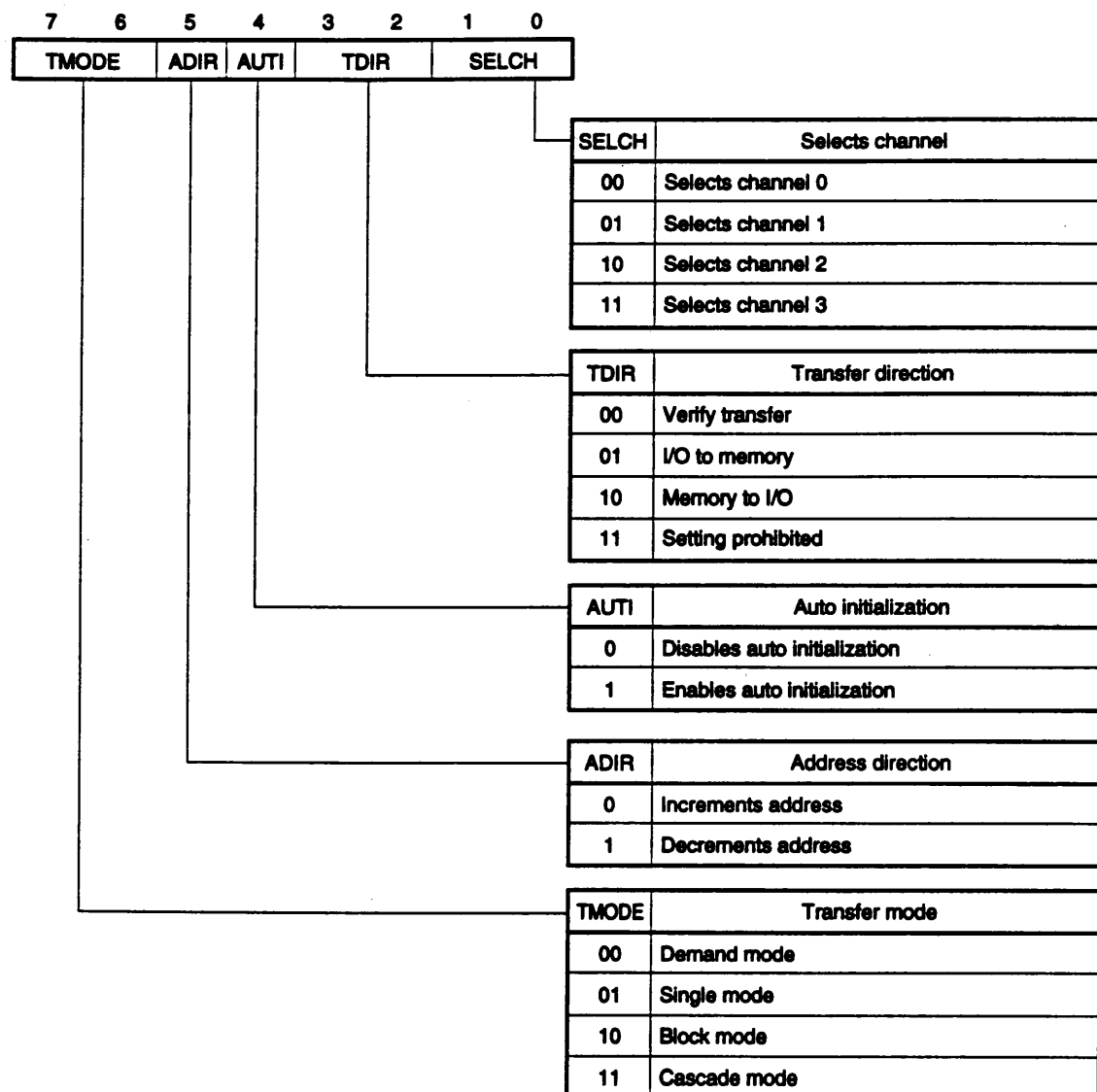
Remark x: don't care

The M0 through M3 bits specify whether each of the four DMA channels is masked or unmasked. When any of these bits is set to 1, the corresponding channel is masked, and the DMA request of that channel (DMARQ pin input) is disabled. When the bit is cleared to 0, the channel is unmasked.

(f) Write mode register (DWM)

This register specifies the mode of DMA transfer for each channel.

Figure 5-82. Write Mode Register (DWM)



(i) TMODE bits

These bits specify a DMA transfer mode.

For details, refer to **section 5.10.10 Transfer mode**.

When connecting an external DMA controller to the DMAU in cascade, specify the cascade mode by using these bits (refer to **section 5.10.12 Cascade connection**).

(ii) ADIR bit

This bit specifies whether the value of the current address register will increment or decrement when it is to be updated. If this bit is cleared to 0, the value of the current address register will increment by one each time 1 byte has been transferred; if it is set to 1, the register value will decrement by one.

(iii) AUTI bit

By setting this bit, the auto initialization function is enabled.

For details, refer to **section 5.10.11 Auto initialization**.

(iv) TDIR bits

These bits specify a transfer direction.

(v) SELCH bits

These bits specify a DMA channel for which modes are specified by bits 2 through 7.

(g) Clear byte pointer F/F (DCBP), initialize (DINT), and clear mask register (DCM)

The data written by the three commands, DCBP, DINT, and DCM have no meaning. Any value that is written by a byte-length OUT instruction makes the command valid.

Command	Address	Function
DCBP	CH	Clears byte pointer F/F. Before accessing the address/count register, issue this command.
DINT	DH	Initializes the DMAU (Because same state as that during RESET input).
DCM	EH	Clears masks of all channels and enables acknowledgement of DMA requests.

(4) Expanded DMA address

The V40HL and V50HL have bank registers (BNKR0 through BNKR3) that specify expanded DMA addresses A16 through A19 of channels 0 through 3 in the μ PD71037 mode. The I/O address to read/write these registers in the μ PD71037 mode is specified by OPHA and bank address register (BADR) in the system I/O area.

(a) I/O addresses of bank registers

The higher 8 bits (A8 through A15) of the I/O addresses of the bank registers (BNKR0 through BNKR3) are set by OPHA, and the lower 8 bits (A0 through A7) are set by the bank address register (BADR). The bits that can be set differs depending on the value of the IOAG bit of SCTL, such as the other on-chip peripheral relocation registers.

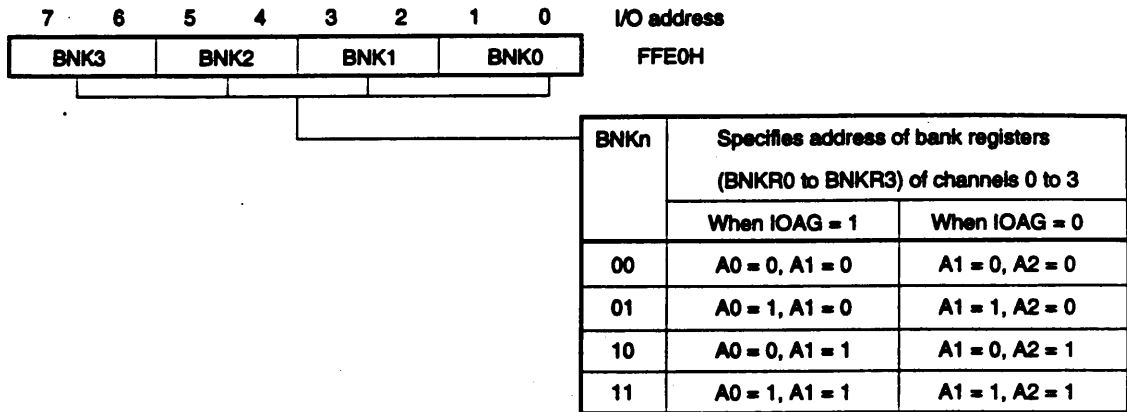
Figure 5-83. Bank Address Register (BADR)

(a) When IOAG = 1								I/O address
7	6	5	4	3	2	1	0	
A7	A6	A5	A4	A3	A2	—	—	FFE1H

(b) When IOAG = 0								I/O address
7	6	5	4	3	2	1	0	
A7	A6	A5	A4	A3	—	—	A0	FFE1H

The I/O addresses of the bank registers are determined by the addresses set to registers OPHA and BADR. In addition, the bank registers selected by A0 and A1 (when IOAG = 1) or A1 and A2 (when IOAG = 0) are programmable.

Figure 5-84. Bank Select Register (BSEL)



(b) Bank registers (BNKR0 to BNKR3)

Figure 5-85. Bank Registers (BNKR0 to BNKR3)

7	6	5	4	3	2	1	0
0	0	0	0	A19	A18	A17	A16

Expanded DMA address bits A16 through A19 are set to the bits 0 through 3 of the bank register. Be sure to clear the bits 4 through 7 of the bank register to 0 at this time. The contents of the bank register are undefined after the μ PD71071 mode has been used. The bank registers are enabled by OPSEL in the system I/O area to be used by the DMAU, and exist in the I/O space only in the μ PD71037 mode. They do not exist on the internal I/O space in the μ PD71071 mode. However, registers BADR and BSEL exist on the system I/O space regardless of the mode of the DMAU. The bank registers use the higher 8 bits of the address register in the μ PD71071 mode.

(c) Carry transfer control

A carry that occurs from A15 to A16 is controlled by the CE bit of SCTL in the system I/O area when the bank registers are used.

CE	Function
0	Carry is not transferred to A16 in μ PD71037 mode
1	Carry is transferred to A16 in μ PD71037 mode

5.10.10 Transfer mode

The transfer mode in the μ PD71071 mode is specified by the TMODE bit of DMD, and the transfer mode in the μ PD71037 mode is specified by the TMODE bit of the write mode register command. The operation in each transfer mode is the same in both the μ PD71071 and μ PD71037 modes.

However, only the bus release mode is available in the μ PD71037 mode as a bus mode.

Table 5-31 shows the types of transfer modes and transfer end conditions.

Table 5-31. Transfer Modes and DMA Service End Conditions

Transfer Mode	Service End Condition
Single mode	Transfer end for each 1 byte/1 word
Demand mode	<ul style="list-style-type: none"> • External $\overline{\text{END}}$ input • Occurrence of terminal count • Withdrawal of DMARQ of service channel • Occurrence of DMARQ with higher priority (bus hold mode)^{Note}
Block mode	<ul style="list-style-type: none"> • External $\overline{\text{END}}$ input • Occurrence of terminal count

Note If a DMARQ with the higher priority is generated, DMA transfer under execution is temporarily stopped and the DMA transfer with the higher priority is executed. When this DMA transfer has been completed, and if DMARQ is active, the first DMA transfer is resumed.

If two or more DMA requests are issued, the operation to be performed slightly differs depending on the combination of the transfer mode and the bus mode. Table 5-32 shows the operations in each combination of the transfer mode and bus mode.

Table 5-32. DMA Operation for Each Combination of Transfer Mode and Bus Mode

Transfer mode \ Bus Mode	Bus Release	Bus Hold
Single	<pre> graph TD A[1 byte/word DMA transfer] --> B([Idle cycle]) </pre>	<pre> graph TD A[1 byte/word DMA transfer] --> B{BUSAK} B -- No --> C([Idle cycle]) B -- Yes --> D{Other channel DMA} D -- No --> C D -- Yes --> E([Service of other channel]) </pre>
Demand	<pre> graph TD A[1 byte/word DMA transfer] --> B{END or TC} B -- Yes --> A B -- No --> C{BUSAK} C -- No --> D{Service channel DMA} D -- Yes --> B D -- No --> E([Idle cycle]) </pre>	<pre> graph TD A[1 byte/word DMA transfer] --> B{BUSAK} B -- No --> C([New service]) B -- Yes --> D{END or TC} D -- Yes --> E{Other channel DMA} E -- Yes --> F([Service of other channel]) E -- No --> G{any DMA} G -- No --> H([Idle cycle]) G -- Yes --> C </pre>
Block	<pre> graph TD A[1 byte/word DMA transfer] --> B{END or TC} B -- Yes --> C([Idle cycle]) B -- No --> D{BUSAK} D -- No --> B D -- Yes --> A </pre>	<pre> graph TD A[1 byte/word DMA transfer] --> B{END or TC} B -- Yes --> C{Other channel DMA} C -- Yes --> D{BUSAK} D -- No --> B D -- Yes --> E{Channel with priority in block mode} E -- Yes --> F([Idle cycle]) E -- No --> G([Service of other channel]) C -- No --> G </pre>

Remark BUSAK is the internal signal of the V40HL and V50HL, and is the HLDAA signal dedicated to the DMAU.

5.10.11 Auto Initialization

The auto initialization function automatically initializes the address and count value when the value of the current count register reaches 0 when a terminal count (\overline{TC}) occurs, or when the \overline{END} signal is input. This function can be set for each channel by DMD (in the μ PD71071 mode) or DWM (in the μ PD71037 mode).

If the \overline{END} signal is input or if a \overline{TC} occurs during service of a channel for which the auto initialization function is specified, the following operations are performed:

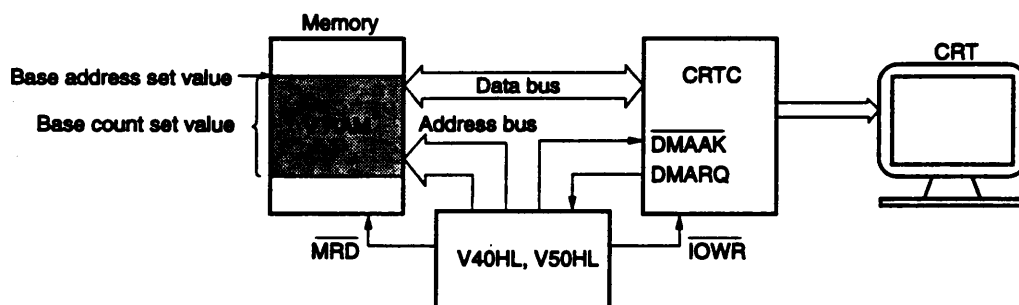
- The contents of the base address register and the base count register are automatically transferred to the current address register and current count register, respectively.
- The relevant bit of the mask register is not set. (The relevant bit of the mask register of the channel for which auto initialization is not set will set).

The auto initialization function is especially useful for executing the DMA transfers indicated below:

(1) Repetitive Input/output of a specified memory area

DMA transfer between a CRT controller and V-RAM can be cited as an example. In this case, if the auto initialization function is executed with the base register and current register set to the same initial value, DMA transfer will execute repeatedly without subsequent CPU operation.

Figure 5-86. Auto Initialization Application Example 1

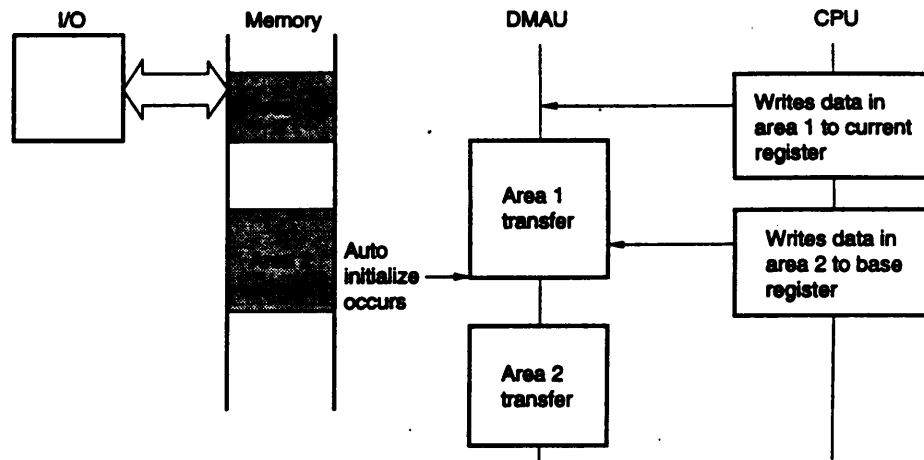


(2) Continuous transfer to more than one memory area

Because writing to the base register only is required for writing to the address/count register, the auto initialization function can be used as follows:

During the bus release mode, when the DMAU operates in the single or demand mode, it may be necessary to transfer two or more contiguous or non-contiguous memory areas. In this case, if information in the next area is set to the base register while one area is being transferred, information in the next area is transferred to the current register using the auto initialization function when a terminal count of the area being transferred occurs. In this way, transfer can be executed continuously. If the auto initialization function is used in this way, the CPU will have enough time to write data to the DMAU.

Figure 5-87. Auto Initialization Application Example 2



5.10.12 Cascade connection

The DMAU can directly handle up to four channels. Any of these channels can be connected to an external DMA controller if so specified using DMD, and this will increase the number of channels. Cascade connection is implemented as follows:

- ★ (1) Connect the HLDRQ and HLDK pins of the μ PD71071 and μ PD71037 to the DMARQ and $\overline{\text{DMAK}}$ pins of any channel of the V40HL and V50HL.
- ★ (2) Set the channel of the μ PD71071 and μ PD71037-connected V40HL and V50HL to the cascade mode.

During the DMA service of the channel set in the cascade mode (cascade channel), the DMAU will always operate in the bus release mode, regardless of the setting of DDC. Consequently, requests from the other channels are ignored during the service of the cascaded channel. When the service of the cascaded channel has ended and the DMAU enters the idle cycle, it will return to the bus mode that was set by DDC.

- ★ The cascade channel only grants the bus control to the connected μ PD71071 and μ PD71037 and does not output an address or control signal.

Figure 5-88. Example of Cascade Connection

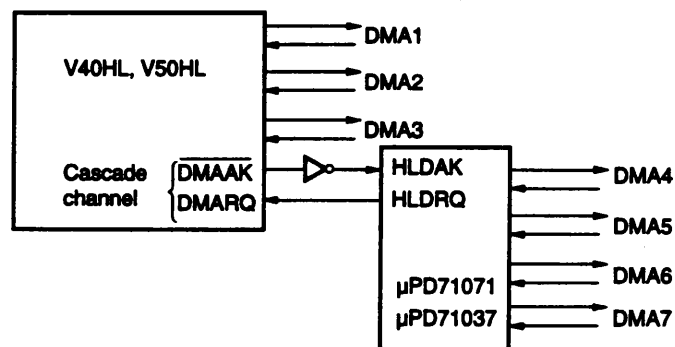
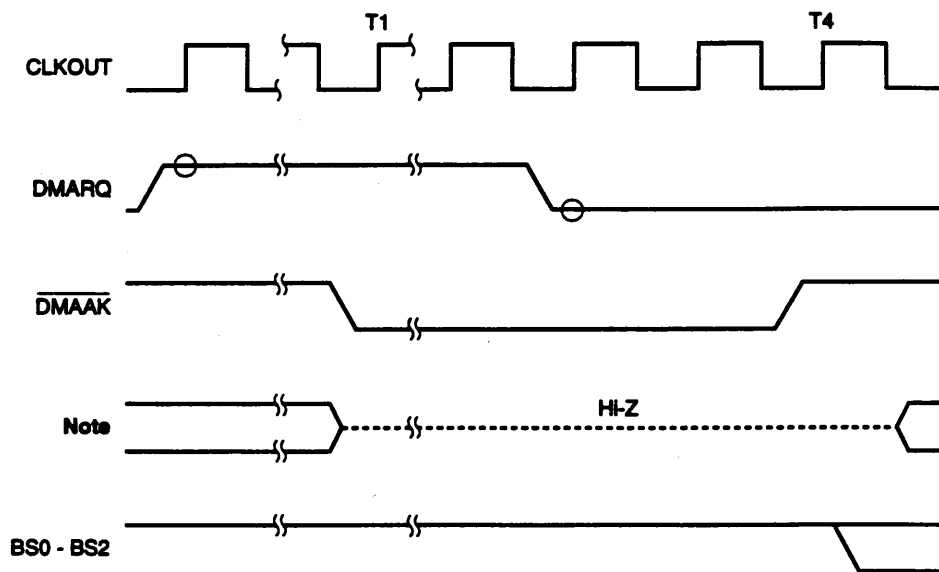


Figure 5-89. Example of Cascade Timing



Note $\overline{AD0}$ to $\overline{AD7}$ (V40HL), A8 to A15 (V40HL), $\overline{AD0}$ to $\overline{AD15}$ (V50HL), A16 to A19, \overline{BUFEN} , $\overline{BUFR/W}$, \overline{MRD} , \overline{MWR} , \overline{IOWR} , $\overline{BUSLOCK}$

Remark A circle indicates the sampling timing.

Table 5-33. Pin Status During Cascade Connection

Pin Name	I/O	Cascade Connection
AD0 to AD15 ^{Note 1}	3-state I/O	Hi-Z
AD0 to AD7 ^{Note 2}	3-state I/O	Hi-Z
A8 to A15 ^{Note 2}	3-state output	Hi-Z
A16/PS0 to A19/PS3	3-state output	Hi-Z
REFRQ	Output	H
HLD _{AK}	Output	L
RESOUT	Output	L
$\overline{\text{MRD}}$	3-state output	Hi-Z
$\overline{\text{MWR}}$	3-state output	Hi-Z
$\overline{\text{IORD}}$	3-state output	Hi-Z
$\overline{\text{IOWR}}$	3-state output	Hi-Z
ASTB	Output	L
$\overline{\text{UBE}}$ ^{Note 1}	3-state output	Hi-Z
High ^{Note 2}	3-state output	Hi-Z
BUSLOCK	3-state output	Hi-Z
BUFR/W	3-state output	Hi-Z
BUFEN	3-state output	Hi-Z
CLKOUT	Output	○
BS0 to BS2	3-state output	H
QS0, QS1	Output	○
TOUT2	Output	○
INTAK	Output	H
$\overline{\text{SRDY}}$		○
TOUT1		○
DMAAK3		○
TxD	Output	○
$\overline{\text{DMAAK0}}$ to $\overline{\text{DMAAK2}}$		○
END/TC	I/O	H

Note 1. V50HL only

2. V40HL only

Remark H: high level, L: low level, H/L: high or low level

Hi-Z: high impedance,

○: not fixed (outputs valid value)

5.10.13 Status transition diagram

The following figures show the status transition of the DMAU that applies to both the μ PD71071 and μ PD71037 modes. In the μ PD71037 mode, however, the bus mode is always fixed to the bus release mode.

