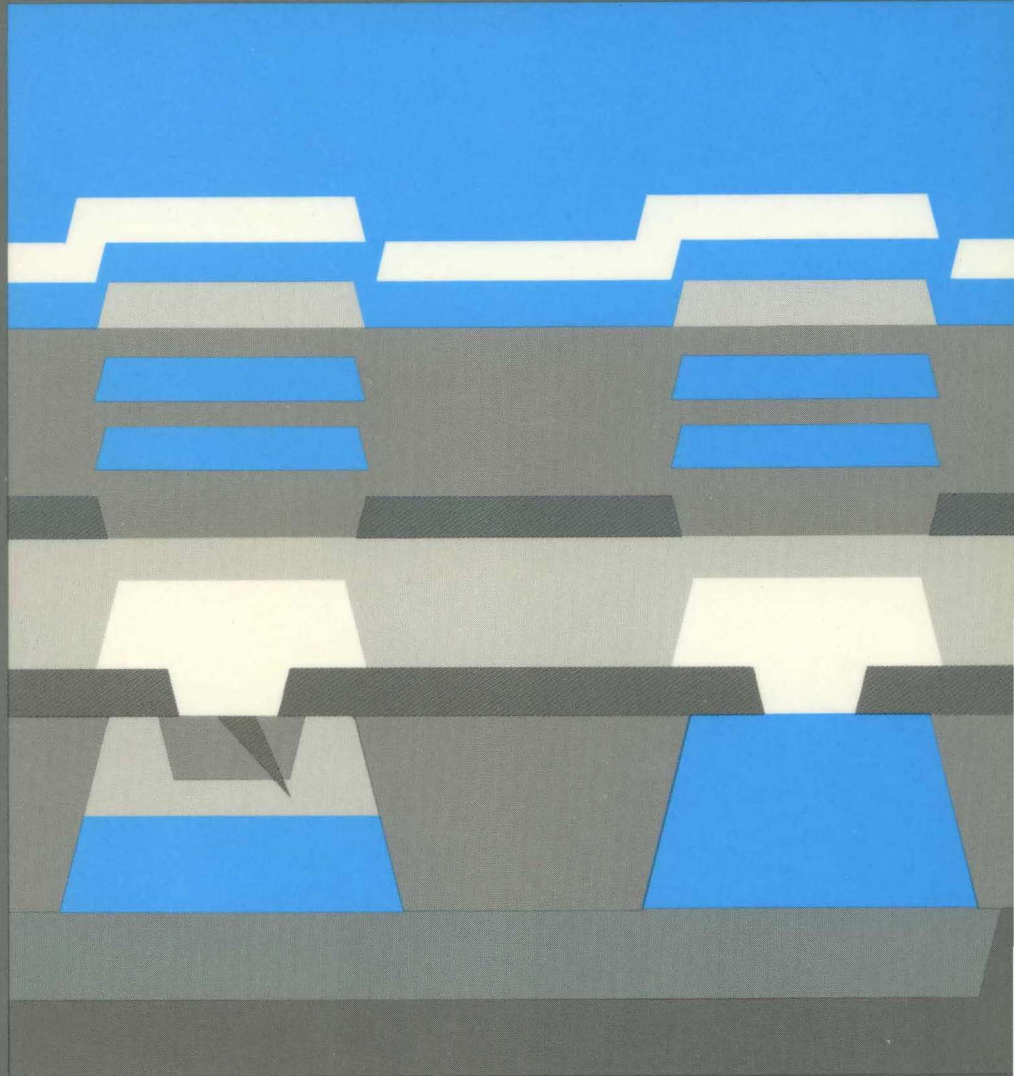


 HITACHI®

 HITACHI®

HD64180S NPU  
HARDWARE MANUAL



HD64180S NPU Hardware Manual

#U16

#U16

# HD64180S NPU Hardware Manual

When using this document, keep the following in mind:

1. This document may, wholly or partially, be subject to change without notice.
2. All rights are reserved: No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without Hitachi's permission.
3. Hitachi will not be held responsible for any damage to the user that may result from accidents or any other reasons during operation of the user's unit according to this document.
4. Circuitry and other examples described herein are meant merely to indicate the characteristics and performance of Hitachi's semiconductor products. Hitachi assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples described herein.
5. No license is granted by implication or otherwise under any patents or other rights of any third party or Hitachi, Ltd.
6. **MEDICAL APPLICATIONS:** Hitachi's products are not authorized for use in **MEDICAL APPLICATIONS** without the written consent of the appropriate officer of Hitachi's sales company. Such use includes, but is not limited to, use in life support systems. Buyers of Hitachi's products are requested to notify the relevant Hitachi sales offices when planning to use the products in **MEDICAL APPLICATIONS**.