Figure 5-90. Idle Cycle

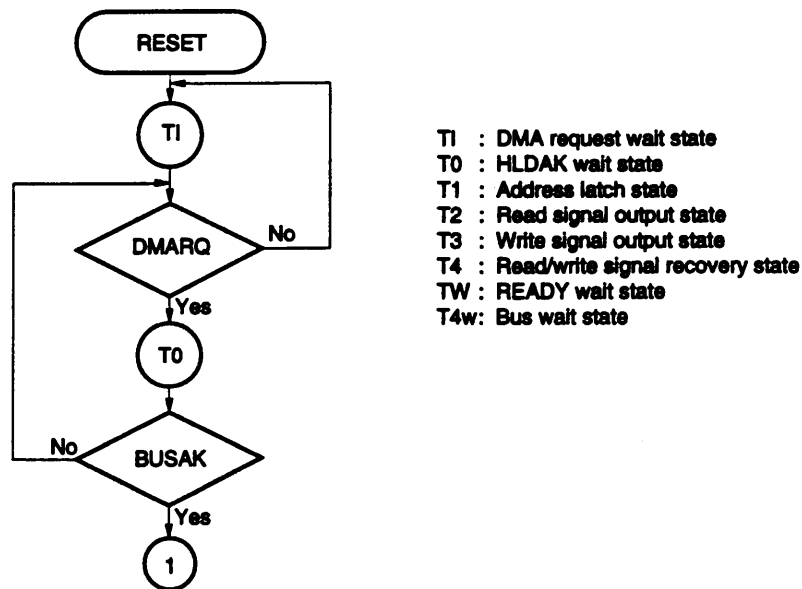


Figure 5-91. DMA Cycle (1/4)

(a) Cascade mode

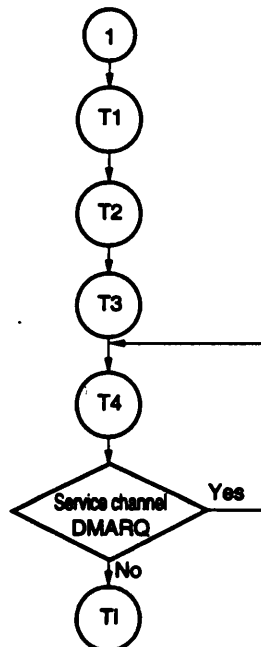
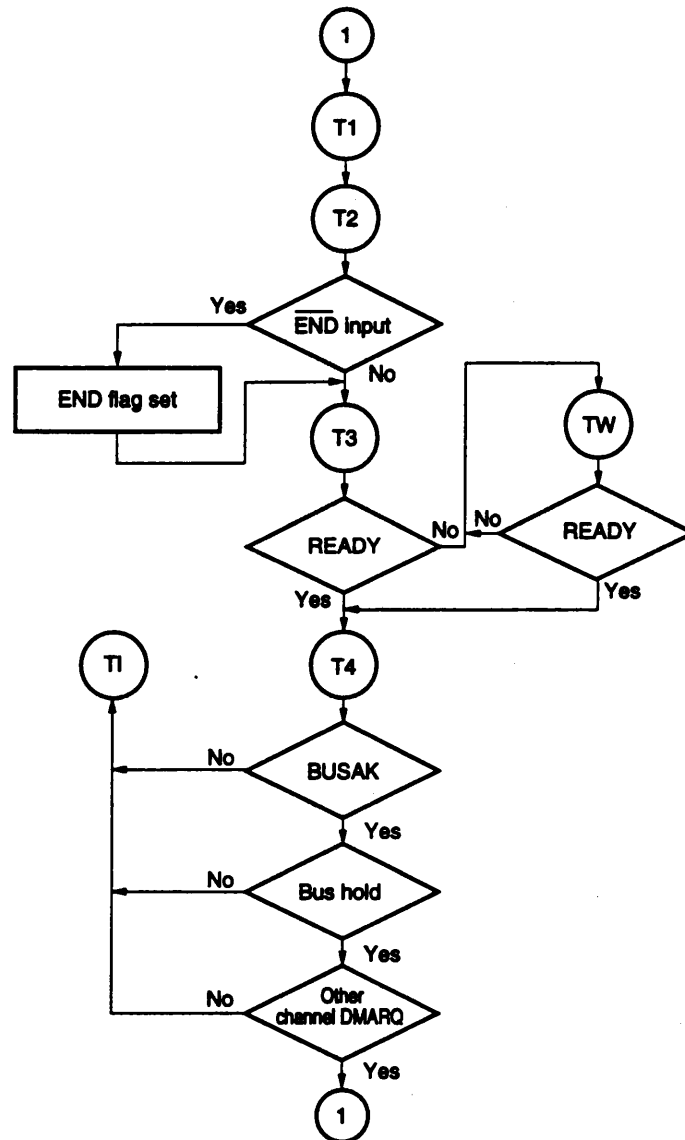


Figure 5-91. DMA Cycle (2/4)

(b) Single mode



(c) Demand mode

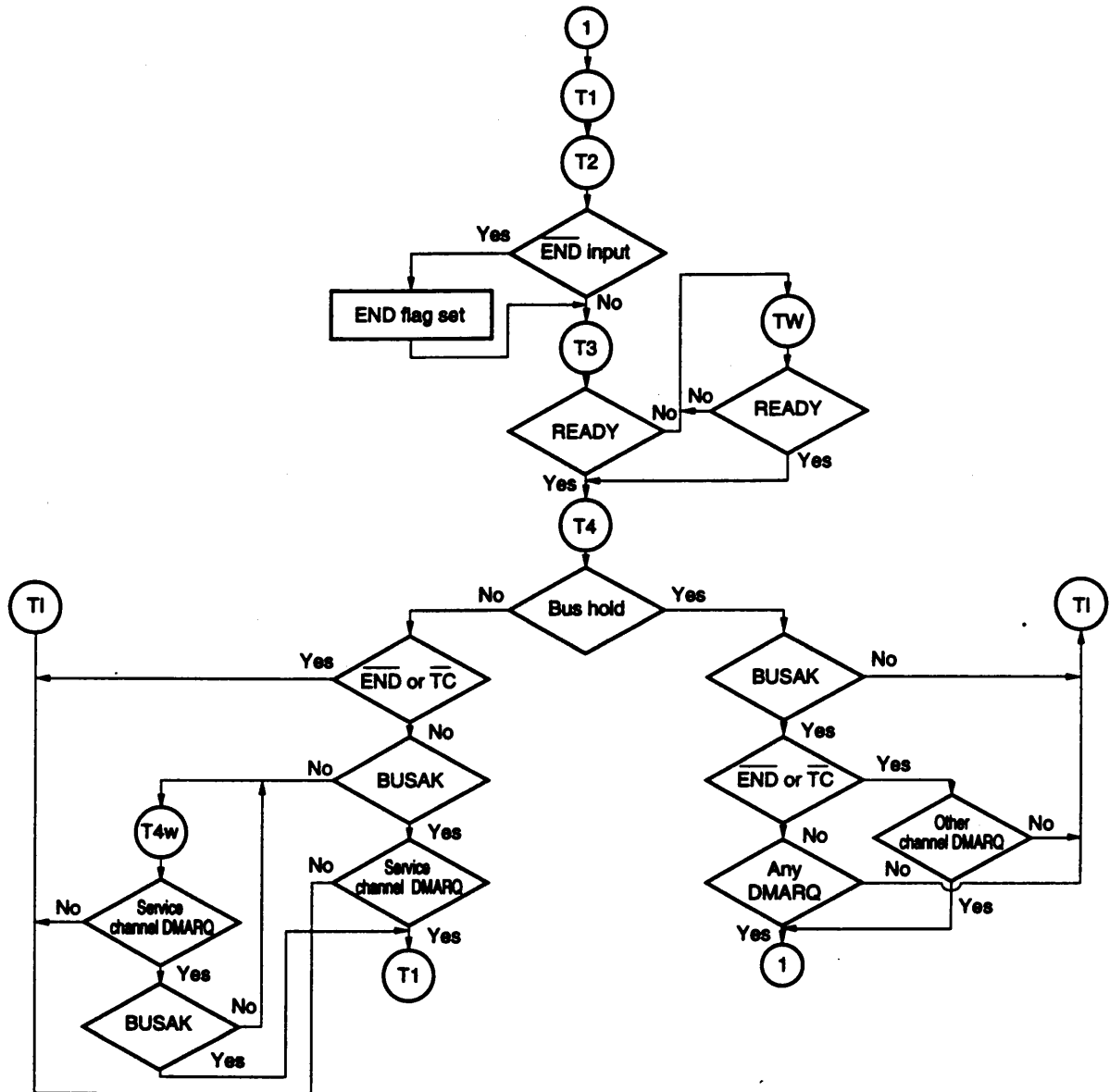
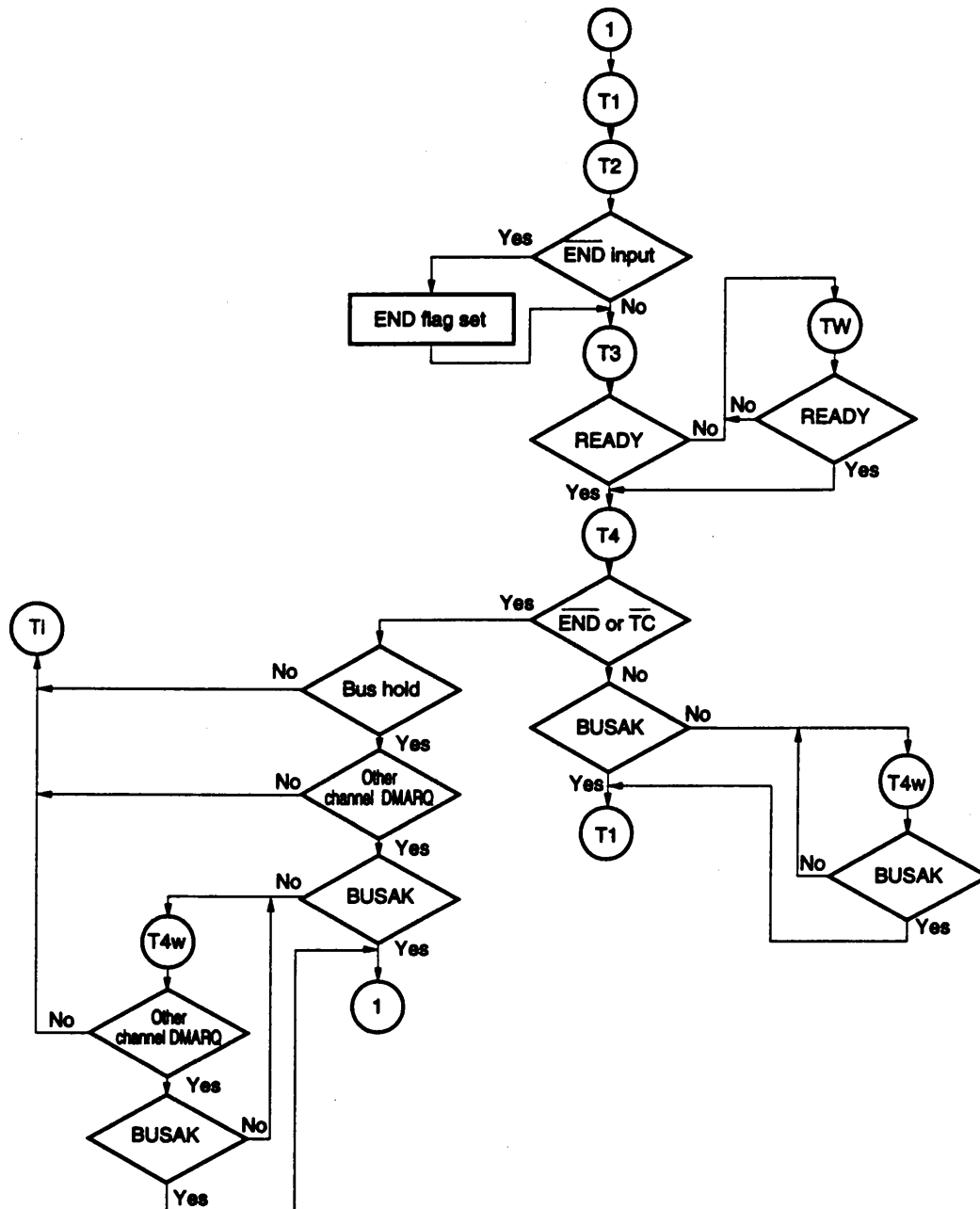


Figure 5-91. DMA Cycle (4/4)

(d) Block mode



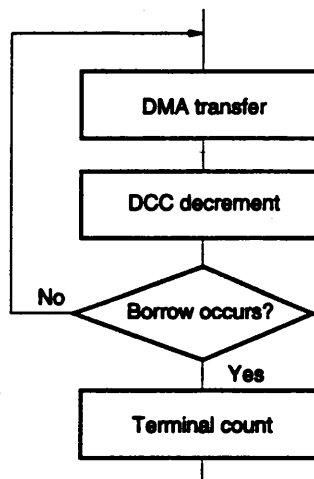
5.10.14 Precautions

(1) Terminal count and number of times of transfer

DCC is decremented each time DMA transfer has been executed to count the number of times of DMA transfer. The terminal count occurs when a borrow occurs, rather than when the value of DCC has reached to 0, as shown by the flowchart in Figure 5-92.

Therefore, the number of times DMA transfer is executed until the terminal count occurs and \overline{TC} is output is one more than the value set to DCC.

Figure 5-92. Terminal Count Generation



(2) Bus wait operation

Usually, the DMAU has the highest level of bus control, and its DMA service does not stop due to a request for the bus from the other bus masters. However, when the REFU is used, and if a refresh request is kept waiting for a long time, the level of bus control of the REFU becomes higher than that of the DMAU, and the REFU requests that the DMAU release the bus.

In this case, the DMAU temporarily stops the DMA service as soon as the DMA cycle currently executing has completed and relinquishes the bus to the REFU. The REFU executes the refresh requests which have been kept pending four or five cycles and then returns bus control to the DMAU again.

In this way, the DMA service may stop temporarily due to the REFU.

This bus wait operation, however, does not affect the DMA function at all, except that DMA service stops temporarily.

(3) Masking DMA requests

Be sure to mask the DMA channels by using the DMA mask register (DMK) when the DMA operation is disabled. If two or more channels are executing DMA, and if an attempt is made to change the contents of DMK with the DMA operation enabled, a terminal count (TC) occurs after the register contents have been read. If the bit corresponding to DMK is set, it will be cleared again by a write access of the program.

[MEMO]

CHAPTER 6 INTERNAL/EXTERNAL CONTROL FUNCTIONS

6.1 Wait Function

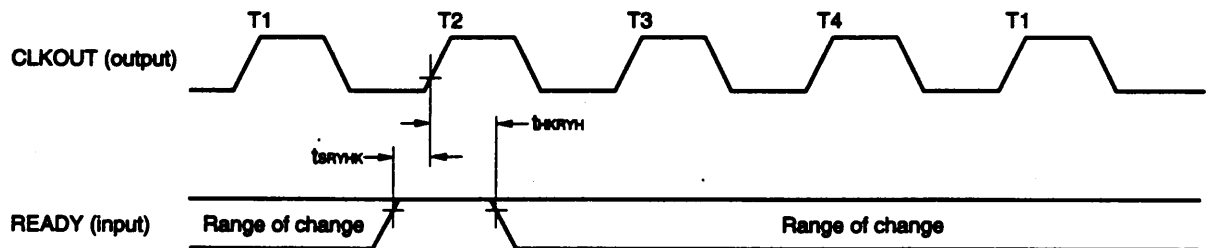
The V40HL/V50HL has a wait function that can extend the bus cycle in order to connect a low-speed memory and I/O. This wait function is implemented by using the READY pin or wait control unit (WCU).

6.1.1 Wait function by using READY pin

Wait state(s) (TW) can be inserted in each bus cycle of the memory cycle, external I/O cycle, DMA cycle, and refresh cycle by controlling the READY pin. However, TW cannot be inserted in the internal I/O cycle (the READY input is ignored).

When no TW is inserted in a bus cycle (no-wait status), the READY pin must be at the high level (ready) in the T2 state of the bus cycle. In the subsequent T3 and T4 states, the READY pin is not sampled, and the bus cycle ends. Figure 6-1 shows the timing in the no-wait status.

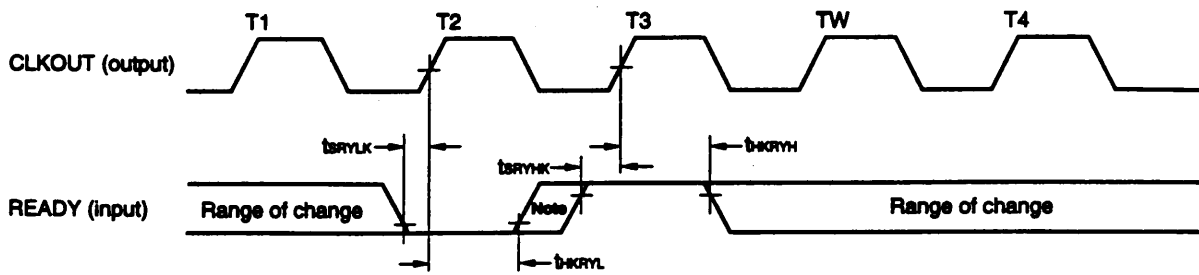
Figure 6-1. Ready Timing (no-wait status)



To insert a TW in a bus cycle, the READY pin must be at the low level (no-ready status) in the T2 state of the bus cycle. To insert another TW, the READY pin must be at the low level in the subsequent T3 state. To insert a third TW, the READY pin must be at the low level in the subsequent TW state. If the READY pin is at the high level in the T3 or TW state, it is assumed that insertion of the TW state has ended, and the bus cycle ends in the subsequent TW or T4 state. Consequently, the TW state is inserted two states after the low level of the READY signal has been sampled. So that execution exits from the TW state and enters the T4 state, make the READY signal high at the rising edge of the clock two states before the T4 state. Figure 6-2 shows an example of inserting one wait state.

The bus status is extended by the number of inserted wait states.

Figure 6-2. Ready Timing (with 1 wait state inserted)



Note Range of change.

6.1.2 Wait function by using WCU

The wait control unit (WCU) functions to insert a TW of 0 to 3 clocks in each bus cycle (memory cycle, external I/O cycle, DMA cycle, or refresh cycle). However, a wait cannot be inserted in the internal I/O cycle.

If a TW of 0 to 3 clocks is sufficient for an application system using the V40HL or V50HL, only the WCU is necessary, and the external READY control circuit is not necessary. In this case, the READY pin must be high.

(1) Wait in CPU cycle

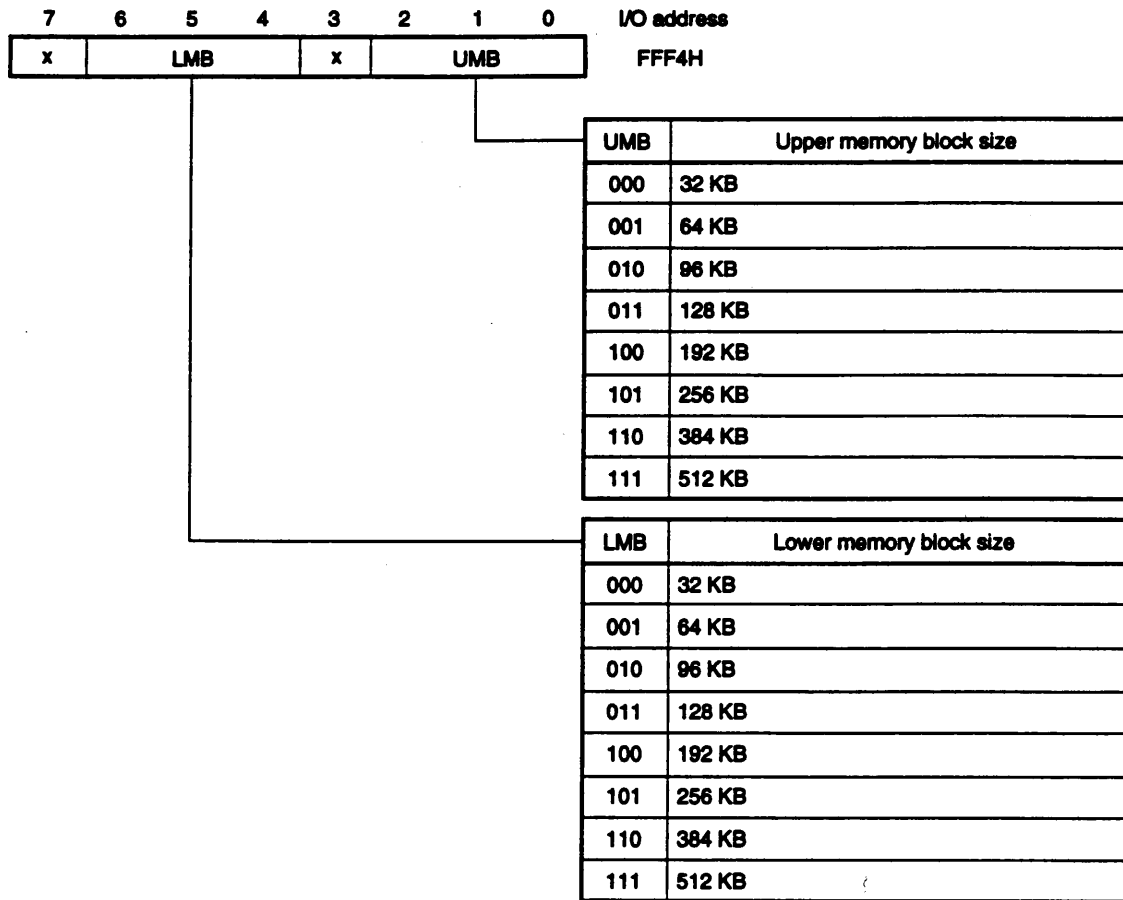
The WCU can insert a TW in the memory cycle and I/O cycle started by the CPU.

The V40HL and V50HL can access a memory space of 1 MB. Because the memory spaces of some systems do not require wait states, setting the same number of wait states for all memory spaces may cause degradation of the system performance. To prevent this, the WCU can divide the 1-MB memory space into three blocks and can independently set zero to three wait states for each of these blocks. In addition, the highest and lowest memory blocks can be further divided into two (subblocks) to which zero to three wait states can also be inserted. This means that the 1-MB memory space can be divided into up to five memory blocks and zero to three wait states can be inserted in each one (refer to Figure 6-6 Dividing Memory Space into Five Blocks).

The V40HL and V50HL can also access an I/O space of 64 KB, and the WCU can divide this I/O space into up to three areas.

(a) Programmable wait memory area setting register (WMB)

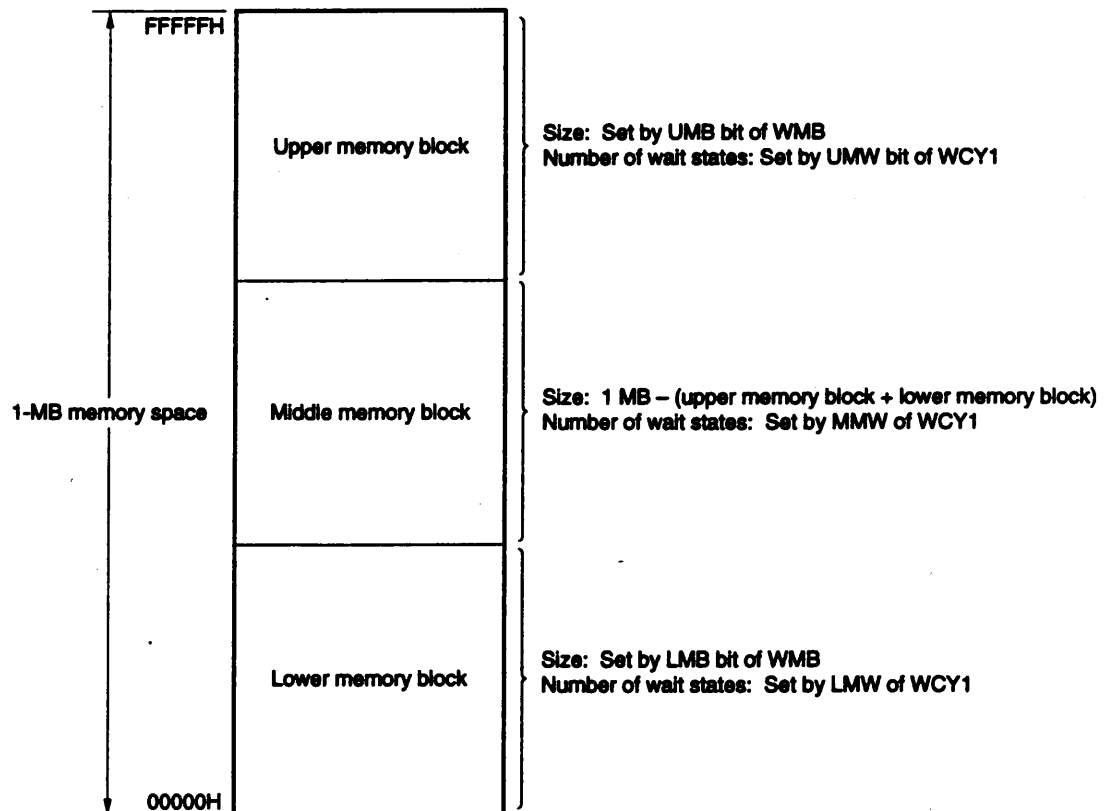
Figure 6-3. Programmable Wait Memory Area Setting Register (WMB)



Remark x: don't care

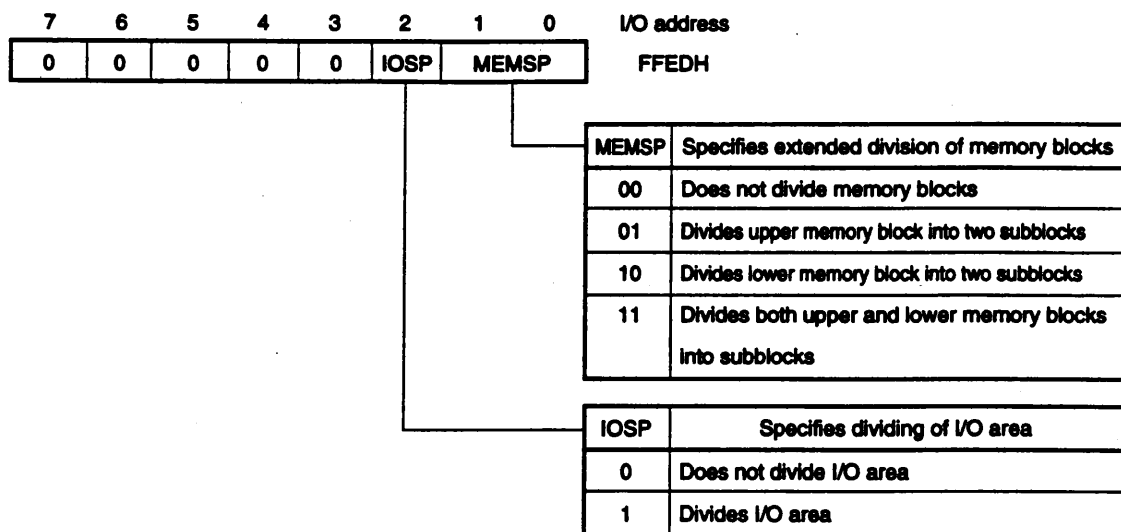
The programmable wait memory area setting register (WMB) divides the 1-MB memory space into three blocks: an upper, middle, and lower memory block. The UMB bit of this register sets the upper memory block, while the LMB bit sets the lower memory block. The rest of the memory space is the middle memory block.