# PREFACE

The HD64180S, network processing unit (NPU), provides multipurpose high-speed communication control functions on a single LSI chip. The HD64180S offers high performance communication protocol processing, as well as user system application processing, at a low cost.

Built-in features, such as an 8-bit CPU, 2 serial I/O channels, and a direct memory access controller (DMAC), support high-speed data transfer by reducing communications overheads.

The HD64180S has a variety of applications. It can be used as a communication subsystem processor or as a controller in a distributed control system for industrial robots.

In addition, the HD64180S is designed to interface with existing communication chips and to be compatible with existing communication software. It can be used with virtually any kind of communication system.

This manual describes HD64180S hardware. For details about programming instructions refer to the HD64180 Programming Manual (#U92).





# How to Use This Manual

This User's Manual provides details about the performance and functions of the HD64180S in addition to information about possible applications. For details about the communication protocols supported by the HD64180S, please refer to published documents.

## Section Contents

This manual consists of fourteen sections and seven appendices.

### – Section 1 Overview

This section outlines the internal configuration and functional blocks (CPU, MSCI, ASCI/CSIO, DMAC, etc.) of the HD64180S, and gives sample applications.

### – Section 2 Pin Assignments and Signal Descriptions

### – Section 3 CPU

This section provides details about the architecture of the built-in CPU, basic operation timing, and chip operating modes (sleep, system stop, etc.). Information about CPU interrupt processing, memory management functions, and associated registers is also provided.

### – Section 4 MSCI

This section gives a general description of the asynchronous, byte synchronous, and bit synchronous communication protocols supported by the built-in multiprotocol serial communications interface (MSCI). It describes how to set registers for various communication functions.

### – Section 5 ASCI/CSIO

This section provides a general description of the asynchronous and clock synchronous communication protocols supported by the built-in asynchronous serial communication interface/clocked serial I/O port (ASCI/CSIO). It also describes how to set registers for various communication functions.

– **Section 6 DMAC**

This section explains the single- and chained-block transfer modes supported by the built-in direct memory access controller (DMAC). Descriptions of internal register functions and its setting are also given.

– **Section 7 Timers**

This section describes built-in timer functions, such as external event signal counting and square waveform generation. Descriptions of internal register functions are also given.

– **Section 8 Refresh Controller**

This section provides information about the built-in refresh controller which makes programming of the DRAM refresh cycle. Descriptions of internal register functions are also given.

– **Section 9 Wait Controller**

This section describes the built-in wait controller which inserts wait states during memory access to one of three physical address spaces or during I/O accesses. Details about the associated internal registers and  $\overline{\text{WAIT}}$  pin are also provided.

– **Section 10 Chip Select Control**

The chip select pins ( $\overline{\text{CS0}}$ ,  $\overline{\text{CS1}}$ , and  $\overline{\text{CS2}}$ ) which indicate access to one of three physical address spaces are described in this section.

– **Section 11 Low Power Dissipation Modes**

This section describes the low power dissipation modes (sleep and system stop).

– **Section 12 Oscillator Circuit**

This section explains the clock supply function and shows how to connect a crystal resonator to the built-in oscillator circuit. This section also gives details about board design and using an external clock.

**– Section 13 Electrical Specifications**

This section lists the electrical characteristics (absolute maximum ratings, recommended operating conditions, DC and AC characteristics) and provides timing diagrams.

**– Section 14 Package Dimensions**

This section shows the dimensions of the HD64180S package.

See sections 1, 4, 5, and 6 for details about the communication functions provided by the HD64180S.