Figure 6-4. Dividing Memory Space into Three Blocks



(b) Extended wait block select register (EXWB)

Figure 6-5. Extended Wait Block Select Register (EXWB)



Caution Be sure to clear bits 3 through 7 to 0.

(i) MEMSP bits

Of the three memory blocks into which the memory space is divided by using the WMB register, these bits further divide the upper and/or lower memory blocks into two subblocks.

(ii) IOSP bit

This bit divides the I/O area into three blocks. The size of each block is set by using the programmable wait I/O block setting register (WIOB).

Figure 6-6. Dividing Memory Space into Five Blocks

(Example of dividing upper and lower memory blocks into two subblocks)
(MEMSP bit = 11B)

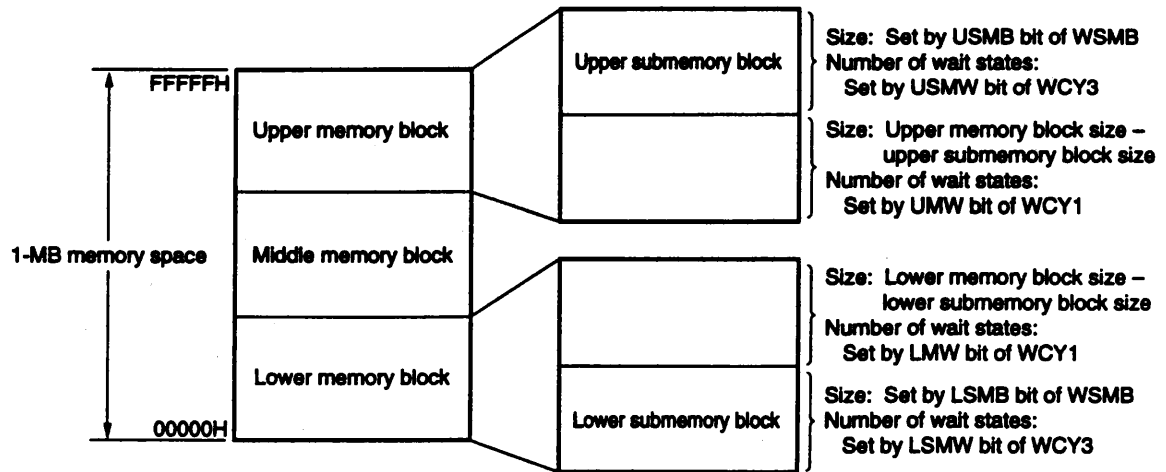
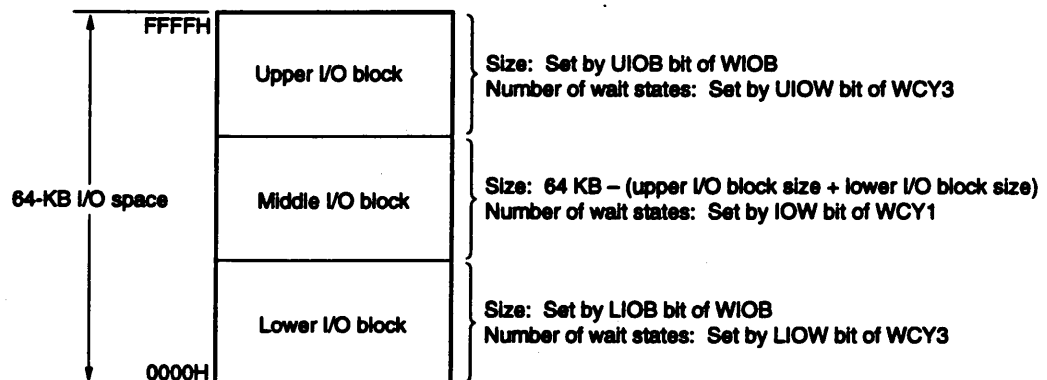
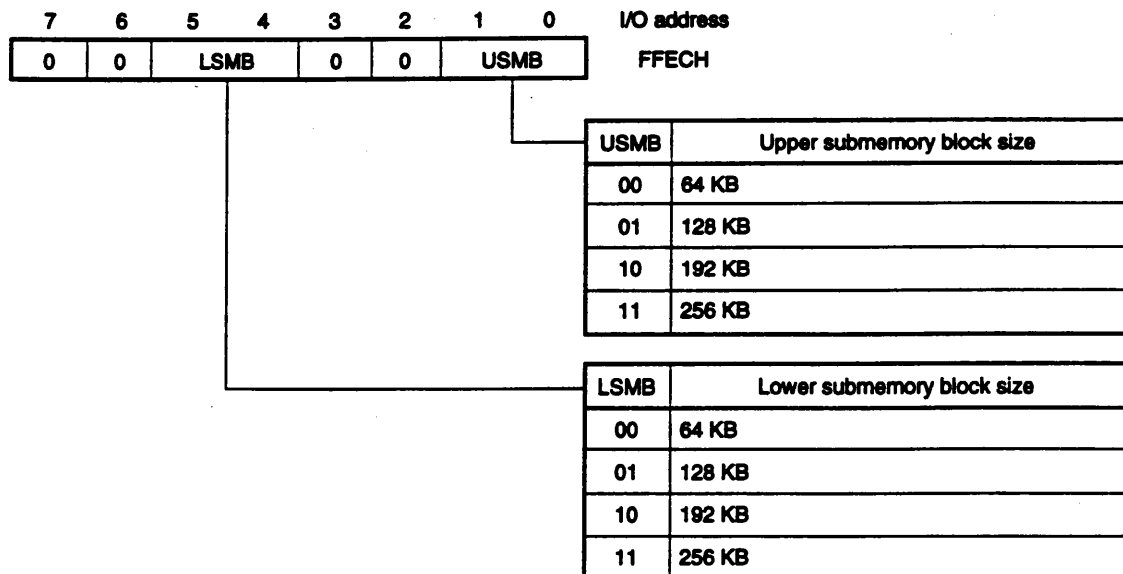


Figure 6-7. Dividing I/O Space into Three Blocks



(c) Wait submemory block setting register (WSMB)

Figure 6-8. Wait Submemory Block Setting Register (WSMB)



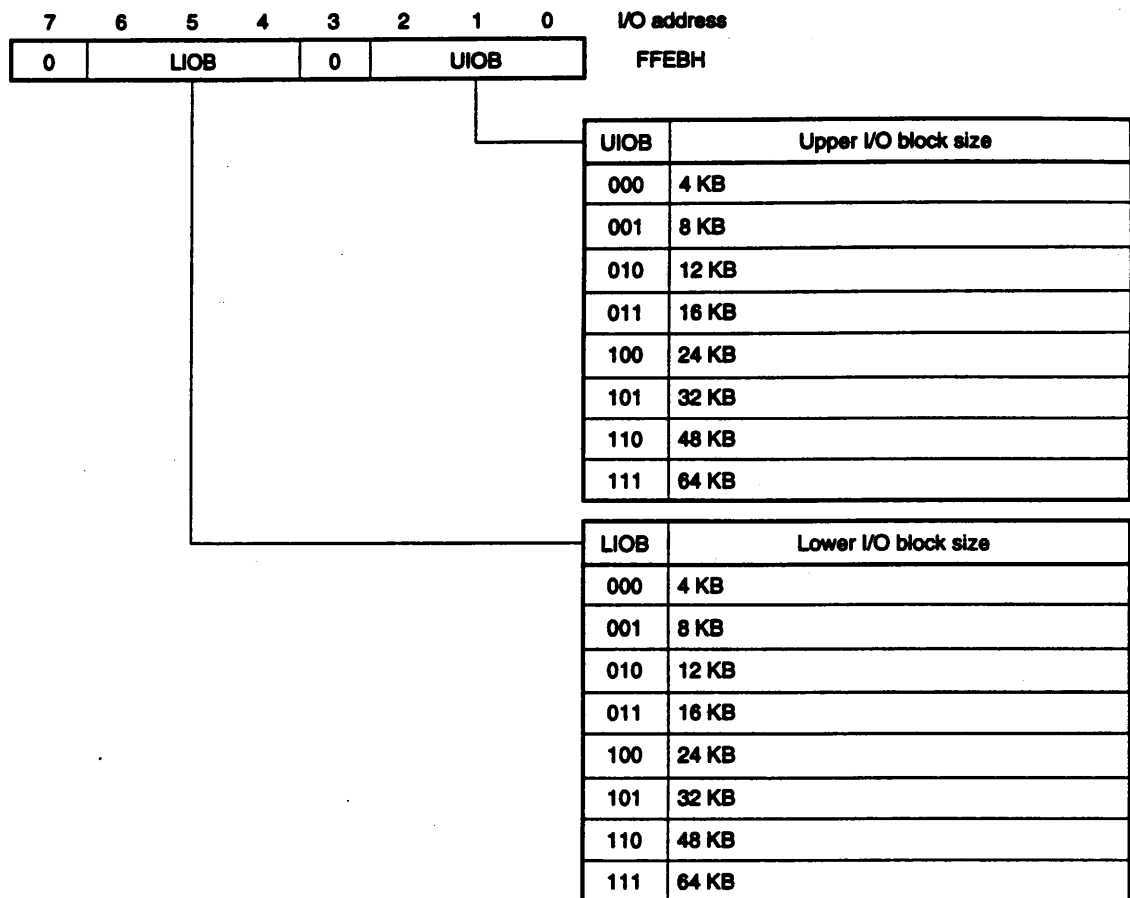
Cautions 1. Be sure to clear bits 2, 3, 6, and 7 to 0.

2. Keep the size specified by the LSMB bits and the size specified by USMB bits to within the size of the memory block specified by the UMB and LMB bits of WMB.

WSMB specifies the size of the upper and lower submemory blocks selected (divided) by the MEMSP bits of EXWB.

(d) Wait I/O block setting register (WIOB)

Figure 6-9. Wait I/O Block Setting Register (WIOB)

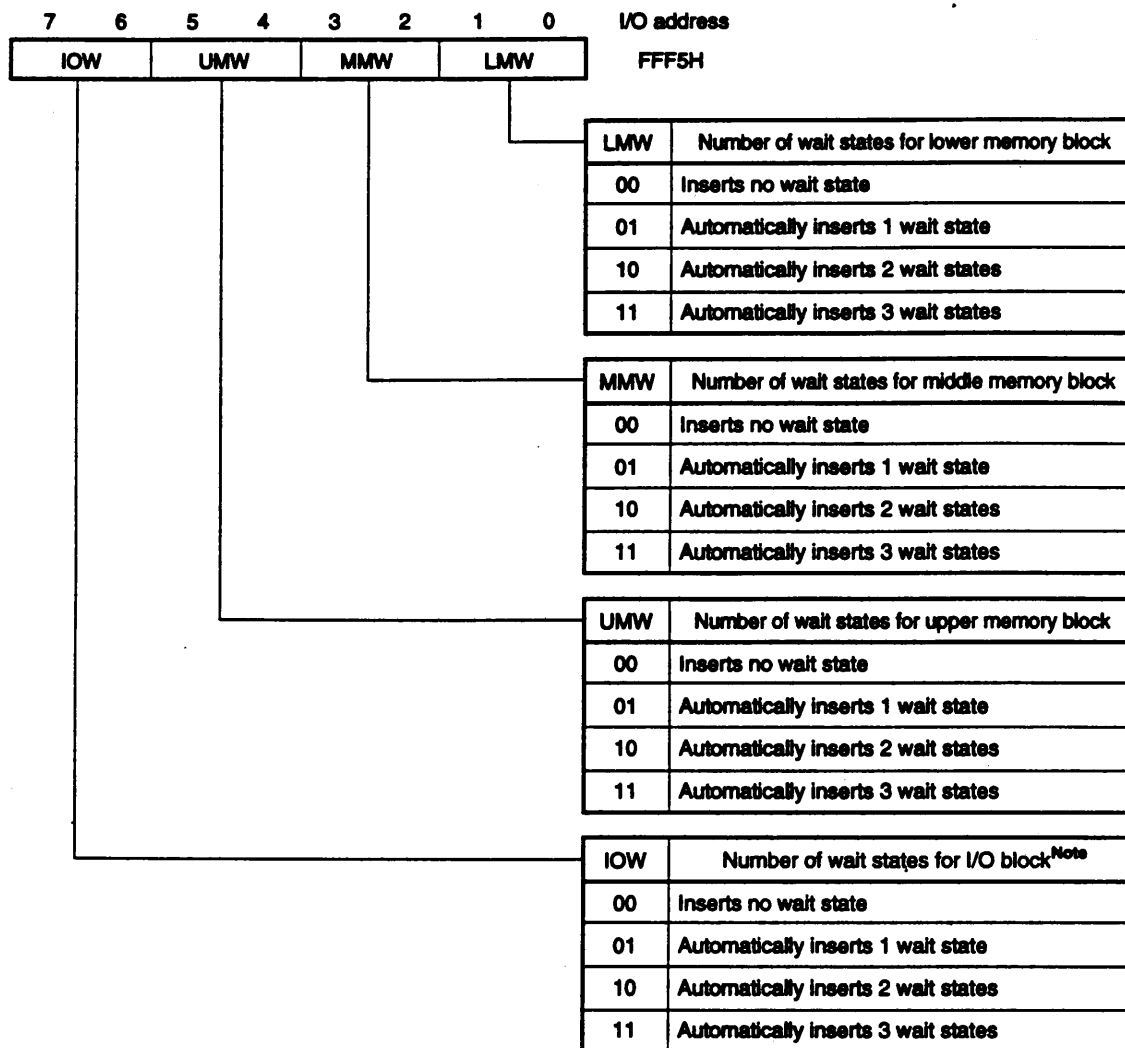


Caution Be sure to clear bits 3 and 7 to 0.

WIOB sets the size of the upper and lower I/O blocks of the three I/O blocks into which the I/O space is divided. The I/O block between the upper and lower blocks is the middle block (refer to Figure 6-7).

(e) Programmable wait cycle setting register 1 (WCY1)

Figure 6-10. Programmable Wait Cycle Setting Register 1 (WCY1)

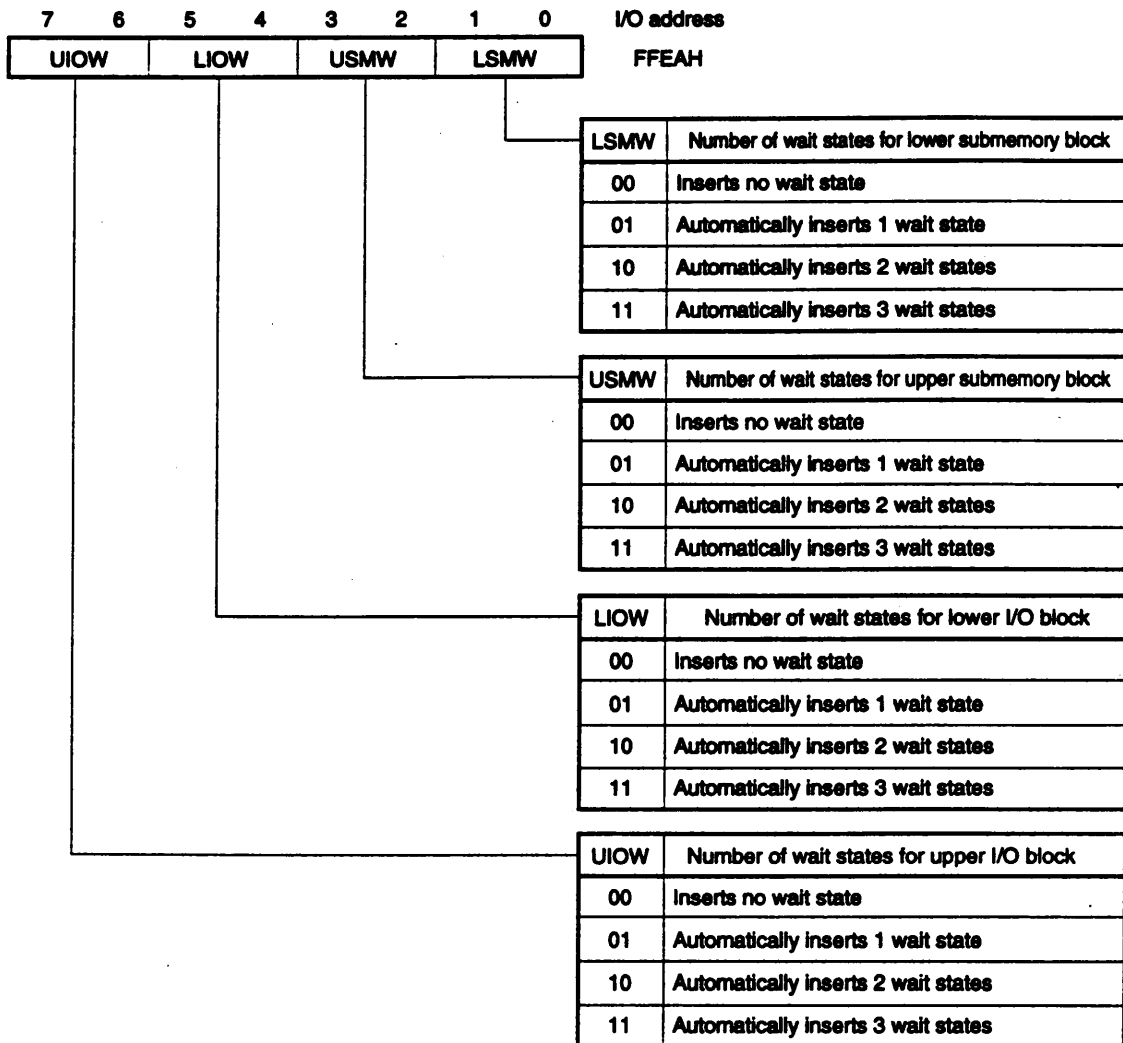


Note If dividing the I/O space is specified by the IOSP bit of EXWB, the number of wait states for the middle I/O block is specified.

WCY1 specifies the number of wait states for the three memory blocks into which the memory space is divided by using WMB, and for the middle I/O block of the I/O space that is divided into three blocks by using WIOB.

(f) Programmable wait cycle setting register 3 (WCY3)

Figure 6-11. Programmable Wait Cycle Setting Register 3 (WCY3)



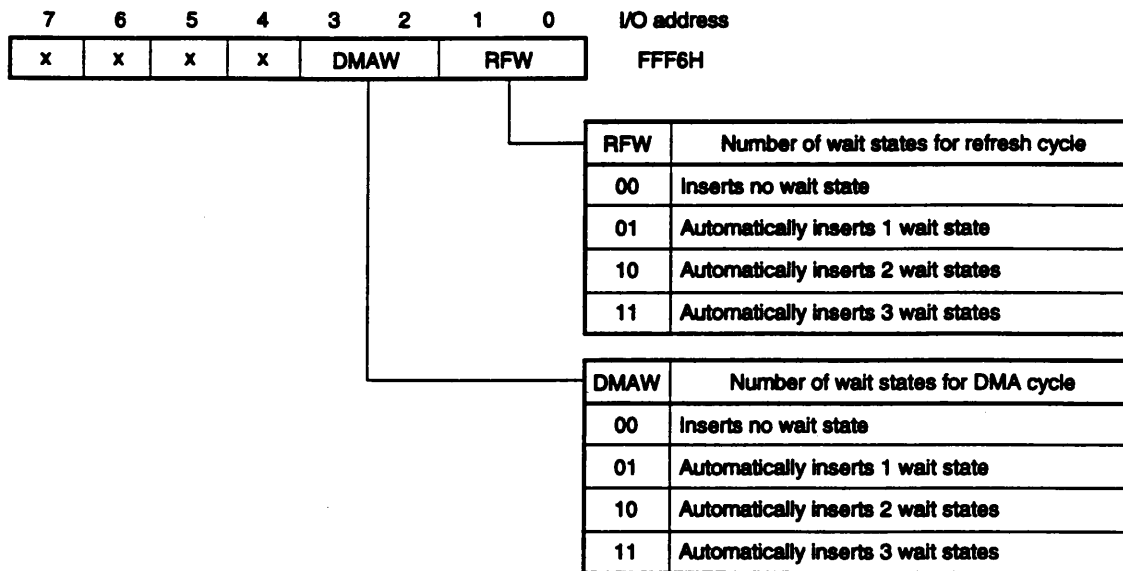
WCY3 has the following functions:

- (i) The UIOW and LIOW bits set the number of wait states for the upper and lower I/O blocks of the I/O space, respectively. (The middle I/O block is set by the IOW bit of WCY1.)
These bits are meaningful only when the dividing I/O space is specified by the IOSP bit of EXWB.
- (ii) The USMW and LSMW bits set the number of wait states for the upper and lower submemory blocks, respectively.
These bits are meaningful only when the upper or lower memory block or both are further divided into subblocks by using the MEMSP bit of EXWB.

(2) Wait states for refresh cycle and DMA cycle

The number of wait states to be inserted in the memory refresh cycle started by REFU and the DMA cycle started by DMAU is specified by using the programmable wait cycle setting register 2 (WCY2) in the system I/O area. In the refresh cycle and DMA cycle, the memory and I/O spaces cannot be divided.

Figure 6-12. Programmable Wait Cycle Setting Register 2 (WCY2)



Remark x: don't care

6.1.3 Relationship between WCU and READY pin

If wait cycles longer than 3 clocks are necessary, the WCU and READY pin can be used in combination. The number of wait cycles specified by the set value of the WCU or the number of wait cycles set by using the READY pin is inserted whichever greater. Figure 6-13 shows the relation between the WCU and READY pin, and Figure 6-14 shows an example of bus timing when four clocks of TWs are inserted.

Table 6-1 shows the validity of the WCU and READY pin in each bus cycle.