# CONTENTS

Section 1. Overview.....	1
1.1 Overview .....	1
1.2 Applications .....	4
1.2.1 Position in Product Line .....	4
1.2.2 Examples of System Configuration.....	5
Section 2. Pin Assignments and Signal Descriptions.....	9
2.1 Pin Assignments.....	9
2.2 Signal Descriptions .....	10
2.2.1 Power Supply .....	10
2.2.2 Clock .....	10
2.2.3 Reset Line.....	11
2.2.4 Address Lines .....	12
2.2.5 Data Lines .....	12
2.2.6 Memory and I/O Interface Lines .....	12
2.2.7 System Control Lines .....	14
2.2.8 Interrupt Lines .....	15
2.2.9 DMA Lines.....	16
2.2.10 Serial I/O (MSCI) Lines .....	17
2.2.11 Serial I/O (ASCI/CSIO) Lines .....	18
2.2.12 Timer Lines .....	19
Section 3. Central Processing Unit (CPU).....	20
3.1 Overview .....	20
3.2 Basic CPU Architecture .....	20
3.2.1 CPU Internal Registers.....	20
3.2.2 Addressing Modes.....	24
3.2.3 Instruction Set .....	28
3.2.4 I/O Space .....	30
3.3 CPU Basic Operation Timing.....	31
3.3.1 Outline.....	31
3.3.2 Opcode Fetch Timing .....	32
3.3.3 Memory Data Read/Write Timing.....	35
3.3.4 I/O Read/Write Timing.....	37
3.3.5 Basic Instruction Execution Timing.....	39
3.3.6 Interfacing Z80-based Peripheral LSIs.....	40

3.4	Chip Operation Modes .....	48
3.4.1	Outline .....	48
3.4.2	Reset Mode .....	50
3.4.3	Normal Operation Mode .....	52
3.4.4	Halt Mode .....	52
3.4.5	Sleep Mode .....	55
3.4.6	System Stop Mode .....	58
3.5	Bus Arbiter .....	59
3.5.1	Overview .....	59
3.5.2	Timing for Passing Bus Control .....	61
3.5.3	Bus Release Mode .....	62
3.5.4	Bus Control Passing .....	65
3.6	Interrupts .....	66
3.6.1	Overview .....	66
3.6.2	Interrupt Control Registers and Interrupt Enable Flags .....	67
3.6.3	TRAP .....	80
3.6.4	Nonmaskable Interrupt (NMI) .....	83
3.6.5	$\overline{INT0}$ .....	86
3.6.6	$\overline{INT1}$ , $\overline{INT2}$ , and the Internal Interrupts (Except TRAP) .....	92
3.6.7	Initial Values of Flags and Registers Associated with Interrupts .....	97
3.6.8	Control Signals for $\overline{INT0}$ , $\overline{INT1}$ , $\overline{INT2}$ , and Internal Interrupts Except TRAP .....	97
3.7	Memory Management Unit (MMU) .....	98
3.7.1	Overview .....	98
3.7.2	MMU Registers .....	99
3.7.3	MMU Operating Space .....	100
3.7.4	MMU Operation .....	101
3.7.5	MMU and Reset .....	104
3.7.6	MMU Operating Precautions .....	105
Section 4. Multiprotocol Serial Communications Interface (MSCI) .....		106
4.1	Overview .....	106
4.1.1	Functions .....	106
4.1.2	Configuration and Operation .....	107
4.1.3	Registers .....	114
4.2	Registers .....	115
4.2.1	MSCI Mode Register 0 (MMD0) .....	115
4.2.2	MSCI Mode Register 1 (MMD1) .....	120



4.2.3	MSCI Mode Register 2 (MMD2)	124
4.2.4	MSCI Control Register (MCTL)	127
4.2.5	MSCI RX Clock Source Register (MRXS)	131
4.2.6	MSCI TX Clock Source Register (MTXS)	134
4.2.7	MSCI Time Constant Register (MTMC)	136
4.2.8	MSCI Command Register (MCMD)	137
4.2.9	MSCI Status Register 0 (MST0)	141
4.2.10	MSCI Status Register 1 (MST1)	145
4.2.11	MSCI Status Register 2 (MST2)	149
4.2.12	MSCI Status Register 3 (MST3)	157
4.2.13	MSCI Frame Status Register (MFST)	160
4.2.14	MSCI Interrupt Enable Register 0 (MIE0)	162
4.2.15	MSCI Interrupt Enable Register 1 (MIE1)	164
4.2.16	MSCI Interrupt Enable Register 2 (MIE2)	168
4.2.17	MSCI Frame Interrupt Enable Register (MFIE)	171
4.2.18	MSCI Synchronous/Address Register 0 (MSA0)	172
4.2.19	MSCI Synchronous/Address Register 1 (MSA1)	174
4.2.20	MSCI Idle Pattern Register (MIDL)	175
4.2.21	MSCI TX/RX Buffer Register (MTRB)	176
4.3	Operation	177
4.3.1	Asynchronous Mode	177
4.3.2	Byte Asynchronous Mode	192
4.3.3	Bit Synchronous Mode	200
4.3.4	Bit Synchronous Loop Mode	209
4.4	Transmit/Receive Clock Selection	215
4.4.1	Overview	215
4.4.2	Supplying the Transmit Clock	216
4.4.3	Supplying the Receive Clock	217
4.4.4	Baud Rate Generator	219
4.4.5	ADPLL	219
4.5	ADPLL	220
4.5.1	Overview	220
4.5.2	Operation	224
4.5.3	Precautions for Using the ADPLL	229
4.5.4	Precaution	229
4.5.5	Detailed Description	230
4.5.6	Examples of Generating the Enter Search Mode Command	231
4.6	Baud Rate Generator	231
4.6.1	Overview	231
4.6.2	Functions	232
4.6.3	Register Set Values and Bit Rates	234