Figure 6-13. Relationship between WCU and READY Pin

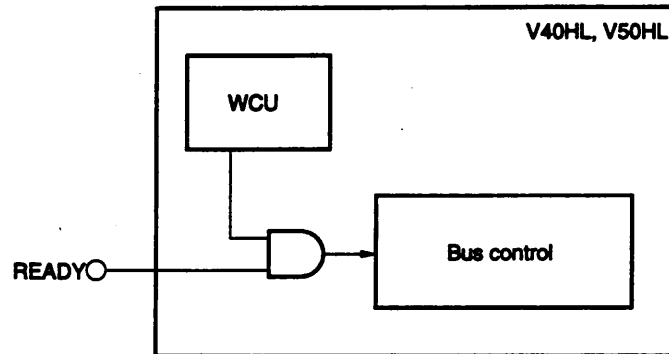
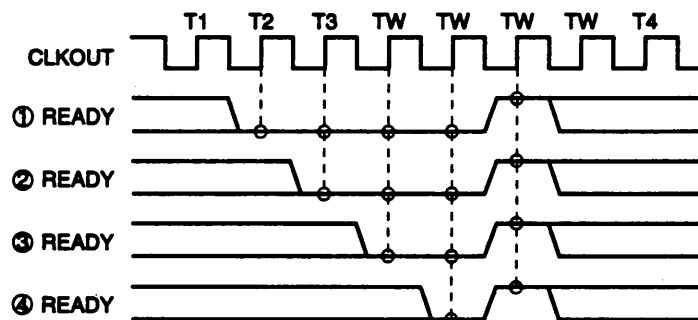


Figure 6-14. Example of Bus Timing When 4 Clocks of TWs Are Inserted



- ① When zero wait state is set by WCU
- ② When one wait state is set by WCU
- ③ When two wait states are set by WCU
- ④ When three wait states are set by WCU

Table 6-1. Validity of Wait State in Each Bus Cycle

Bus Cycle		Inserting Wait State by READY Signal	Inserting Wait State by WCU Setting
External I/O read cycle		○	○
External I/O write cycle			
Interrupt acknowledge cycle	Single mode	x	x
	Expanded mode	○	○ Note 1
CPU memory read cycle		○	○
CPU memory write cycle			(Area can be specified.)
DMA cycle (DMAU)			○
DMA cycle (cascade mode)		x Note 2	x Note 2
Refresh cycle		○	○
Internal I/O read cycle		x	x
Internal I/O write cycle			

○: valid, x: invalid

- ★ Notes 1. When ICU is used in expanded mode, set IOSB bit of EXWB to 0 (The I/O area cannot be divided for wait control).
2. Controlled by DMA controllers cascade-connected.

6.2 Refresh Function

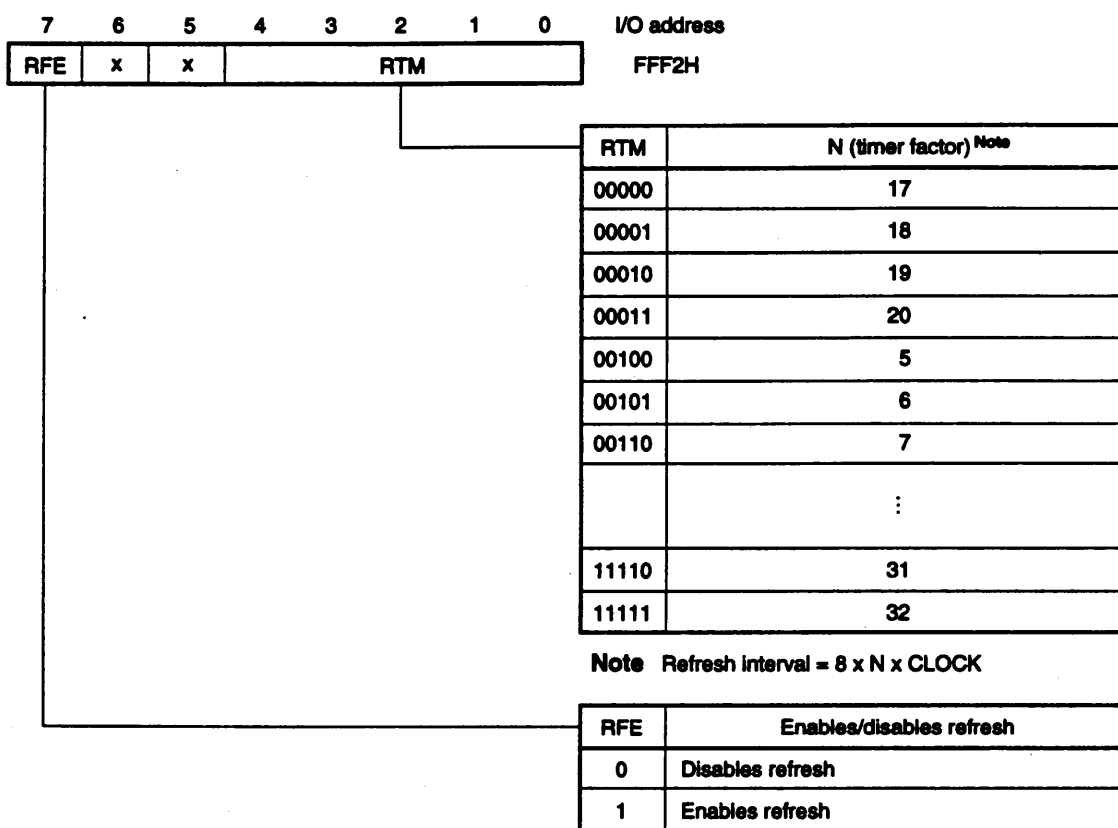
The V40HL and V50HL have a refresh function that is used when DRAM is used for the memory space. The refresh function is implemented by the refresh unit (REFU). REFU outputs a 16-bit refresh address (A0 through A15) and refresh request signals (such as $\overline{\text{REFRQ}}$) to refresh the DRAM.

6.2.1 Refresh address

As the refresh address, a row address of A0 through A15 is output. Each time one refresh cycle ends, the V40HL increments the address by 1 and the V50HL increments the address by two to update the address. This refresh address is not affected by reset when refreshing is enabled (when the RFE bit of the refresh control register (RFC) is set to 1).

6.2.2 Refresh control register (RFC)

Figure 6-15. Refresh Control Register (RFC)



Remark x: don't care

The RFE bit enables or disables the refresh function of the REFU.

The five RTM bits set the refresh interval. If the refresh interval is too short, the throughput of the CPU drops. If RTM is set to 00000 to 00011, the timer factor N is 17 to 20, instead of 1 to 4. The actual refresh interval is determined by $8 \times N \times \text{CLOCK}$. For example, where the operating clock is 8 MHz and $N = 10$, the refresh interval is:

$$8 \times 10 \times \frac{1}{8,000,000} \text{ (sec)} = 10 \text{ (}\mu\text{sec)}$$

The RFE and RTM bits are set or reset as follows by the reset input:

- RFE bit $\left\{ \begin{array}{ll} \text{When power turned on} & \dots \text{ Undefined} \\ \text{By reset input during operation} & \dots \text{ Not affected} \end{array} \right.$
- RTM: initialized to 01000 ($N = 9$)

Cautions 1. The RFE bit is undefined when there is a power-on reset (refreshing may be enabled or disabled).

To disable refreshing without executing the refresh cycle even only once, clear the RFE bit to 0 within 72 clocks after RESOUT has become inactive.

2. If the refresh request is kept pending in the refresh queue, the refresh cycle will execute even if the RFE bit is cleared to 0 (disabling refreshing) until the contents of the refresh queue become 0. The refresh queue is cleared at reset.

6.2.3 REFU bus control

Normally, the REFU has the lowest priority in using the bus. Therefore, even if the REFU periodically issues a refresh request, the refresh cycle is not executed unless the bus is available. In this case, the refresh request for the refresh cycle not executed is stored in the refresh queue. The refresh request in this queue is executed when the bus is available, and the contents of the refresh queue are decremented.

If the bus is available only for a short time and if the contents of the refresh queue reach the maximum value of 7, the REFU is given the highest priority in using the bus, and the refresh cycle will execute continuously after execution of the current bus cycle has completed. This will continue until the contents of the refresh queue reaches 3. After that, the REFU is given the lowest priority again.

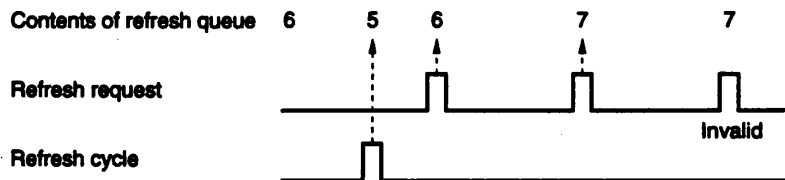
A refresh request that is issued when the content of the refresh queue is 7 is ignored.

While the CPU executes an instruction with BUSLOCK prefix, all bus masters other than the CPU, and including the REFU, cannot use the bus. Therefore, care must be exercised when appending the BUSLOCK prefix to an instruction that takes a long time to execute.

Table 6-2. REFU Bus Control

Priority	Refresh Operation
Lowest	Executed by detecting release of bus in advance
Highest	Immediately starts execution after current bus cycle ends, and execution continues until contents of refresh queue reach 3

Figure 6-16. Invalid Refresh Request

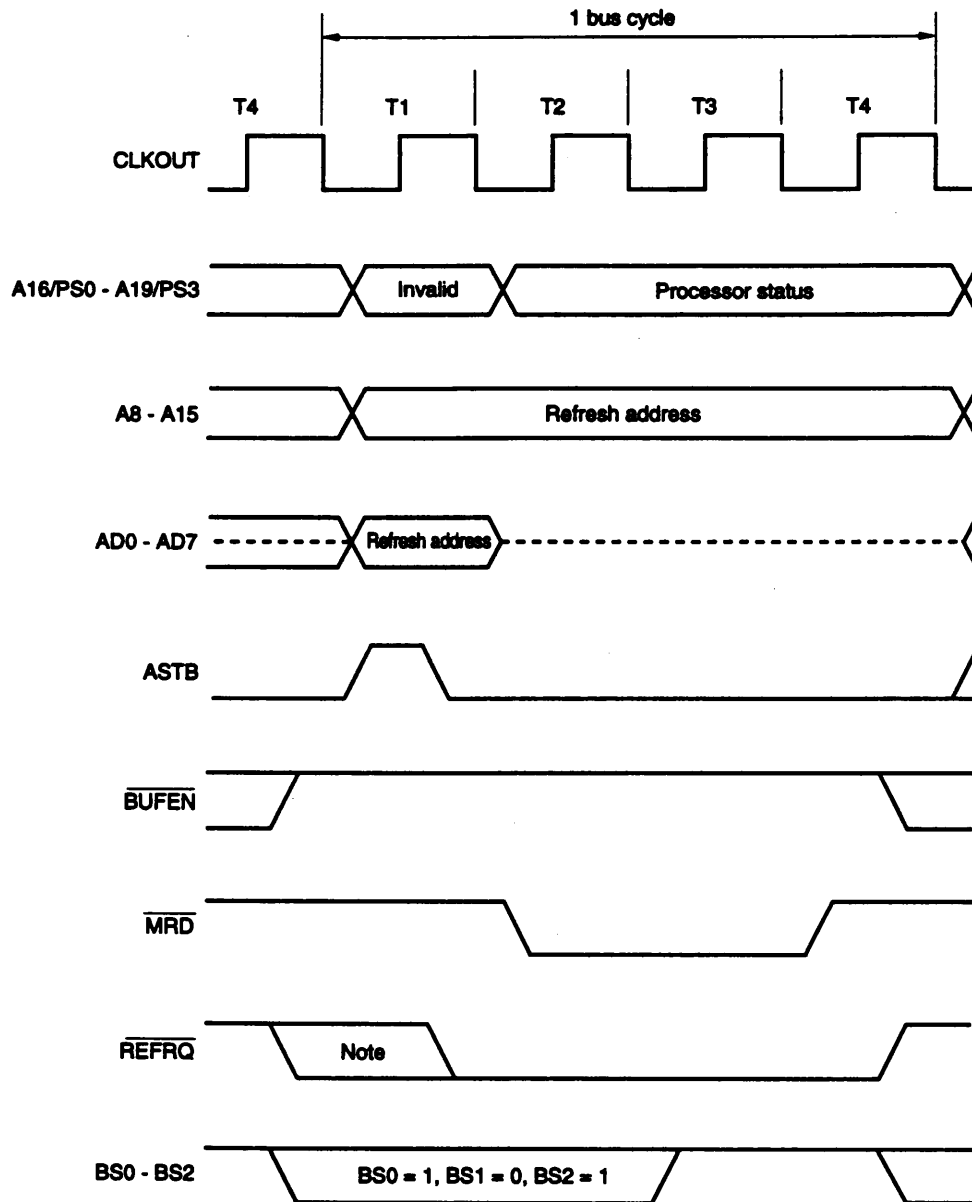


6.2.4 Refresh timing

The V40HL and V50HL can change the output timing of the $\overline{\text{REFRQ}}$ signal. $\overline{\text{REFRQ}}$ is output in synchronization with the rising of the T1 state of the refresh cycle when the EREF bit of SCTL is 0. When the EREF bit is 1, $\overline{\text{REFRQ}}$ is output in synchronization with the rising of the T4 state immediately before the refresh cycle (for details on setting, refer to Figure 4-8 System Control Register (SCTL)). Figure 6-17 shows the refresh timing.

Figure 6-17. Refresh Timing (1/2)

(a) V40HL

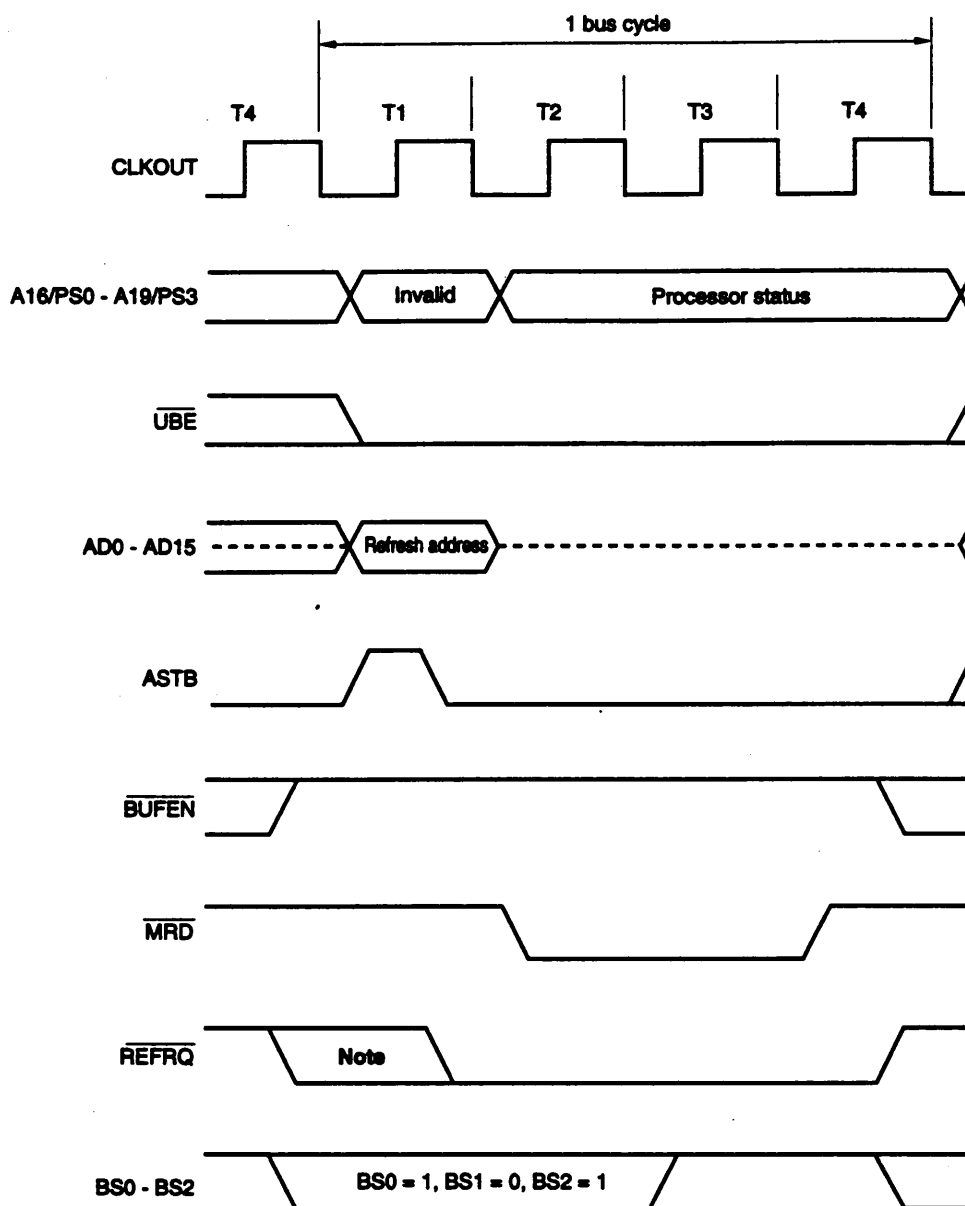


Note A low level is output when the EREF bit of SCTL is 1.

Remark The broken lines indicate the high-impedance state.

Figure 6-17. Refresh Timing (2/2)

(b) V50HL



Note A low level is output when the EREF bit of SCTL is 1.

Remark The broken lines indicate the high-impedance state.

6.3 Interrupt Function

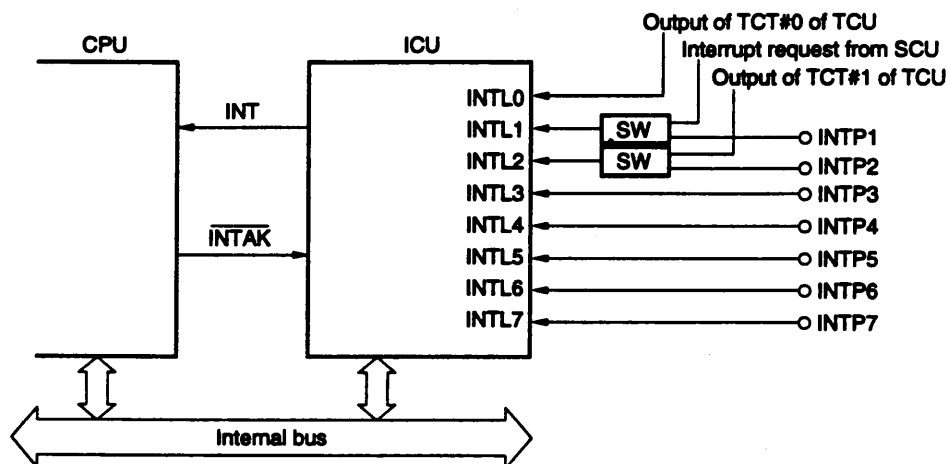
The interrupts of the V40HL and V50HL can be broadly classified into those triggered by hardware interrupt requests (NMI input, INTP1 through INTP7 inputs, and requests from internal peripheral units) and those triggered by software. All of these interrupts are vectored interrupts. They are selected by specifying automatically (fixed vector) or each time (variable vector) one location of an interrupt vector table prepared in advance, to determine the start address of the corresponding interrupt routine.

★ (1) Hardware Interrupt

(a) Maskable interrupt (INTL0-INTL7 input)

Each interrupt request is arbitrated by the ICU and issued to the CPU. Figure 6-18 shows the interrupt requests to the ICU.

Figure 6-18. ICU Inputs



INTL3 through INTL7 are input from external pins (INTP3 through INTP7). INTL1 and INTL2 can be input from external pins (INTP1 and INTP2) or TCU and SCU by software. INTL0 is fixed to input from TCU. For details, refer to 5.9 Interrupt Control Unit (ICU).

CPU operation when the ICU generates a maskable interrupt request differs as follows according to the status of the IE flag of PSW.

(I) IE flag = 0

The CPU ignores interrupt requests and continues executing instructions in sequence.

(II) IE flag = 1

Following completion of the instruction currently being executed, the CPU accepts the interrupt request and begins interrupt processing.

Upon receiving an interrupt request, the CPU issues the interrupt acknowledge cycle to supply an interrupt vector from the ICU or external interrupt controller.

Interrupt requests are not latched inside the CPU, therefore the CPU needs to hold an interrupt request it accepts until it generates the first interrupt acknowledge cycle.

(b) Non-maskable Interrupt (NMI Input)

NMI input is used to notify the microprocessor of catastrophic events such as sudden fluctuations in the power supply (sudden failure), memory errors, bus errors, etc. Interrupts requested by NMI input are called non-maskable interrupts because they cannot be masked by software.

No interrupt vector needs to be supplied from outside for non-maskable interrupts because they are fixed to vector 2 of the interrupt vector table. Therefore, no interrupt acknowledge cycle is issued when a non-maskable interrupt request is accepted.

(2) Software Interrupt

*

Software interrupts are issued in synchronization with instruction execution in response to causes inside the CPU. Like non-maskable interrupts, the acceptance of interrupt requests cannot be disabled with the IE flag of PSW. Moreover, no acknowledge cycle is issued.

Software interrupts are divided into the following four types.

- Interrupts issued by interrupt instruction (BRK and BRKV)
- Interrupts issued upon error detection during execution of DIV, DIVU or CHKIND instruction
- Interrupts issued by specifying BRK flag of PSW (Single-step interrupts)
- Interrupts issued as the result of the execution of emulation mode instructions (BRKEM and CALLEM)

The characteristics of software interrupts are summarized below.

- Interrupt vectors are either included in instruction codes (BRK instruction) or determined beforehand.
- No interrupt acknowledge cycle is issued even when an interrupt request is accepted.
- Interrupt request acceptance cannot be masked except for break interrupts.

Table 6-3. Interrupt Sources

Interrupt Source		Clocks ^{Note}	Vector Number	Priority
Hardware	NMI (rising-edge active)	58/38	2	2
	ICU	68/49	32 to 255	3
Software	Divide error of DIVU	65/45	0	1
	Divide error of DIV	65 to 75/45 to 55		
	CHKIND boundary over	72 to 75/52 to 55	5	
	BRKV	52/40	4	
	BRK3	50/38	3	
	BRK imm8		32 to 255	
	BRKEM imm8			
	CALLN imm8	58/38		
	BRK flag (single step)		1	4

★ **Note** Minimum number of clocks required from CPU acceptance of interrupt request up to branching to the interrupt processing routine.

Remark The number of clocks shown at the left of the slash mark apply to the V40HL, and those shown at the right apply to the V50HL.

Figure 6-19 shows the interrupt vector table. This table is mapped to a 1-KB memory area of addresses 000H through 3FFH and can have 256 vectors (where 1 vector consists of 4 bytes).

Figure 6-19. Interrupt Vector Table

3FFH		Vector 255	For general-purpose use • BRK imm8 instruction • BRKEM instruction • ICU interrupt • CALLN instruction	
3FCH				
3FBH				
080H	32			
07FH				
07CH	31			
07BH				
			Reserved	
018H	6			
017H			CHKIND instruction	
	5			
014H			BRKV instruction	
013H	4			
010H			BRK 3 instruction	
00FH	3			
00CH			NMI input	
00BH	2			
008H			Break flag	
007H	1			
004H			Divide error	
003H	0			
000H				

Because the sources that use vectors 0 through 5 are specified, and vectors 6 through 31 are reserved, these vectors are not for general purpose use.

Vectors 32 through 255 are for general purpose use and can be used for a 2-byte break instruction, BRKEM instruction, ICU interrupt input, and CALLN instruction (during emulation).

One interrupt vector consists of 4 bytes. The 2 bytes of the lower address are loaded as an offset to the PC, and the 2 bytes of the higher addresses are loaded as a base to the PS.

Example Vector 0

000H	001H
002H	003H

PS ← (003H, 002H)

PC ← (001H, 000H)

Based on this format, the programmer initializes the contents of each vector at the beginning of the program. The basic step to jump to an interrupt processing routine is as follows:

TA ← vector, low (offset)
 TC ← vector, high (segment base)
 SP ← SP - 2, (SP + 1, SP) ← PSW
 IE ← 0, BRK ← 0, MD ← 1
 SP ← SP - 2, (SP + 1, SP) ← PS
 PS ← TC
 SP ← SP - 2, (SP + 1, SP) ← PC
 PC ← TA

Because the IE and BRK flags of the PSW are reset, the maskable interrupts (ICU) and single-step interrupt are disabled when the interrupt routine is started.

6.3.1 Operation of ICU Interrupt

If an interrupt request from the ICU is detected in the interrupt enable status ($IE = 1$) at the end of execution of a certain instruction, and if the NMI or hold request is not active at the same time, the ICU interrupt request is acknowledged, and the interrupt acknowledge cycle is started (refer to Figures 6-20 and 6-21).

If the interrupt request is issued only from the ICU, the CPU receives a vector number via the internal bus of the V40HL or V50HL. If the μ PD71059 is cascade-connected to the ICU as a slave interrupt controller, and if the slave issues an interrupt request, the CPU receives a vector number via the external data bus. Consequently, the operation to be performed in the interrupt acknowledge cycle when an interrupt request is issued from a master (ICU) is slightly different from the operation to be performed when an interrupt request is issued from a slave.

The interrupt acknowledge cycle consists of two bus cycles. In the first cycle, the \overline{INTAK} , \overline{ASTB} , and address signals are output. The output address is a meaningless value and no data can be read from or written to this address. At this time, the refresh request, DMA request, and hold request are not accepted, and the $\overline{BUSLOCK}$ signal is output to disable the other devices from using the bus.

The operation of the second acknowledge cycle differs depending on whether the interrupt request has been issued from the slave or master. If the interrupt request is issued from the master, the \overline{INTAK} , \overline{ASTB} , and (meaningless) address signals are output, and the CPU receives an interrupt vector number output by the ICU to the internal data bus. If the interrupt request is issued from the slave, a slave address for the interrupt controller of the slave is output to address bits A8 through A10 at the same time as \overline{ASTB} (any other address is meaningless). When \overline{INTAK} is subsequently output, the interrupt controller specified by the slave address outputs an interrupt vector number to the data bus (AD0 through AD7), and the CPU receives this vector number.

In this way, after the second acknowledge cycle has ended, the interrupt vector table corresponding to the interrupt vector number the CPU has received is accessed, the contents of the PSW, PS, and PC are saved, an interrupt start address based on the contents of the interrupt vector table is output, and the interrupt processing routine is started.

The first acknowledge cycle is necessary for only synchronizing the ICU or slave interrupt controller.

The interrupt acknowledge cycles of the V40HL and V50HL differ from each other as follows:

(1) Interrupt acknowledge cycle of V40HL

The first acknowledge cycle and the second acknowledge cycle always occur in succession.

If the T1 state continues for the duration of five clocks after the second acknowledge cycle, the instruction fetch cycle (which is actually invalid because execution shifts to the interrupt processing routine) may follow.

If the instruction fetch cycle is completed in four clocks, one clock of T1 state is inserted. This means that at least five clocks are inserted in between the second acknowledge cycle and vector table read cycle.

(2) Interrupt acknowledge cycle of V50HL

Three clocks of the T1 state may be inserted or an instruction fetch cycle (four clocks min.) may be inserted in between the first and second acknowledge cycles.

Therefore, five clocks of T1 state are always inserted in between the second acknowledge cycle and vector table read cycle.

Both the V40HL and V50HL reads an 8-bit vector number from the ICU or slave interrupt controller in the second acknowledge cycle. However, the number of subsequent cycles necessary for reading the interrupt vector table and saving the contents of PSW, PS, and PC differs between the V40HL and V50HL. This is because of the difference in the data bus width. The V40HL reads an offset word and a segment word, and saves the contents of the PSW, PS, and PC in two bus cycles each, whereas the V50HL executes these operations in 1 bus cycle each.

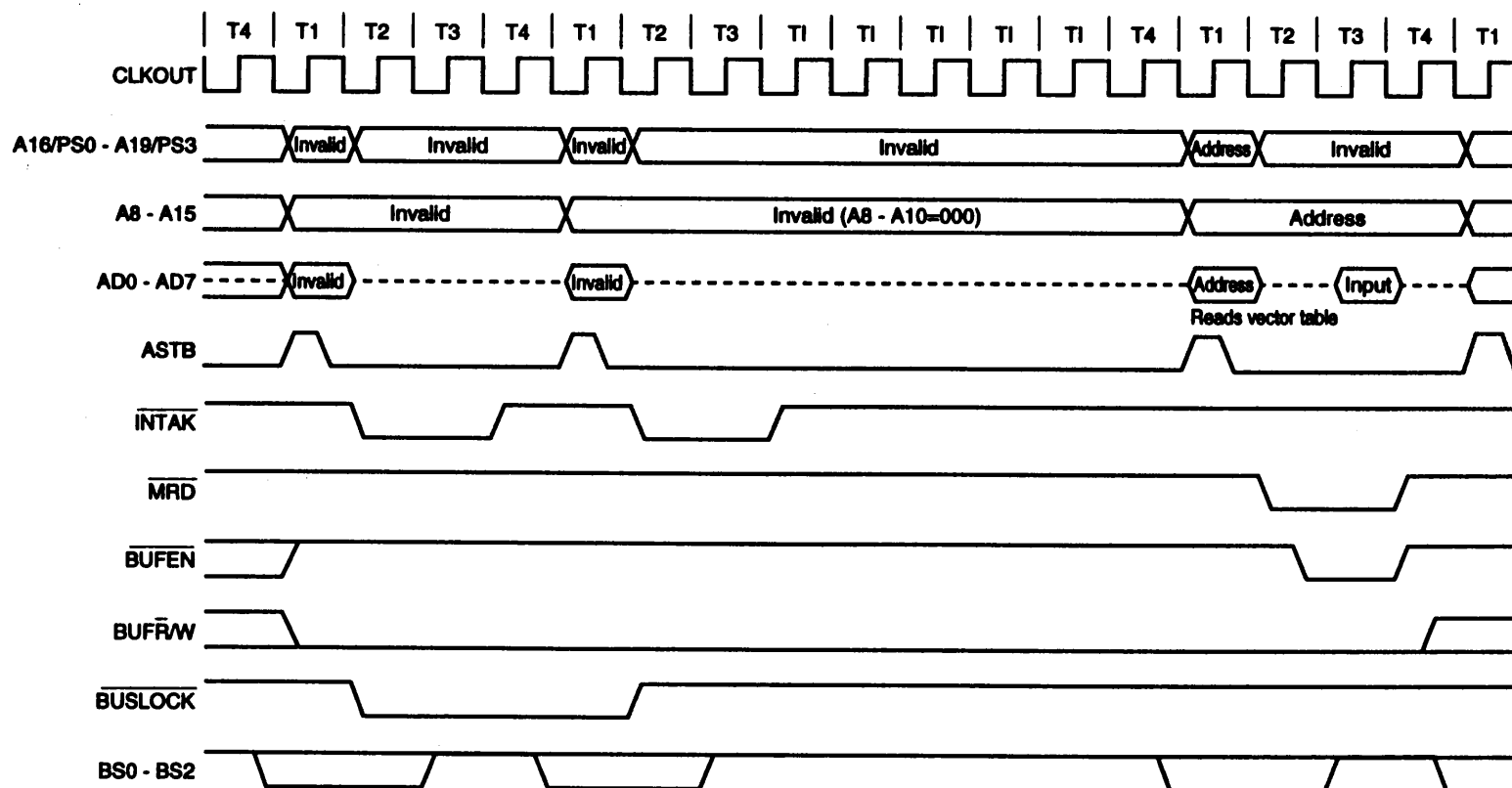
The $\overline{\text{UBE}}$ signal of the V50HL is low in the first and second acknowledge cycles and while the offset word and segment word are subsequently accessed.

- Interrupt acknowledge operation of V40HL
 1. Acknowledge cycle (first)
 2. Acknowledge cycle (second)
 3. Reading of lower byte of offset word
 4. Reading of higher byte of offset word
 5. Reading of lower byte of segment word
 6. Reading of higher byte of segment word
 7. Saving of lower byte of PSW
 8. Saving of higher byte of PSW
 9. Saving of lower byte of PS
 10. Saving of higher byte of PS
 11. Saving of lower byte of PC
 12. Saving of higher byte of PC
 13. Output of interrupt start address

- Interrupt acknowledge operation of V50HL
 1. Acknowledge cycle (first)
 2. Acknowledge cycle (second)
 3. Reading of offset word
 4. Reading of segment word
 5. Saving of PSW
 6. Saving of PS
 7. Saving of PC
 8. Output of interrupt start address

Figure 6-20. Example of Interrupt Acknowledge Timing (single mode) (1/2)

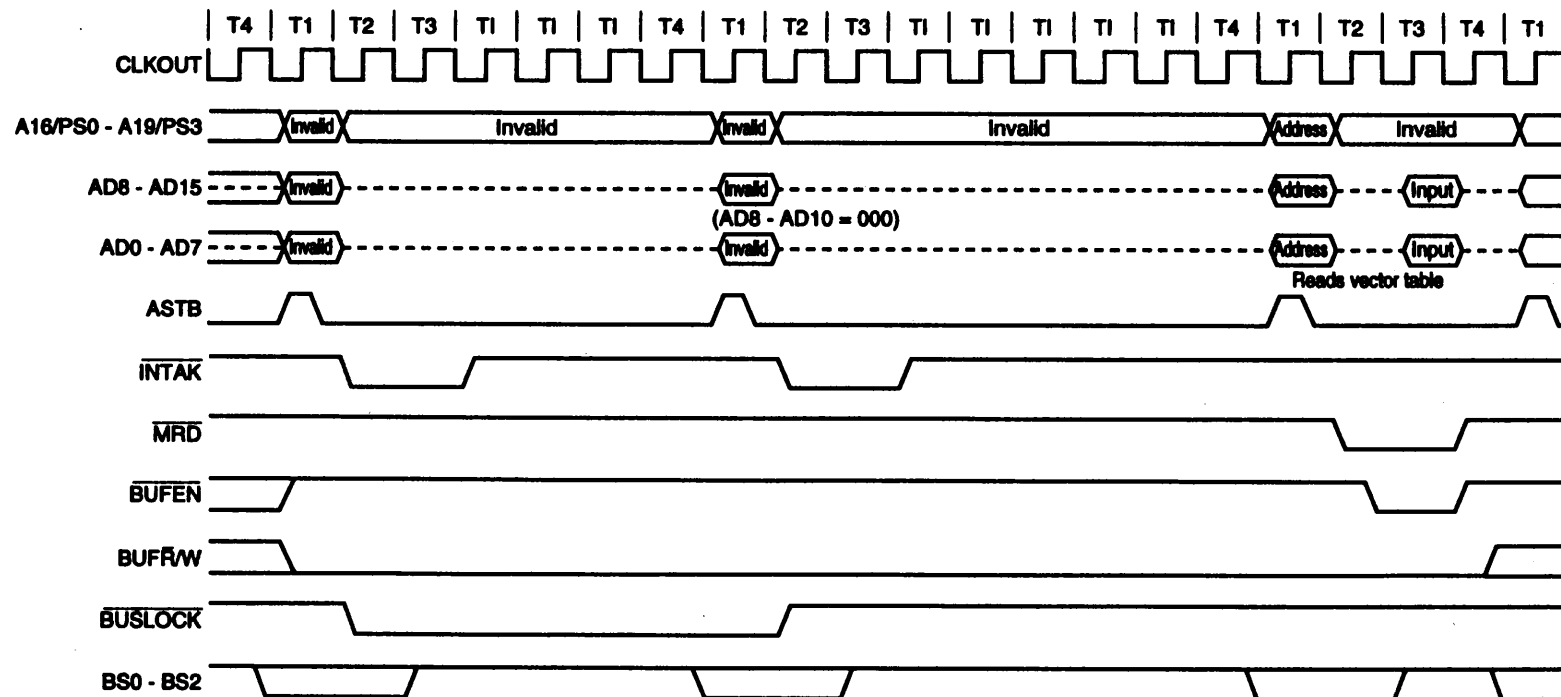
(a) V40HL



Remark The broken lines indicate the high-impedance state.

Figure 6-20. Example of Interrupt Acknowledge Timing (single mode) (2/2)

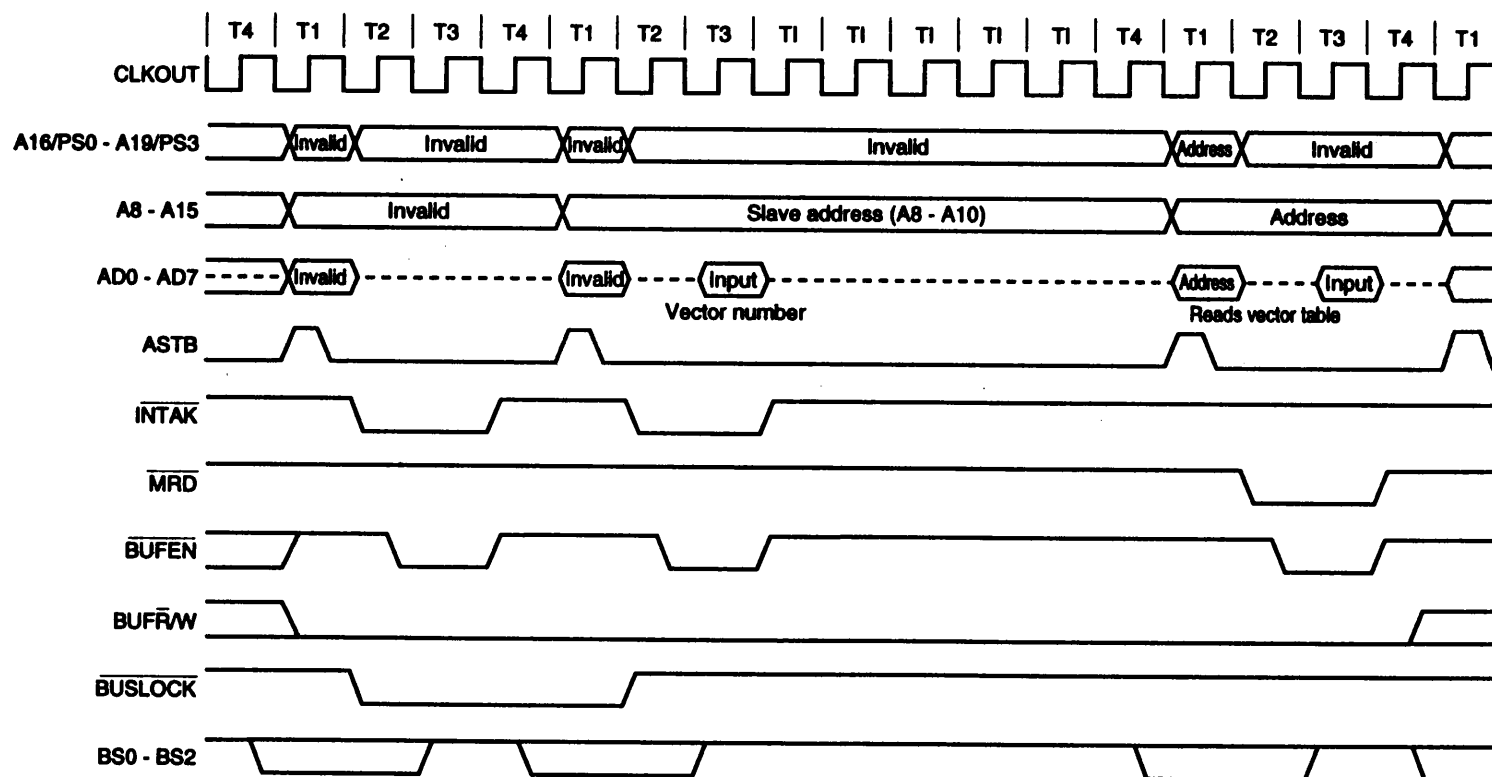
(b) V50HL



Remark The broken lines indicate the high-impedance state.

Figure 6-21. Example of Interrupt Acknowledge Timing (expanded mode) (1/2)

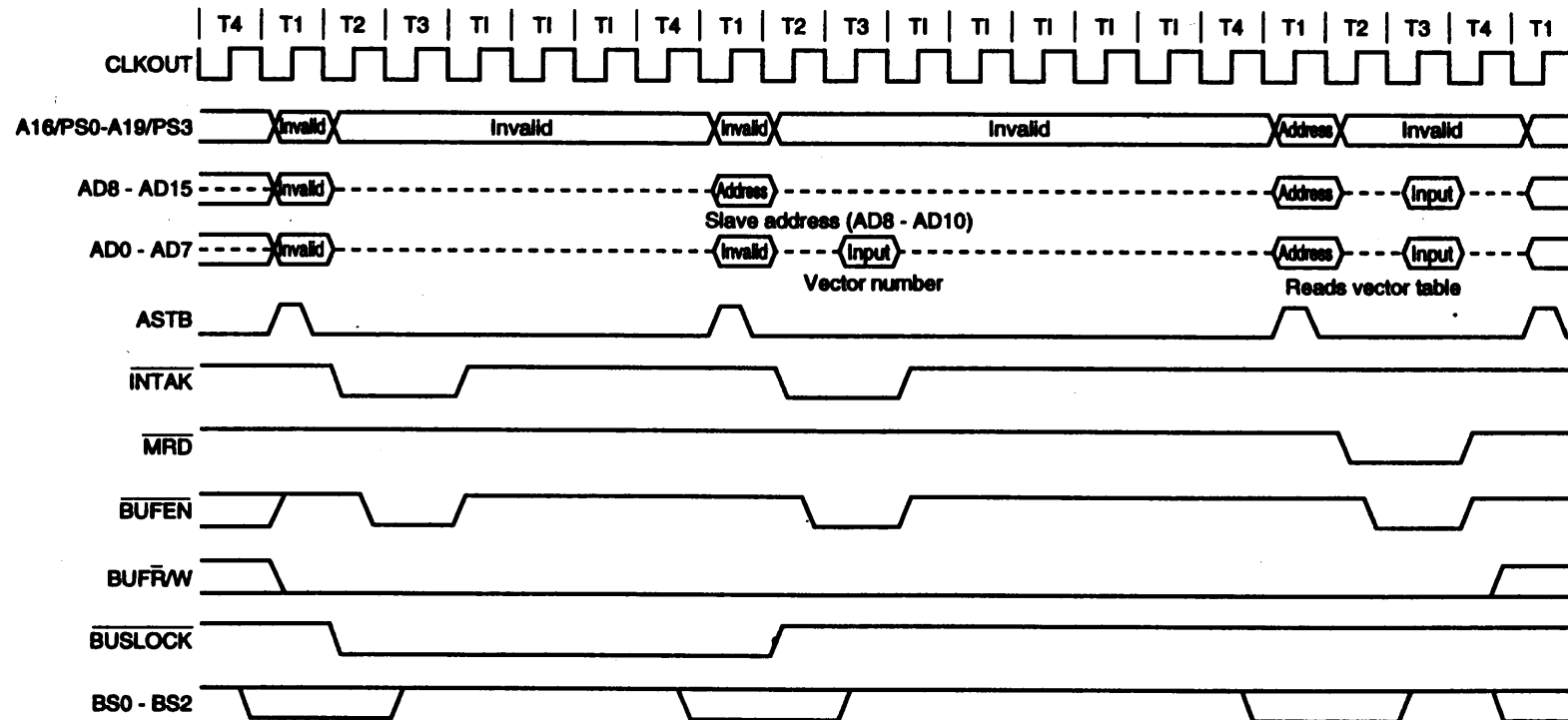
(a) V40HL



Remark The broken lines indicate the high-impedance state.

Figure 6-21. Example of Interrupt Acknowledge Timing (expanded mode) (2/2)

(b) V50HL



Remark The broken lines indicate the high-impedance state.

6.3.2 BRK flag (single-step) interrupt

The V40HL and V50HL have a single-step interrupt function useful for debugging the program. This interrupt is controlled by the BRK flag (bit 8 of the PSW). However, there is no instruction available that directly sets or resets the BRK flag. To manipulate the BRK flag, save the PSW to the stack, manipulate the flag, and then restore the PSW. In this way, the BRK flag is indirectly set or reset.

When the BRK flag has been set, the next one instruction is executed, and then the interrupt routine (monitor program, etc.) specified by vector 1 is started. At this time, the BRK flag and IE flag are reset.

When vector 1 interrupt has started once, the instructions of the interrupt routine are executed not on a one-by-one basis, but continuously in the same manner as the other interrupts. While the instructions are executed, the statuses of the internal registers and flags, and memory contents can be checked or dumped.

In this interrupt routine, the number of times the single-step operation is to be executed is checked. If it is all right to end the single-step operation, the BRK flag in the stack is reset by a memory manipulation instruction and execution returns to the main routine. Consequently, instructions can be executed successively after execution has returned to the main routine.

If execution returns without the BRK flag operation, BRK = 1 saved to the stack is restored to PSW. In this case, vector 1 interrupt occurs again after one instruction in the main routine has been executed.

6.3.3 Timing when interrupts not acknowledged

The NMI input, ICU interrupt request, and single-step interrupt request are not acknowledged at the following timing (1) through (4), that is, in between an instruction that directly sets data to the segment register or three types of prefixes and the next one instruction. Only the ICU interrupt is not acknowledged in between the EI instruction in (5) and the next one instruction.

- (1) Between MOV sreg, reg16; MOV sreg, mem16; MOV reg16, sreg; MOV mem16, sreg; or POPsreg instruction and next one instruction
- (2) Between segment override prefix (PS:, SS:, DS0:, or DS1:) and next one instruction
- (3) Between repeat prefix (REPC, REPNC, REP, REPE, REPZ, REPNE, or REPNZ) and next one instruction
- (4) Between bus lock prefix (BUSLOCK) and next one instruction
- (5) Between EI instruction and next one instruction (Only ICU interrupt is not acknowledged.)

However, the NMI request signal generated in the interrupt disable timing (1) to (4) above is kept pending internally, and is acknowledged after the next one instruction has been executed.

6.3.4 Interrupt processing during execution of block processing instruction

If an external interrupt (NMI or ICU interrupt in the EI status) occurs in the middle of execution of the primitive block transfer/comparison, or primitive I/O instruction, the CPU acknowledges the interrupt, and branches to the corresponding interrupt address. At the beginning of the interrupt processing routine started in this way, CW which serves as a block data counter, is saved to the stack. If CW is restored at the end of the interrupt processing routine and execution is returned to the original routine by using the RETI instruction, the block processing that has been interrupted can be resumed.

Up to three types of prefixes that are placed at addresses before that of the block processing instruction can be recorded. When returning from the interrupt processing routine, the return addresses are modified (one address per type of prefix) and saved so that a return to the addresses with the prefixes is possible.

To use these functions effectively, it is necessary to keep the total number of prefixes placed before a block processing instruction to within 3.

Example 1. Good example

In this example, all BUSLOCK, REPC, and SS: function effectively when execution returns from the NMI interrupt processing.

```
BUSLOCK
REPC
NMI → CMPBKB SS: src-block, dst-block
```

2. Poor example

In this example, the number of prefixes is regarded as 3 (all repeat prefixes are regarded as the same), and three return addresses are modified. Actually, however, execution cannot return to BUSLOCK from the interrupt processing routine. It returns to REP because the prefixes take up four addresses.

```
BUSLOCK
REP
REPC
NMI → CMPBK SS: src-block, dst-block
```

6.4 Bus Hold Function

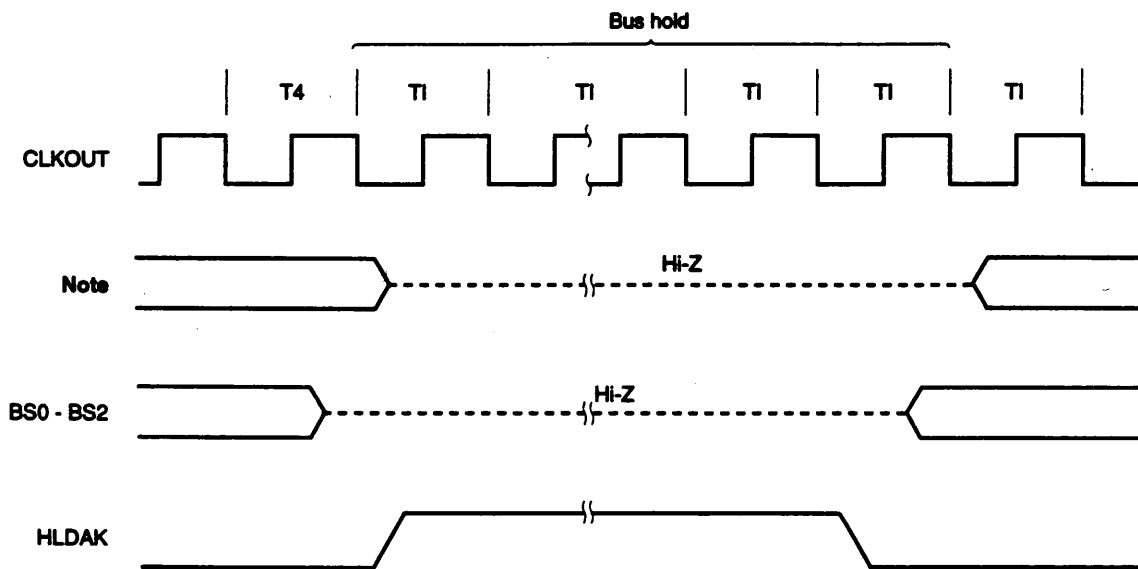
An external device can function as a bus master by requesting the V40HL and V50HL to release the address, address/data, and control buses. When the V40HL or V50HL accepts a hold request (HLDRQ) from the external device, it asserts the HLDAC signal active (high) and releases the bus. While the HLDAC signal is active, the address bus, address/data bus, and three-state output control signals go into a high-impedance state.

If the refresh request or DMA request that has a priority higher than HLDRQ occurs in the hold acknowledge status, HLDAC goes low, and the V40HL or V50HL requests the external device to return bus control (in which HLDRQ is asserted inactive (low)). If an internal and external bus request overlap in this manner, the high-level width of the HLDAC signal is one clock minimum. For the details on the bus priority, refer to 5.4 Bus Arbitration Unit (BAU).

The timing chart in Figure 6-22 illustrates how the V40HL or V50HL enters and ends the bus hold status. Signals BS0 through BS2 have a phase difference of 1/2 clock from the other signals.

Table 6-4 shows the statuses of the output pins in the hold status.

Figure 6-22. Bus Hold Timing



Note A16/PS0-A19/PS3, $\overline{\text{MRD}}$, $\overline{\text{MWR}}$, $\overline{\text{IORD}}$, $\overline{\text{IOWR}}$, $\overline{\text{BUFR}/\text{W}}$, $\overline{\text{BUFEN}}$, $\overline{\text{BUSLOCK}}$, AD0 to AD7 (V40HL), A8 to A15 (V40HL), AD0 to AD15 (V50HL), $\overline{\text{UBE}}$ (V50HL)

Table 6-4. Pin Status in Hold Status

Pin Name	I/O	In Hold Status
AD0 to AD15 ^{Note 1}	3-state I/O	Hi-Z
AD0 to AD7 ^{Note 2}	3-state I/O	Hi-Z
A8 to A15 ^{Note 2}	3-state output	Hi-Z
A16/PS0 to A19/PS3	3-state output	Hi-Z
REFRQ	Output	H
HLDK	Output	H
RESOUT	Output	L
MRD	3-state output	Hi-Z
MWR	3-state output	Hi-Z
IORD	3-state output	Hi-Z
IOWR	3-state output	Hi-Z
ASTB	Output	L
UBE ^{Note 1}	3-state output	Hi-Z
High ^{Note 2}	3-state output	Hi-Z
BUSLOCK	3-state output	Hi-Z
BUF \overline{R} /W	3-state output	Hi-Z
BUFEN	3-state output	Hi-Z
CLKOUT	Output	O
BS0 to BS2	3-state output	Hi-Z
QS0, QS1	Output	O
TOUT2	Output	O
INTAK	Output	H
SRDY		O
TOUT1		O
DMAAK3	Output	H
TxD		O
DMAAK0 to DMAAK2	Output	H
END/TC	I/O	Hi-Z

Notes 1. V50HL only

2. V40HL only

Remark H : high level, L: low level, Hi-Z: high impedance,
O: not fixed (outputs valid value)

6.5 Variable Instruction Cycle Time Function

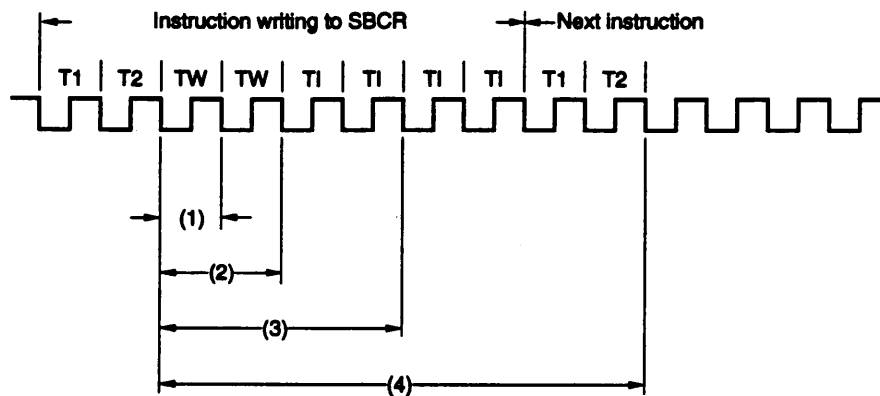
The V40HL and V50HL have a function to reduce the power dissipation by setting the division ratio of the internal clock to SBCR in the system I/O area and thereby reducing the operating frequency.

Table 6-5. Relationship between Division Ratio and Internal Clock

Oscillation Frequency	Division Ratio ^{Note}	Internal Clock (f_{cux})
40 MHz	1/2 (00)	20 MHz
	1/4 (01)	10 MHz
	1/8 (10)	5 MHz
	1/16 (11)	2.5 MHz

Note Parentheses indicate the set value of the CLKC bit of SBCR.

Caution The timing at which the current operating frequency actually changes after a new operating frequency has been set is undefined. However, the operating frequency changes within 8 clocks from the clock next to T2 in the write cycle in which data is written to SBCR, as follows:



- (1) Range of operating frequency change if the frequency division ratio is changed from 1/16
- (2) Range of operating frequency change if the frequency division ratio is changed from 1/8
- (3) Range of operating frequency change if the frequency division ratio is changed from 1/4
- (4) Range of operating frequency change if the frequency division ratio is changed from 1/2

6.6 Standby Function

The V40HL and V50HL have a standby function that sets program execution in the standby status. This function is effected by the HALT instruction.

6.6.1 Features

(1) HALT mode

In this mode, the supply of the clock to the internal circuits of the CPU (except the circuits used to cancel the HALT mode) is stopped.

(2) STOP mode

In this mode, the supply of all the clocks to the CPU and internal I/O is stopped.

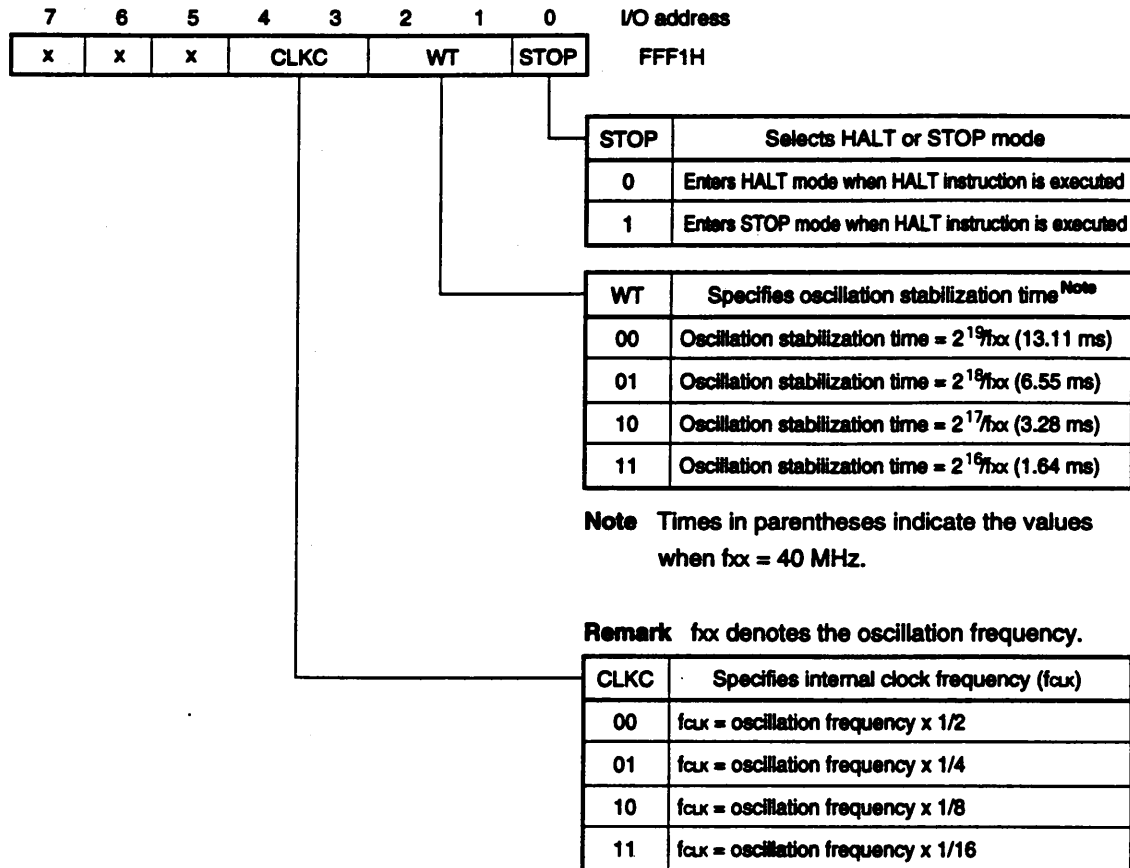
Use this mode when an oscillator is connected to the X1 and X2 pins.

Remark The HALT or STOP mode is selected by the STOP bit of SBCR in the system I/O area.

6.6.2 Standby control register (SBCR)

The standby function of the V40HL and V50HL is controlled by SBCR in the system I/O area.

Figure 6-23. Standby Control Register (SBCR)



Remark x: don't care

The oscillation stabilization time depends on the characteristics of the resonator used and the oscillation circuit. Therefore, set a value that satisfies the oscillation stabilization time, by using the WT bit, based on the result of actual evaluation.

To cancel the STOP mode by using the non-maskable interrupt request (NMI input) or maskable interrupt request (INTPn input), the oscillation stabilization time set by SBCR will be obtained.

To cancel the STOP mode by using the $\overline{\text{RESET}}$ input, obtaining of the oscillation stabilization time is ensured by the active level width of the $\overline{\text{RESET}}$ input. Make the $\overline{\text{RESET}}$ input active for a period of time that is sufficient for the time required for oscillation stabilization.

6.6.3 HALT mode

(1) Setting

The V40HL and V50HL enter the HALT mode when the HALT instruction is executed with the STOP bit of SBCR cleared to 0.

(2) HALT mode

In the HALT mode, the internal clock is supplied to the CPU circuits that have a function necessary for cancelling the HALT mode and those that are related to the bus hold control function. The supply of the clock to the other circuits stops. As a result, the power dissipation can be reduced to about 1/3 of the normal level. Even in the HALT mode, the clock is supplied to peripheral functions, such as the interrupt, DMA, timer/counter, refresh, and bus hold functions, enabling operations in the HALT mode. The output pins when the respective functions do not operate are fixed to the status shown in Table 6-6.

★

Table 6-6. Pin Status in HALT Mode (when peripheral functions do not operate)

Pin Name	I/O	In HALT Mode
AD0 to AD15 ^{Note 1}	3-state I/O	H/L
AD0 to AD7 ^{Note 2}	3-state I/O	H/L
A8 to A15 ^{Note 2}	3-state output	H/L
A16/PS0 to A19/PS3	3-state output	H/L
REFRQ	Output	O
HLD _{AK}	Output	O
RESOUT	Output	L
MRD	3-state output	H
MWR	3-state output	H
IORD	3-state output	H
IOWR	3-state output	H
ASTB	Output	L
UBE ^{Note 1}	3-state output	H
High ^{Note 2}	3-state output	H
BUSLOCK	3-state output	H/L
BUF _{R/W}	3-state output	H/L
BUF _{EN}	3-state output	H
CLKOUT	Output	O
BS0 to BS2	3-state output	H
QS0, QS1	Output	L
TOUT2	Output	O
INTAK	Output	H
SRDY		O
TOUT1		O
DMAAK3	Output	O
TxD		O
DMAAK0 to DMAAK2	Output	O
END/TC	I/O	O

Notes 1. V50HL only

2. V40HL only

Remark H: high level, L: low level, H/L: high or low level,
 Hi-Z: high impedance, O : not fixed (outputs valid value)

(3) Cancelling the HALT mode

The HALT mode that has been once set can be cancelled by using external signals in the following three ways:

(a) Using RESET Input

When the RESET signal is input, the V40HL and V50HL are initialized, and start normal operation by fetching the instruction at address FFFF0H because PS (FFFFH) and PC (0000H) are set anew.

(b) Using Interrupt (NMI Input or INTPn Input)

The HALT mode is also cancelled when a maskable interrupt request that can be acknowledged occurs or when an NMI signal is input. The CPU starts execution of the interrupt processing program. When the RETI instruction is executed at the end of the interrupt processing program, the PC, PS, and PSW are restored from the stack, and execution resumes starting from the instruction next to the HALT instruction.

If INTPn is made active for a duration of 5 clocks or longer in the interrupt disable status (IE = 0), the HALT mode is cancelled, and execution resumes starting from the instruction next to the HALT instruction (the interrupt processing is not executed).

If INTPn is masked (if the M0 through M7 bits of the interrupt mask word register (IMKW) are set to 1), the HALT mode is not cancelled.

(c) Using HLDRQ Input

When the HLDRQ signal (bus hold request) is input, a bus hold acknowledge signal (HLDACK) is immediately generated, the address/data bus and control signals go into the high-impedance state, and bus control is granted to the bus master requesting the bus. When the bus hold request signal is deasserted, the CPU acquires bus control again, and enters the standby status again. Therefore, the bus hold request is not the cancellation of the standby status.

6.6.4 STOP mode

(1) Setting

The V40HL and V50HL enter the STOP mode when the HALT instruction is executed with the STOP bit of SBCR set to 1.

(2) STOP mode

In the STOP mode, the internal clock stops completely. As a result, the power dissipation is reduced to 50 μ A maximum at 5 V (about 1/3600 of the normal operation at 5 V, 20 MHz) and 30 μ A maximum at 3 V (about 1/1800 of the normal operation at 3 V, 10 MHz). All the functions, except TCU (when the TCLK input is used), stop. When the peripheral functions do not operate, the output pins are fixed to the status shown in Table 6-7 in the same manner as in the HALT mode.

When the peripheral functions operate, the output pins retain the bus status immediately before the HALT instruction was executed to set the STOP mode.

Table 6-7. Pin Status in STOP Mode (when peripheral functions do not operate)

Pin Name	I/O	In STOP Mode
AD0 to AD15 ^{Note 1}	3-state I/O	H/L
AD0 to AD7 ^{Note 2}	3-state I/O	H/L
A8 to A15 ^{Note 2}	3-state output	H/L
A16/PS0 to A19/PS3	3-state output	H/L
REFRQ	Output	H
HLD $\overline{\text{AK}}$	Output	L
RESOUT	Output	L
MRD	3-state output	H
MWR	3-state output	H
IORD	3-state output	H
IOWR	3-state output	H
ASTB	Output	L
UBE ^{Note 1}	3-state output	H
High ^{Note 2}	3-state output	H
BUSLOCK	3-state output	H/L
BUF $\overline{\text{R}}$ /W	3-state output	H/L
BUFEN	3-state output	H
CLKOUT	Output	L
BS0, BS1	3-state output	H
BS2	3-state output	L
QS0, QS1	Output	L
TOUT2	Output	H/L
INTAK	Output	H
SRDY		H
TOUT1		H/L
DMAAK3	Output	H
TxD		H
DMAAK0 to DMAAK2	Output	H
END/TC	I/O	Hi-Z

Notes 1. V50HL only

2. V40HL only

Remark H: high level, L: low level, H/L: high or low level,
Hi-Z: high impedance

(3) Cancelling the STOP mode

The STOP mode that has been once set can be cancelled in the following three ways:

(a) Using $\overline{\text{RESET}}$ Input

When the $\overline{\text{RESET}}$ signal is input, the V40HL and V50HL are initialized, and start normal operation by fetching the instruction at address FFFF0H because PS (FFFFH) and PC (0000H) are set anew. It is necessary that the $\overline{\text{RESET}}$ input be made active during the oscillation stabilization time period (if the $\overline{\text{RESET}}$ signal is made inactive midway, the operation after that cannot be guaranteed).

(b) Using maskable Interrupt (INTPn Input)

The STOP mode is also cancelled when an interrupt request that can be acknowledged is input to the INTPn pin.

In the interrupt enable status, the interrupt processing starts as soon as the STOP mode is cancelled. At this time, the INTPn pin must be kept active until the interrupt acknowledge cycle is issued. If it is made inactive midway, either of the following two situations will take place:

- (i) The standby mode is changed from STOP to HALT.
- (ii) Execution will resume starting from the instruction next to the HALT instruction.

In the interrupt disable status, execution resumes starting from the instruction next to the HALT instruction after the STOP mode has been released (interrupt processing is not executed). At this time, the INTPn pin must be kept active during the oscillation stabilization time. If it is made inactive midway, the HALT mode may be set instead of the STOP mode.

If INTPn is masked (if the M0 through M7 bits of the interrupt mask word register (IMKW) are set to 1), the STOP mode is not cancelled.

(c) Using non-maskable Interrupt

If the NMI pin is active (high), the STOP mode is cancelled, and NMI processing is executed after the oscillation stabilization time has elapsed. However, the NMI pin must be kept active during the oscillation stabilization time (until the NMI vector read cycle at address 8). If it is made inactive midway, the HALT mode may be set instead of the STOP mode.

Caution If the NMI pin is made active before the HALT instruction is executed, the STOP mode is not set.

6.7 Reset Function

When a low-level signal is input to the $\overline{\text{RESET}}$ pin for a duration of 4 clocks or longer, the V40HL and V50HL are reset.

6.7.1 CPU reset operation

The CPU is initialized as follows at reset:

Table 6-8. Resetting CPU

Target	Initial Value																																				
PFP	0000H																																				
PC	0000H																																				
PS	FFFFH																																				
SS	0000H																																				
DS0	0000H																																				
DS1	0000H																																				
PSW	<table><tr><td></td><td colspan="4">MD</td><td>V</td><td>DIR</td><td>IE</td><td>BRK</td></tr><tr><td>High</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td></td><td colspan="2">S</td><td>Z</td><td colspan="2">AC</td><td>P</td><td colspan="2">CY</td></tr><tr><td>Low</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr></table>		MD				V	DIR	IE	BRK	High	1	1	1	1	0	0	0	0		S		Z	AC		P	CY		Low	0	0	0	0	0	0	1	0
	MD				V	DIR	IE	BRK																													
High	1	1	1	1	0	0	0	0																													
	S		Z	AC		P	CY																														
Low	0	0	0	0	0	0	1	0																													
Other registers	Undefined																																				
Queue	Clear																																				

If a low-level signal is input to the $\overline{\text{RESET}}$ pin for a duration of 4 clocks or longer and then the $\overline{\text{RESET}}$ pin is made high, the CPU starts prefetching instructions from address FFFF0H.

6.7.2 Reset operations of system I/O area and internal peripheral units

Some of the system I/O area and internal peripheral units are initialized at reset. Tables 6-9 and 6-10 list the system I/O area and internal peripheral units that are reset and the initial values. Those not listed in these tables are undefined at reset because they are not initialized.

Table 6-9. Resetting System I/O Area (1/2)

Target	Initial Value	Remarks
OPCN	<div>7 6 5 4 3 2 1 0</div> <div>- - - - 0 0 0 0</div>	<ul style="list-style-type: none"> INTL1 and INTL2 are pin inputs. DMA channel 3 can be used. INTAK is output from pin.
OPSEL	<div>7 6 5 4 3 2 1 0</div> <div>- - - - 0 0 0 0</div>	SCU, TCU, ICU, and DMAU cannot be used.
SCTL	<div>[V40HL]</div> <div>7 6 5 4 3 2 1 0</div> <div>0 0 0 0 0 0 0 1</div> <div>[V50HL]</div> <div>7 6 5 4 3 2 1 0</div> <div>0 0 0 0 0 0 0 0</div>	<ul style="list-style-type: none"> REFRQ normal timing Input clock of SCU ... TOUT1 Carry is not transferred to A16 in μPD71037 mode. μPD71071 mode is selected. V40HL ... 8-bit boundary V50HL ... 16-bit boundary
WCY2	<div>7 6 5 4 3 2 1 0</div> <div>- - - - 1 1 1 1</div>	Wait state of 3 clocks is inserted in both DMA and refresh cycles.
WCY1	<div>7 6 5 4 3 2 1 0</div> <div>1 1 1 1 1 1 1 1</div>	Wait state of 3 clocks is inserted in both memory and external I/O cycles.
WMB	<div>7 6 5 4 3 2 1 0</div> <div>- 1 1 1 - 1 1 1</div>	Upper and lower memory blocks are set to 512 KB, respectively.
RFC	<div>7 6 5 4 3 2 1 0</div> <div>- - - 0 1 0 0 0</div>	Timer factor (N) = 9
SBCR	<div>7 6 5 4 3 2 1 0</div> <div>0 0 0 0 0 0 0 0</div>	<ul style="list-style-type: none"> HALT mode is set by HALT instruction. Oscillation stabilization time = $2^{19}/f_{osc}$ Internal clock frequency (f_{clk}) = oscillation frequency x 1/2
TCKS	<div>7 6 5 4 3 2 1 0</div> <div>- - - 0 0 0 0 0</div>	Clock of TCT#0 to TCT#2 of TCU is internal clock divided by two.
EXWB	<div>7 6 5 4 3 2 1 0</div> <div>- - - - - 0 0 0</div>	<ul style="list-style-type: none"> Extended division of memory space ... None Extended division of I/O space ... None
WSMB	<div>7 6 5 4 3 2 1 0</div> <div>- - 1 1 - - 1 1</div>	<ul style="list-style-type: none"> Upper submemory block size ... 256 KB Lower submemory block size ... 256 KB
WIOB	<div>7 6 5 4 3 2 1 0</div> <div>- 1 1 1 - 1 1 1</div>	<ul style="list-style-type: none"> Upper I/O submemory block size ... 64 KB Lower I/O submemory block size ... 64 KB

Remark -: Undefined

Table 6-9. Resetting System I/O Area (2/2)

Target	Initial Value								Remarks
WCY3	7	6	5	4	3	2	1	0	<ul style="list-style-type: none"> • Number of upper and lower submemory block wait states ... 3 • Number of upper and lower I/O block wait states ... 3
	1	1	1	1	1	1	1	1	
BRC	7	6	5	4	3	2	1	0	Baud rate counter value: 02H
	0	0	0	0	0	0	0	0	

Remark -: Undefined

Table 6-10. Resetting Internal Peripheral Unit

(a) SCU

Target	Initial Value	Remarks																
SMD	<table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr></table>	7	6	5	4	3	2	1	0	0	1	0	0	1	0	1	1	<ul style="list-style-type: none">• Baud rate ... x 64• Character length ... 7 bits• Parity ... None• Stop bit length ... 1 bit
7	6	5	4	3	2	1	0											
0	1	0	0	1	0	1	1											
SCM	<table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>-</td><td>-</td><td>0</td><td>0</td><td>0</td><td>0</td><td>-</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0	-	-	0	0	0	0	-	0	<ul style="list-style-type: none">• Transmission/reception disabled• SRDY = 1 (mask)
7	6	5	4	3	2	1	0											
-	-	0	0	0	0	-	0											
SIMK	<table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>1</td><td>1</td></tr></table>	7	6	5	4	3	2	1	0	-	-	-	-	-	-	1	1	Interrupt from SCU is masked.
7	6	5	4	3	2	1	0											
-	-	-	-	-	-	1	1											
SST	<table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0	1	0	0	0	0	1	0	0	<ul style="list-style-type: none">• No break detection• No error• Buffer not-ready
7	6	5	4	3	2	1	0											
1	0	0	0	0	1	0	0											

(b) DMAU

Target	Initial Value	Remarks																																
DCH	<table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>-</td><td>-</td><td>-</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table>	7	6	5	4	3	2	1	0	-	-	-	0	0	0	0	1	<ul style="list-style-type: none">DMA channel 0 is selectedCurrent (read)Base and current (write)																
7	6	5	4	3	2	1	0																											
-	-	-	0	0	0	0	1																											
DMD	<table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>-</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0	0	0	0	0	0	0	-	0	<ul style="list-style-type: none">Demand mode with address incrementedAuto initialize disabledVerify transfer, byte transfer																
7	6	5	4	3	2	1	0																											
0	0	0	0	0	0	-	0																											
DDC	<div>[Upper]</div> <table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>0</td><td>0</td></tr></table> <div>[Lower]</div> <table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>-</td><td>-</td><td>0</td><td>0</td><td>-</td><td>0</td><td>-</td><td>-</td></tr></table>	7	6	5	4	3	2	1	0	-	-	-	-	-	-	0	0	7	6	5	4	3	2	1	0	-	-	0	0	-	0	-	-	<ul style="list-style-type: none">Normal write timingFixed priorityDMA enabledBus release modeWait state disabled during verification
7	6	5	4	3	2	1	0																											
-	-	-	-	-	-	0	0																											
7	6	5	4	3	2	1	0																											
-	-	0	0	-	0	-	-																											
DST	<table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0	0	0	0	0	0	0	0	0	<ul style="list-style-type: none">All channels no requestsAll channels not terminated																
7	6	5	4	3	2	1	0																											
0	0	0	0	0	0	0	0																											
DMK	<table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	7	6	5	4	3	2	1	0	-	-	-	-	1	1	1	1	Requests from all channels masked																
7	6	5	4	3	2	1	0																											
-	-	-	-	1	1	1	1																											

Remark -: Undefined

6.7.3 Output pin status at reset

The output pin status is as follows after the $\overline{\text{RESET}}$ signal has been input until the $\overline{\text{RESET}}$ goes high again.

Table 6-11. Pin Status at Reset

Pin Name	I/O	At Reset
AD0 to AD15 ^{Note 1}	3-state I/O	Hi-Z
AD0 to AD7 ^{Note 2}	3-state I/O	Hi-Z
A8 to A15 ^{Note 2}	3-state output	H/L
A16/PS0 to A19/PS3	3-state output	H/L
$\overline{\text{REFRQ}}$	Output	H
HLD $\overline{\text{AK}}$	Output	L
RESOUT	Output	H
$\overline{\text{MRD}}$	3-state output	H
$\overline{\text{MWR}}$	3-state output	H
$\overline{\text{IORD}}$	3-state output	H
$\overline{\text{IOWR}}$	3-state output	H
ASTB	Output	L
$\overline{\text{UBE}}$ ^{Note 1}	3-state output	H/L
High ^{Note 2}	3-state output	H
$\overline{\text{BUSLOCK}}$	3-state output	H
$\overline{\text{BUF}}\overline{\text{R}}/\overline{\text{W}}$	3-state output	H
$\overline{\text{BUFEN}}$	3-state output	H
CLKOUT	Output	○
BS0 to BS2	3-state output	H
QS0, QS1	Output	L
TOUT2	Output	○
$\overline{\text{INTAK}}$	Output	H
$\overline{\text{SRDY}}$		H
TOUT1		H
$\overline{\text{DMAAK3}}$	Output	H
TxD		H
$\overline{\text{DMAAK0}}$ to $\overline{\text{DMAAK2}}$	Output	H
$\overline{\text{END}}/\overline{\text{TC}}$	I/O	Hi-Z

Notes 1. V50HL only

2. V40HL only

Remark H: high level, L: low level, H/L: high or low level,
Hi-Z: high impedance, ○ : not fixed (outputs valid value)

[MEMO]

CHAPTER 7 μ PD8080AF EMULATION

The V40HL and V50HL have two CPU operating modes, the native mode in which the instructions of the V40HL/V50HL are executed, and the emulation mode in which the instructions of the μ PD8080AF are executed. These modes are selected by using a specially prepared instruction or interrupt processing. In addition, as a flag that controls the mode, an MD flag is provided to the most significant bit position of the PSW.

7.1 Changing from Native Mode to Emulation Mode

To change the operating mode from the native mode to the emulation mode, two types of instructions can be used: BRKEM (Break for Emulation) and RETI (Return from Interrupt).

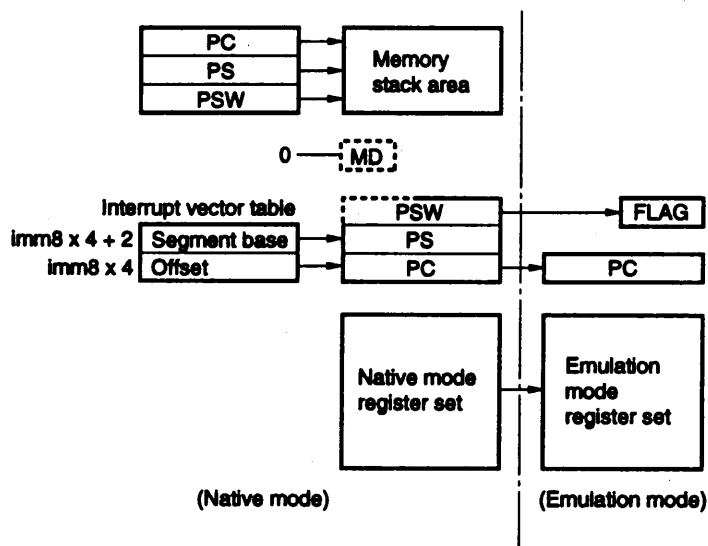
7.1.1 BRKEM Imm8 Instruction

This is a basic instruction that starts the emulation mode. This instruction saves PSW, PS, and PC to the stack, resets the MD flag to 0, and loads the segment base of the interrupt vector specified by the operand to PS and the offset to PC.

In this way, the emulation mode starts as interrupt processing (however, MD = 0). In this mode, program execution starts from an address indicated by PC loaded. The address is in a 64-KB segment space specified by PS loaded. The instruction codes to be fetched in this mode are interpreted and executed as the instructions of the μ PD8080AF (refer to Figure 7-1).

When the BRKEM imm8 instruction is executed, the MD flag is enabled and can be written, and the value saved to the stack is restored to the MD flag when the RETI or POP PSW instruction is executed.

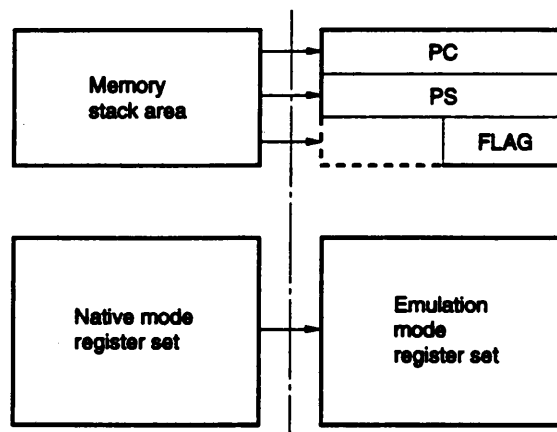
Figure 7-1. Changing Mode by BRKEM Instruction



7.1.2 RETI Instruction

In the native mode (MD = 1), this is a general instruction used when returning from an interrupt routine that was started by the BRK instruction or an external interrupt. In the emulation mode, however, if this instruction is executed at the end of an interrupt processing routine started in the native mode by an interrupt instruction (CALLN) or external interrupt, the mode switches to emulation mode again. This is because the value of the mode flag (MD = 0) that was saved to the stack when the mode was changed from the emulation mode to the native mode is restored to the MD flag when PSW is restored. As a result, the CPU is set in the emulation mode, though PS, PC, and PSW are restored in the same manner as the normal operation by the RETI instruction (refer to Figure 7-2).

Figure 7-2. Changing Mode by RETI Instruction



7.2 Changing Mode from Emulation Mode to Native Mode

The operating mode can be changed from the emulation mode to the native mode in the following four ways:

- (1) $\overline{\text{RESET}}$ input
- (2) NMI input or ICU interrupt request
- (3) CALLN (Call Native) instruction
- (4) RETEM (Return from Emulation) instruction

7.2.1 $\overline{\text{RESET}}$ Input

The normal reset operation is performed in the same manner as in the native mode. Emulation stops.

7.2.2 NMI and ICU request

The normal interrupt operation is performed in the same manner as in the native mode. In the interrupt processing routine in the native mode, the emulation mode can be set by using the RETI instruction, and emulation can be resumed (refer to Figure 7-3).

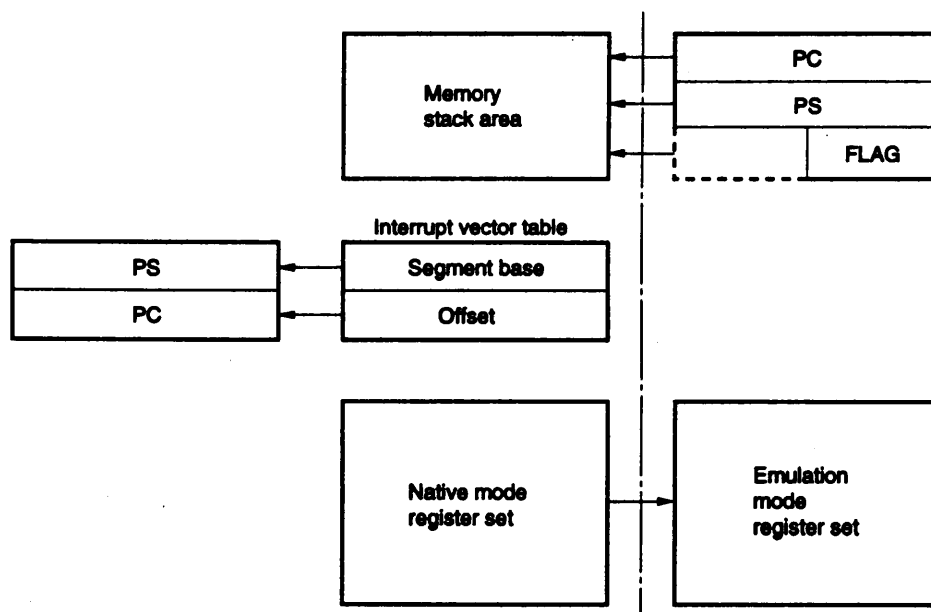
7.2.3 CALLN Imm8 Instruction

This instruction is dedicated to the emulation mode in which an undefined code of the μ PD8080AF has been assigned. When the CALLN instruction is executed in the emulation mode, the CPU saves PS, PC, and PSW to the stack area (at this time, MD = 0 is saved), sets the MD flag to 1, and loads the segment base of an interrupt vector specified by the operand to PS and the offset to PC (refer to Figure 7-3).

The interrupt routine started in the native mode by the CALLN instruction executes the RETI instruction at the end to restore the emulation mode.

By using this function, a program prepared for the V40HL and V50HL can be executed while the program of the μ PD8080AF is executed, and the result of the program of the V40HL/V50HL can be used.

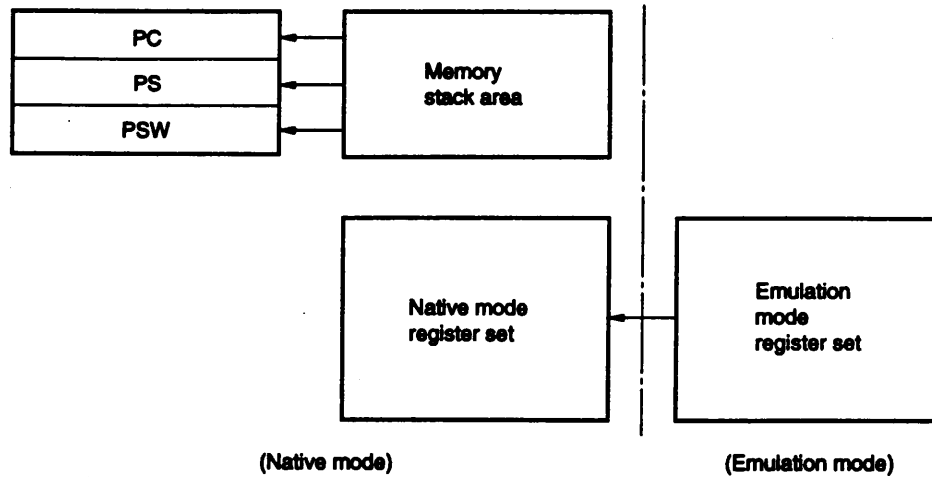
Figure 7-3. Changing Mode by NMI Input, ICU Interrupt Request, and CALLN Instruction



7.2.4 RETEM Instruction

This instruction is dedicated to the emulation mode in which an undefined code of the μ PD8080AF has been assigned. If the RETEM instruction is executed in the emulation mode, the CPU restores PS, PC, and PSW, so that execution returns from interrupt processing, and then returns to the native mode. At this time, the content ("1") of the MD flag saved in the native mode by the BRKEM instruction to the stack area is restored. As a result, the CPU is set in the native mode (refer to Figure 7-4).

Figure 7-4. Changing Mode by RETEM Instruction



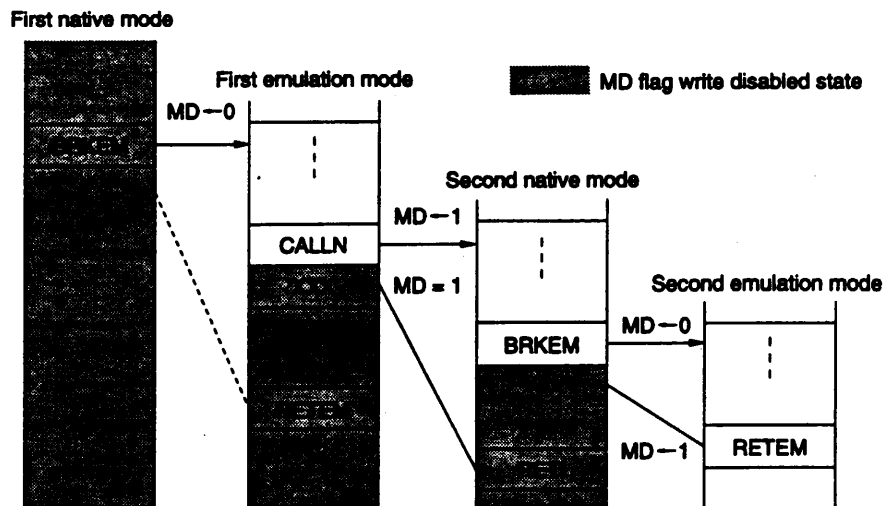
If the RETEM instruction is executed, the MD flag is disabled and cannot be written. Therefore, the MD flag cannot be restored even if the RETI or POP PSW instruction is executed.

7.2.5 Nesting of emulation

The emulation mode cannot be called again by the BRKEM instruction in the native mode called by the NMI, ICU interrupt, or CALLN instruction in the emulation mode.

If emulation is nested in this way, the MD flag is set to an unexpected value, making normal operation impossible.

Figure 7-5. Trouble due to Nesting of Emulation



In Figure 7-5, the CPU which has returned from the second emulation mode to the second native mode, disables writing to the MD flag using the RETEM instruction. Consequently, the MD flag is not restored and remains set to 1 even when the RETI instruction is executed. If instruction XXX of the first emulation is executed in this status, the CPU cannot perform the normal operation because it is in the native mode (MD = 1).

7.3 Emulation Mode

When the operating mode switches between the native mode and emulation mode, the CPU environments are transferred between the modes as shown in Figure 7-6.

The lower 8 bits (AL) of AW, and the higher and lower 8 bits of BW, CW, and DW of the V40HL and V50HL play the roles of the accumulator and six general-purpose registers of the μ PD8080AF.

The lower 8 bits of the V40HL and V50HL take the place of the FLAG of the μ PD8080AF. Each bit of the FLAG corresponds to the bits of the lower byte of the PSW.

Figure 7-6. Correspondence of Register Set

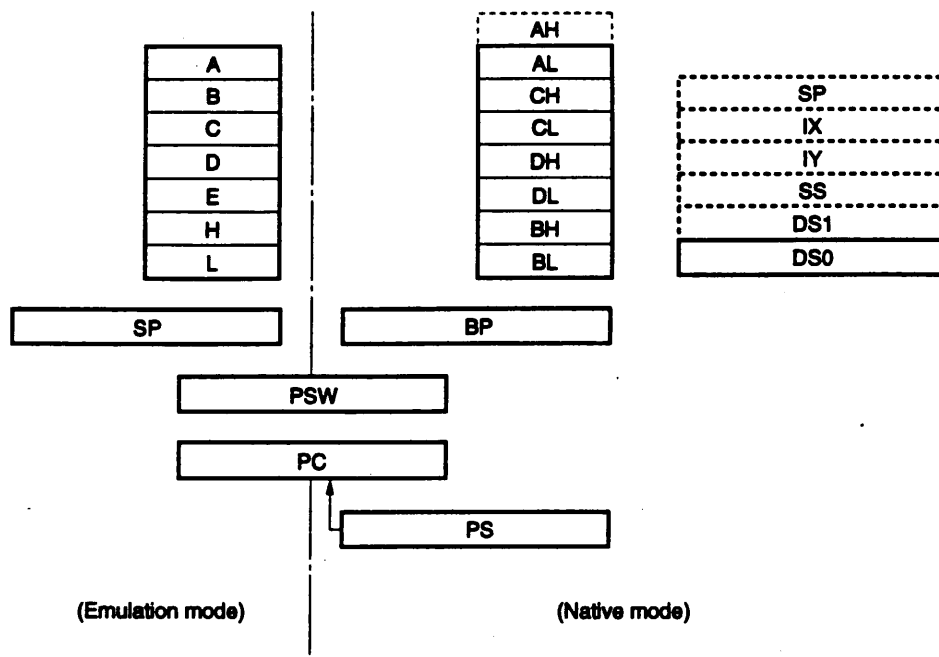


Figure 7-7. Correspondence of PSW and FLAG

9	8	7	6	5	4	3	2	1	0	
	IE	BRK	S	Z	0	AC	0	P	1	CY
										PSW

7	6	5	4	3	2	1	0	
S	Z	0	AC	0	P	1	C	
								FLAG

When the stack is manipulated, SP serves as the stack pointer in the native mode, and BP serves as the stack pointer in the emulation mode. Using an independent stack pointer like this reserves an independent stack area for both mode, and prevents the stack in one mode from being destroyed by improper stack manipulation in another mode.

AH, SP, IX, IY, and the four segment registers (PS, SS, DS0, and DS1) in the native mode are not affected by the emulation mode.

In the emulation mode, the segment base of the program is determined by PS (which is determined by an interrupt vector when the emulation mode is set). The segment base of the memory operand (including stack) is determined by DS0 (whose contents are determined by the programmer immediately before the emulation mode is set).

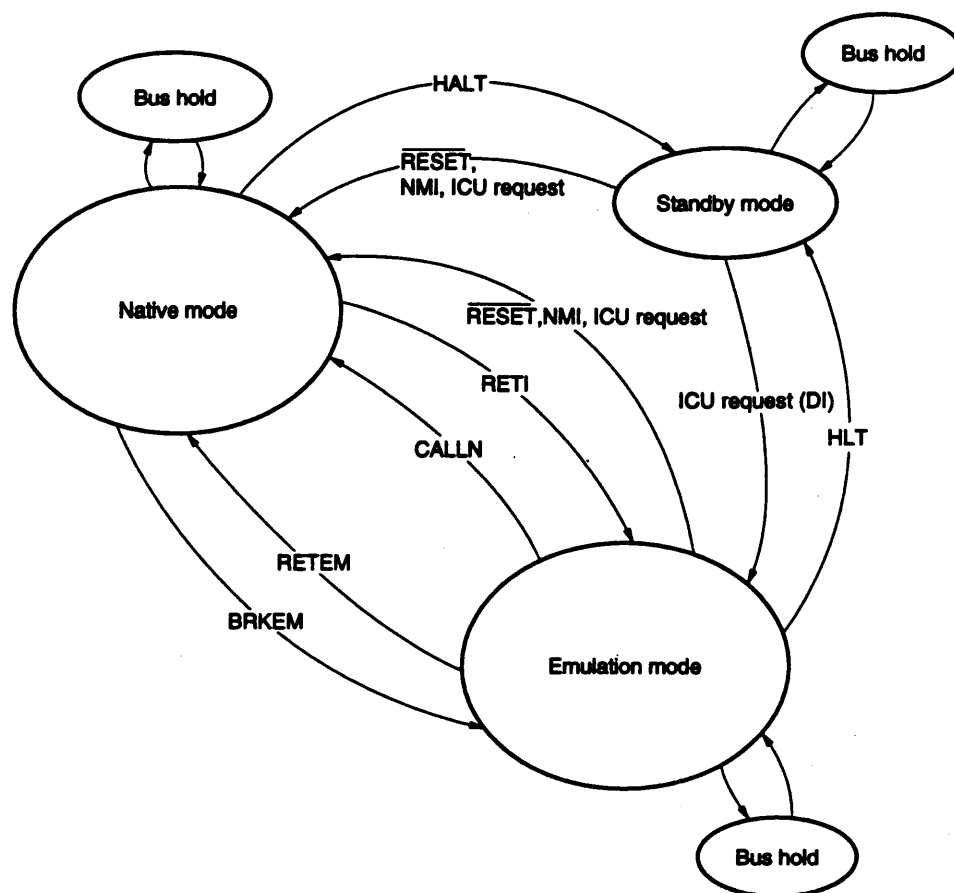
Even in the emulation mode, the bus hold function (hold request/acknowledge) and standby function (due to an HLT instruction) are valid in the same manner as in the native mode.

In the emulation mode, the V40HL and V50HL operate in terms of hardware. Therefore, the input/output operations with the peripheral circuits and memory are performed in the same manner as the V40HL and V50HL. However, the operations that are performed only by the instructions dedicated to the V40HL and V50HL, such as $\overline{\text{BUSLOCK}}$ signal output, are not performed.

PS3 (processor status signal) always outputs a high level in the emulation mode. In the native mode, it always output a low level, except in the refresh cycle.

Figure 7-8 shows the mode transition of the CPU.

Figure 7-8. CPU Mode Transition



When the standby mode is set in the emulation mode, the emulation mode is set again by the ICU interrupt request in the interrupt disable status, and the program is restarted from the instruction next to the HLT instruction. However, if the ICU interrupt request is issued at $\overline{\text{RESET}}$ or NMI input, or in the interrupt enable status, the native mode is set.

In this case, the emulation mode can be restored using the RETI instruction executed in the interrupt routine of NMI or ICU.

When the standby mode is set in the native mode, the original native mode can be restored using $\overline{\text{RESET}}$ or NMI input or an ICU interrupt request (interrupt enable/disable).

[MEMO]

APPENDIX A REGISTER INDEX (IN ALPHABETICAL ORDER)

Abbreviation	Name	Unit	Page
BADR	Bank address register	DMAU (37)	216
BNKR0 to BNKR3	Bank register	DMAU (37)	217
BRC	Baud rate counter	SCU	139
BSEL	Bank selection register	DMAU (37)	217
DBA	DMA base address register	DMAU (71)	195
DBC	DMA base count register	DMAU (71)	194
DCA	DMA current address register	DMAU (71)	195
DCBP	Clear byte pointer F/F	DMAU (37)	215
DCC	DMA current count register	DMAU (71)	194
DCH	DMA channel register	DMAU (71)	192
DCM	Clear mask register	DMAU (37)	215
DDC	DMA device control register	DMAU (71)	196
DICM	DMA initialize command register	DMAU (71)	191
DINT	Initialization	DMAU (37)	215
DMD	DMA mode control register	DMAU (71)	199
DMK	DMA mask register	DMAU (71)	203
DRCA0 to DRCA3	Read current address register	DMAU (37)	207
DRCC0 to DRCC3	Read current count register	DMAU (37)	207
DRST	Read status register	DMAU (37)	208
DST	DMA status register	DMAU (71)	202
DULA	DMAU low address register	System I/O	48
DWAM	Write all mask register	DMAU (37)	213
DWBCA0 to DWBCA3	Write base and current address register	DMAU (37)	207
DWBCC0 to DWBCC3	Write base and current count register	DMAU (37)	207
DWC	Write command register	DMAU (37)	209
DWM	Write mode register	DMAU (37)	214
DWRQ	Write request register	DMAU (37)	211
DWSM	Write single mask register	DMAU (37)	212

APPENDIX A REGISTER INDEX (IN ALPHABETICAL ORDER)

Abbreviation	Name	Unit	Page
EXWB	Extended wait block selection register	WCU	236
IIS	Interrupt in-service register	ICU	168
IIW1	Interrupt initialization word 1 register	ICU	152
IIW2	Interrupt initialization word 2 register	ICU	155
IIW3	Interrupt initialization word 3 register	ICU	156
IIW4	Interrupt initialization word 4 register	ICU	157
IMDW	Interrupt mode word register	ICU	165
IMKW	Interrupt mask word register	ICU	160
IPFW	Interrupt priority finish word register	ICU	161
IPOL	Interrupt polling register	ICU	167
IRQ	Interrupt request register	ICU	167
IULA	ICU low address register	System I/O	48
OPCN	On-chip peripheral connection register	System I/O	42
OPHA	On-chip peripheral high address register	System I/O	47
OPSEL	On-chip peripheral selection register	System I/O	43
RFC	Refresh control register	REFU	245
SBCR	Standby control register	CG	267
SCM	Serial command register	SCU	134
SCTL	System control register	System I/O	44
SIMK	Serial interrupt mask register	SCU	138
SMD	Serial mode register	SCU	132
SRB	Serial receive data buffer	SCU	131
SST	Serial status register	SCU	136
STB	Serial transmit data buffer	SCU	131
SULA	SCU low address register	SCU	47
TCKS	Timer/clock selection register	TCU	104
TCTn	Timer count register	TCU	106
TMD	Timer mode register (mode word)	TCU	102
	Timer mode register (count latch command)		107
	Timer mode register (multiple latch command)		108
TSTn	Timer status register	TCU	109
TULA	TCU low address register	System I/O	47
WCY1	Programmable number of wait cycles register 1	WCU	240
WCY2	Programmable number of wait cycles register 2	WCU	242
WCY3	Programmable number of wait cycles register 3	WCU	241
WIOB	Wait I/O block setting register	WCU	239
WMB	Programmable wait memory area setting register	WCU	234
WSMB	Wait submemory block setting register	WCU	238

Remark DMAU (71): DMAU in the μ PD71071 mode
DMAU (37): DMAU in the μ PD71037 mode

[MEMO]

APPENDIX B GENERAL INDEX**[A]**

A16 to A19 ... 23
A8 to A15 ... 23
AC flag ... 82
AD0 to AD15 ... 22
Address generation ... 65
Address modification circuit (ADM) ... 78
Addressing [DMAU: μ PD71037 mode] ... 204
Addressing [DMAU: μ PD71071 mode] ... 188
Addressing [ICU] ... 149
Addressing [SCU] ... 127
Addressing [TCU] ... 99
ADIR bit [DMD] ... 200
ADIR bit [DWM] ... 215
ADM ... 78
ALU ... 80
Applications ... 17
Architecture ... 35
Arithmetic logic unit (ALU) ... 80
ASTB ... 25
AUTI bit [DMD] ... 200
AUTI bit [DWM] ... 215
Auto initialization ... 220
Auxiliary carry flag (AC) ... 82
AW ... 79

[B]

BADR ... 216
Bank address register (BADR) ... 216
Bank registers (BNKR0 to BNKR3) ... 217
Bank selection register (BSEL) ... 217
BASE bit ... 192, 193
Based addressing ... 68
Based indexed addressing ... 69
Based pointer (BP) ... 79
BAU ... 90
Baud rate clock generation diagram ... 140
Baud rate counter (BRC) ... 139
BCU ... 76
BD bit [TMD] ... 103
BD bit [TSTn] ... 109
BF bit ... 133
BHLD bit ... 198
Bit addressing ... 70
BIU ... 89
BKD bit ... 137
BNK0 bit to BNK3 bit ... 217
BNKR0 to BNKR3 ... 217
BP ... 79
BRC ... 139
Break flag (BRK) ... 84

- BRK flag (single step) interrupt ... 261
- BRK flag ... 84
- BS0 to BS2 ... 27
- BSEL ... 217
- $\overline{\text{BUFEN}}$... 27
- $\text{BUF}\overline{\text{R}}/\text{W}$... 27
- Bus arbitration unit (BAU) ... 90
- Bus control unit (BCU) ... 76
- Bus cycle and wait state validity ... 244
- Bus hold function ... 263
- Bus hold mode ... 198
- Bus hold timing ... 263
- Bus interface unit (BIU) ... 89
- Bus masters list ... 90
- Bus release mode ... 198
- Bus wait operation ... 91
- $\overline{\text{BUSLOCK}}$... 26
- BW ... 79
- [C]**
- Carry flag (CY) ... 81
- Cascade connections ... 222
- Cascade timing examples ... 223
- CE bit ... 45
- CG ... 87
- Character bits ... 128
- CL bit [SMD] ... 133
- $\overline{\text{CL}}$ bit [TMD] ... 108
- Clear byte pointer F/F (DCBP) ... 215
- Clear mask register (DCM) ... 215
- CLKC bit ... 267
- CLKOUT ... 27
- Clock generator (CG) ... 87
- CMODE bit [TMD] ... 103
- CMODE bit [TSTn] ... 109
- Control pin, triggerable one-shot ... 114
- Count end signal output ... 112
- Count latch command ... 107
- Count mode ... 111
- CPU reset ... 274
- CPU ... 75
- CS0 bit ... 104
- CS1 bit ... 104
- CS2 bit ... 104
- CT0 bit to CT2 bit ... 108
- CW ... 79
- CY flag ... 81
- [D]**
- Data addresses ... 66
- Data pointer (DP) ... 77
- Data segment 0 register (DS0) ... 78
- Data segment 0 ... 63
- Data segment 1 register (DS1) ... 78
- Data segment 1 ... 63
- DBA ... 195

- DBC ... 194
 DCA ... 195
 DCBP ... 215
 DCC ... 194
 DCH ... 192
 DCM ... 215
 DDC ... 196
 DDMA bit [DDC] ... 197
 DDMA bit [DWC] ... 210
 DICM ... 191
 DINT ... 215
 DIR flag ... 84
 Direct addressing ... 65, 68
 Direction flag (DIR) ... 84
 DMA base address register (DBA) ... 195
 DMA base count register (DBC) ... 194
 DMA channel register (DCH) ... 192
 DMA control unit (DMAU) ... 179
 DMA current address register (DCA) ... 195
 DMA current count register (DCC) ... 194
 DMA cycle ... 184
 DMA device control register (DDC) ... 196
 DMA initialize command register (DICM) ... 191
 DMA mask register (DMK) ... 203
 DMA mode control register (DMD) ... 199
 DMA request acknowledge wait time ... 93
 DMA status register (DST) ... 202
 DMA timing ... 185
 DMAAK0 to DMAAK2 ... 30
 DMAAK3 ... 29
 DMAM bit ... 45
 DMARQ0 to DMARQ2 ... 30
 DMARQ3 ... 30
 DMAU basic operation ... 183
 DMAU low-address register (DULA) ... 48
 DMAU operation flow ... 183
 DMAU ... 179
 DMAW bit ... 242
 DMD ... 199
 DMK ... 203
 DP ... 77
 DRCA0 to DRCA3 ... 207
 DRCC0 to DRCC3 ... 207
 DRST ... 208
 DS bit ... 43
 DS0 ... 78
 DS1 ... 78
 DST ... 202
 Dual data bus mode ... 71
 DULA ... 48
 DW ... 79
 DWAM ... 213
 DWBCA0 to DWBCA3 ... 207
 DWBCC0 to DWBCC3 ... 207
 DWC ... 209
 DWM ... 214

DWRQ ... 211

DWSM ... 212

Dynamic relocation ... 64

[E]

EAG ... 72, 85

ECL bit ... 135

Effective address generation circuit (EAG) ... 72, 85

Emulation mode ... 279, 283

$\overline{\text{END}}$... 30

EREF bit ... 45

Exception nesting mode ... 165

EXCN bit ... 165

Execution unit (EXU) ... 76

Expanded DMA address ... 216

Expanded nesting mode ... 159

Expanded wait block selection register (EXWB) ... 236

Expanded write timing (DMAU: μ PD71037 mode) ... 209

Expanded write timing (DMAU: μ PD71071 mode) ... 197

Expansion mode ... 173

EXTN bit ... 157

EXU ... 76

EXW bit [DDC] ... 197

EXW bit [DWC] ... 209

EXWB ... 236

[F]

FE bit ... 137

FI bit ... 161

Functional descriptions ... 14

[G]

General description of the ICU registers ... 152

General purpose registers (AW, BW, CW, DW) ... 79

GND ... 31

[H]

HALT mode ... 268

Handling of unused pins ... 31

Hardware triggered strobe (retriggerable) ... 122

HLDK ... 24

HLDRQ ... 23

Hold request acceptance waiting time ... 93

[I]

IC ... 31

ICU interrupt operation ... 254

ICU low-address register (IULA) ... 48

ICU ... 148

Idle cycle ... 183

IE flag ... 84

II4 bit ... 154

IIS ... 168

IIW1 ... 152

IIW2 ... 155

IIW3 ... 156

- IIW4 ... 157
- IL0 bit to IL2 bit ... 161
- IMDW ... 165
- IMKW ... 160
- Immediate addressing ... 67
- Incomplete interrupts ... 178
- Index registers (IX, IY) ... 79
- Indexed addressing ... 69
- Initialization (DINT) ... 215
- Input/output circuit type ... 31
- Instruction addresses ... 65
- Instruction cycle time varying function ... 265
- Instruction decoder ... 86
- INTAK ... 29
- Interfacing with memory ... 36, 37
- Internal block diagram [CG] ... 88
- Internal block diagram [CPU] ... 76
- Internal block diagram [DMAU] ... 180
- Internal block diagram [ICU] ... 148
- Internal block diagram [SCU] ... 126
- Internal block diagram [TCU] ... 98
- Internal block diagram [V40HL] ... 11
- Internal block diagram [V50HL] ... 12
- Internal block functions ... 75
- Internal/external control functions ... 231
- Internal peripheral unit mapping examples ... 46
- Internal peripheral unit reset operations ... 274
- Interrupt acknowledge cycle ... 254
- Interrupt acknowledge timing examples (expansion mode) ... 176, 259
- Interrupt acknowledge timing examples (single mode) ... 171, 257
- Interrupt control unit (ICU) ... 148
- Interrupt enable flag (IE) ... 84
- Interrupt functions ... 250
- Interrupt in-service register (IIS) ... 168
- Interrupt initialization word 1 register (IIW1) ... 152
- Interrupt initialization word 2 register (IIW2) ... 155
- Interrupt initialization word 3 register (IIW3) ... 156
- Interrupt initialization word 4 register (IIW4) ... 157
- Interrupt mask word register (IMKW) ... 160
- Interrupt mode word register (IMDW) ... 165
- Interrupt polling register (IPOL) ... 167
- Interrupt priority, finish word register (IPFW) ... 161
- Interrupt processing during execution of block instruction processing instruction ... 262
- Interrupt request register (IRQ) ... 167
- Interrupt sequence ... 169
- Interrupt sources ... 251
- Interrupt vectors table ... 252
- Interrupts not accepted timings ... 261
- INTP1 to INTP7 ... 29
- I/O addressing ... 67
- IOAG bit ... 45
- I/O mapping and accessing ... 39

I/O read/write timing ... 49

 $\overline{\text{IORD}}$... 25

I/O space dividing into three blocks ... 237

IOSP bit ... 236

IOW bit ... 240

 $\overline{\text{IOWR}}$... 25

IPFW ... 161

IPOL ... 167

IRQ ... 167

IRSW bit ... 42

IS bit ... 43

IS/ $\overline{\text{IR}}$ bit ... 167

IULA ... 48

IX ... 79

IY ... 79

[L]

LC ... 73, 85

LEV bit ... 153

LIOB bit ... 239

LIOW bit ... 241

LMB bit ... 234

LMW bit ... 240

Logical addresses ... 62

Loop counter (LC) ... 73, 85

LSMB bit ... 238

LSMW bit ... 241

[M]

M0 bit to M3 bit [DWAM] ... 213

M0 bit to M7 bit [IMKW] ... 160

M0 to M3 ... 203

Mark status ... 128

MD flag ... 84

Memory addressing ... 68

Memory area 3-partition diagram ... 235

Memory area 5-partition diagram ... 237

Memory indirect addressing ... 66

Memory I/O read and write timing ... 49

Memory mapped I/O configuration ... 40

Memory mapping and access methods ... 35

Memory read/write timing ... 49

MEMSP bit ... 236

Micro address register ... 86

Micro instruction ROM ... 86

Micro instruction sequence circuit ... 86

MMW bit ... 240

Mode flag (MD) ... 84

 $\overline{\text{MRD}}$... 25 $\overline{\text{MRDY}}$ bit ... 135

Multiple latch command ... 108

 $\overline{\text{MWR}}$... 25**[N]**

Native mode ... 279

- NC bit ... 109
- NMI ... 24
- Non-memory addressing ... 66
- Normal FI command ... 162
- Normal nesting mode ... 157
- Normal rotating FI command ... 163
- [O]**
- OL bit ... 109
- On-chip peripheral connection register (OPCN) ... 42
- On-chip peripheral high address register (OPHA) ... 47
- On-chip peripheral relocation register ... 46
- On-chip peripheral selection register (OPSEL) ... 43
- OPCN ... 42
- OPHA ... 47
- OPSEL ... 43
- Ordering information ... 3
- Output pin status at reset ... 278
- OVE bit ... 137
- Overflow flag (V) ... 83
- [P]**
- P flag ... 82
- Parity bit ... 128
- Parity flag (P) ... 82
- PC relative addressing ... 65
- PC ... 73, 85
- PE bit ... 137
- PF bit ... 42
- PFP ... 73, 77
- Physical addresses ... 62
- Pin configuration ... 4
- Pin functions ... 19
- Pin processing when not used ... 31
- Pin status in HALT mode ... 269
- Pin status in STOP mode ... 272
- Pointers (SP, BP) ... 79
- POL bit ... 166
- $\overline{\text{POLL}}$... 27
- Precautions [DMAU] ... 229
- Precautions [SCU] ... 146
- Precautions [TCU] ... 124
- Prefetch pointer (PFP) ... 73, 77
- Prefetch queue (V40HL: Q0 to Q3, V50HL: Q0 to Q5) ... 77
- Program counter (PC) ... 73, 85
- Program segment register (PS) ... 78
- Program segment ... 62
- Program status word (PSW) ... 80
- Programmable wait cycle number register 1 (WCY1) ... 240
- Programmable wait cycle number register 2 (WCY2) ... 242
- Programmable wait cycle number register 3 (WCY3) ... 241
- Programmable wait memory area register (WMB) ... 234
- PS bit [SMD] ... 133
- PS bit [TCKS] ... 104
- PS ... 78

-
- PS0 to PS3 ... 23
 - PSW ... 80
 - [Q]**
 - Q0 to Q5 ... 77
 - QS0, QS1 ... 28
 - [R]**
 - Rate generator ... 116
 - RBRDY bit ... 137
 - RE bit ... 135
 - Read current address registers (DRCA0 to DRCA3) ... 207
 - Read current count registers (DRCC0 to DRCC3) ... 207
 - Read status register (DRST) ... 208
 - READY ... 24
 - Ready timing (1 wait) ... 232
 - Ready timing (no wait) ... 231
 - Refresh address ... 245
 - Refresh control register (RFC) ... 245
 - Refresh control unit (REFU) ... 97
 - Refresh cycle wait ... 242
 - Refresh functions ... 245
 - Refresh timing ... 247
 - $\overline{\text{REFRQ}}$... 23
 - REFU bus control ... 247
 - REFU ... 97
 - Register addressing ... 66
 - Register indirect addressing ... 65, 68
 - Relationship between WCU and READY pin ... 243
 - RES bit ... 191
 - $\overline{\text{RESET}}$... 24
 - $\overline{\text{RESET}}$ and READY signal synchronization ... 89
 - Reset functions ... 274
 - RESOUT ... 24
 - RFC ... 245
 - RFE bit ... 245
 - RFW bit ... 242
 - RM bit ... 138
 - ROT bit [DDC] ... 197
 - ROT bit [DWC] ... 210
 - RP bit ... 161
 - RQ0 bit to RQ3 bit [DRST] ... 208
 - RQ0 bit to RQ3 bit [DST] ... 202
 - RTM bit ... 245
 - RWM bit [TMD] ... 103
 - RWM bit [TSTn] ... 109
 - RxD ... 30
 - [S]**
 - S flag ... 83
 - S1 bit to S7 bit ... 156
 - SBCR ... 267
 - SBRK bit ... 135
 - SC bit [SCTL] ... 45
 - SC bit [TMD] ... 103, 107
 - SCM ... 134

-
- SCTL ... 44
 - SCU low-address register (SULA) ... 47
 - SCU operation procedure ... 130
 - SCU ... 125
 - Segment mode ... 62
 - Segment registers (PS, SS, DS0, DS1) ... 78
 - SEL bit ... 192
 - SELCH bit [DCH] ... 193
 - SELCH bit [DWM] ... 215
 - SELCH bit [DWRQ] ... 211
 - SELCH bit [DWSM] ... 212
 - Self FI command ... 164
 - Serial command register (SCM) ... 134
 - Serial control unit (SCU) ... 125
 - Serial data format ... 128
 - Serial interrupt masking register (SIMK) ... 138
 - Serial mode register (SMD) ... 132
 - Serial receive data buffer (SRB) ... 131
 - Serial receive operation ... 144
 - Serial status register (SST) ... 136
 - Serial transmit data buffer (STB) ... 131
 - Serial transmit operation ... 142
 - SFI bit ... 159
 - Sign flag (S) ... 83
 - SIL bit ... 161
 - SIMK ... 138
 - Single mode ... 169
 - $\overline{\text{SL}}$ bit ... 108
 - SMD ... 132
 - SMK bit ... 212
 - SNGL bit ... 154
 - SNM bit ... 165
 - Software-triggered strobe ... 120
 - SP ... 79
 - Specified circuit command ... 163
 - Specified circuit FI command ... 163
 - Specified FI command ... 162
 - Speeding up instruction execution ... 71
 - Square wave generator ... 118
 - SR bit ... 167
 - SRB ... 131
 - $\overline{\text{SRDY}}$... 29
 - SRQ bit ... 211
 - SS bit ... 43
 - SS ... 78
 - SST ... 136
 - Stack pointer (SP) ... 79
 - Stack segment register (SS) ... 78
 - Stack segment ... 63
 - Standby control register (SBCR) ... 267
 - Standby function ... 266
 - Start bit ... 128
 - Status transition diagram ... 225
 - STB ... 131

- STL bit ... 132
- STOP bit ... 267
- STOP mode ... 271
- Stop bit ... 129
- SULA ... 47
- Supply a clock to the TCU ... 105
- System control register (SCTL) ... 44
- System I/O area reset operation ... 274
- System I/O area ... 41
- [T]**
- TA ... 72, 80
- Table of pin status during cascading ... 224
- Table of pin status in hold status ... 264
- TB ... 72, 80
- TBRDY bit ... 137
- \overline{TC} ... 30
- TC ... 186
- TC0 bit to TC3 bit [DRST] ... 208
- TC0 bit to TC3 bit [DST] ... 202
- TCKS ... 104
- TCLK ... 29
- TCT#0 to TCT#2 ... 106
- TCTL2 ... 29
- TCU basic operation procedure ... 100
- TCU low-address register (TULA) ... 47
- TCU ... 98
- TDIR bit [DMD] ... 200
- TDIR bit [DWM] ... 215
- \overline{TE} bit ... 135
- TEMP ... 77
- Temporary communications register (TEMP) ... 77
- Temporary register C (TC) ... 80
- Temporary register/shifter A/B (TA/TB) ... 72, 80
- Terminal count (TC) ... 186
- Timer/clock selection register (TCKS) ... 104
- Timer/counter unit (TCU) ... 98
- Timer count registers #0 to #2 (TCT#0 to TCT#2) ... 106
- Timer mode register (TMD) ... 102, 107, 108
- Timer status registers (TSTn) ... 109
- Timing of bus priority switching ... 92
- Timing when interrupts not acknowledged ... 261
- TM bit ... 138
- TMD [count latch command] ... 107
- TMD [mode word] ... 102
- TMD [multiple latch command] ... 108
- TMD ... 102
- TMODE bit [DMD] ... 199
- TMODE bit [DWM] ... 215
- TOUT1 ... 29
- TOUT2 ... 29
- Transfer mode ... 218
- Transmission and reception baud rates ... 139
- TS bit ... 43
- TSTn ... 109

TULA ... 47

TxD ... 29

[U] $\overline{\text{UBE}}$... 26

UIOB bit ... 239

UIOW bit ... 241

UMB bit ... 234

UMW bit ... 240

USMB bit ... 238

USMW bit ... 241

[V]

V flag ... 83

V3 bit to V7 bit ... 155

V40 and V50 differences ... 13

V_{DD} ... 30**[W]**W/ $\overline{\text{B}}$ bit ... 200

Wait by READY pin ... 231

Wait by WCU ... 233

Wait control unit (WCU) ... 96

Wait function ... 231

Wait I/O block setting register (WIOB) ... 239

Wait in CPU cycle ... 233

Wait in DMA cycle ... 242

Wait submemory block setting register (WSMB) ... 238

WCU ... 96

WCU / READY pin relationship ... 243

WCY1 ... 240

WCY2 ... 242

WCY3 ... 241

WEV bit ... 198

WIOB ... 239

WMB ... 234

Write all mask register (DWAM) ... 213

Write base and current address registers (DWBCA0 to DWBCA3) ... 207

Write base and current count registers (DWBCC0 to DWBCC3) ... 207

Write command register (DWC) ... 209

Write mode register (DWM) ... 214

Write request register (DWRQ) ... 211

Write single mask register (DWSM) ... 212

WSMB ... 238

WT bit ... 267

[X]

X1, X2 ... 27

[Z]

Z flag ... 83

Zero flag (Z) ... 83

[Others]

68-pin plastic QFJ [V40HL] ... 3, 6

68-pin plastic QFJ [V50HL] ... 3, 9

80-pin plastic QFP [V40HL] ... 3, 4

80-pin plastic QFP [V50HL] ... 3, 7

80-pin plastic TQFP [V40HL] ... 3, 5

80-pin plastic TQFP [V50HL] ... 3, 8

μ PD71037 mode ... 204

μ PD71071 mode ... 188

μ PD8080AF emulation ... 279

μ PD71037 mode / μ PD71037 differences ... 181

μ PD71071 mode / μ PD71037 mode differences ... 181

μ PD71071 mode / μ PD71071 differences ... 181