4.7	Internal Interrupts .....	241
4.7.1	Interrupt Types and Sources .....	241
4.7.2	Interrupt Clear .....	241
4.7.3	Interrupt Enable Conditions .....	244
4.8	Application Examples .....	246
4.8.1	Serial Data Transfer by the CPU and DMAC .....	246
4.8.2	Maximum Bit Rate .....	247
4.8.3	Example of Transmit by Programmed I/O (Bi-sync Mode).....	249
4.8.4	Example of Receive by Programmed I/O (Bi-sync Mode) .....	253
4.8.5	Example of Transmit in DMA Chained Block Transfer Mode (bit synchronous HDLC mode) .....	256
4.8.6	Example of Receive in DMA Chained Block Transfer Mode (bit synchronous HDLC mode) .....	257
4.9	Reset Operation .....	258

## Section 5. Asynchronous Serial Communications Interface/

	Clocked Serial I/O Port (ASCI/CSIO) .....	259
5.1	Overview .....	259
5.1.1	Functions .....	259
5.1.2	Configuration and Operation.....	260
5.1.3	Registers .....	263
5.2	Registers .....	264
5.2.1	ASCI Mode Register 0 (MD0) .....	264
5.2.2	ASCI Mode Register 1 (MD1) .....	267
5.2.3	ASCI Mode Register 2 (MD2) .....	269
5.2.4	ASCI Control Register (CTL) .....	271
5.2.5	ASCI RX Clock Source Register (RXS) .....	272
5.2.6	ASCI TX Clock Source Register (TXS) .....	274
5.2.7	ASCI Time Constant Register (TMC).....	277
5.2.8	ASCI Command Register (CMD) .....	278
5.2.9	ASCI Status Register 0 (ST0) .....	280
5.2.10	ASCI Status Register 1 (ST1) .....	283
5.2.11	ASCI Status Register 2 (ST2) .....	287
5.2.12	ASCI Status Register 3 (ST3) .....	290
5.2.13	ASCI Interrupt Enable Register 0 (IE0) .....	292
5.2.14	ASCI Interrupt Enable Register 1 (IE1) .....	294

5.2.15	ASCI Interrupt Enable Register 2 (IE2) .....	297
5.2.16	ASCI TX/RX Buffer Register (TRB).....	299
5.3	Operation.....	300
5.3.1	Asynchronous Mode.....	300
5.3.2	Clocked Serial Mode .....	313
5.4	Transmit/Receive Clock Selection .....	317
5.4.1	Overview .....	317
5.4.2	Supplying the Transmit Clock.....	319
5.4.3	Supplying the Receive Clock .....	319
5.4.4	Baud Rate Generator .....	319
5.5	Baud Rate Generator .....	320
5.5.1	Overview .....	320
5.5.2	Functions .....	321
5.5.3	Register Set Values and Bit Rates .....	322
5.6	Internal Interrupts.....	328
5.6.1	Interrupt Types and Sources .....	328
5.6.2	Interrupt Clear .....	329
5.6.3	Interrupt Request Conditions.....	331
5.7	Application Examples .....	333
5.7.1	Serial Data Transfer by the CPU .....	333
5.7.2	Maximum Bit Rates.....	333
5.8	Generating MSCI-compatible Programs.....	335
5.8.1	MSCI Compatibility .....	335
5.8.2	Precautions for Generating MSCI-compatible Programs.....	335
5.9	Reset Operation.....	336
Section 6. Direct Memory Access Controller (DMAC) .....		337
6.1	Overview .....	337
6.1.1	Functions .....	337
6.1.2	Configuration and Operation.....	339
6.1.3	Registers .....	341
6.2	Registers.....	343
6.2.1	Destination Address Register (DAR) L, H, B (Buffer Address Register (BAR) L, H, B) .....	343
6.2.2	Source Address Register (SAR) L, H, B (Chain Pointer Base (CPB)) .....	344
6.2.3	Current Descriptor Address Register (CDA) L, H.....	345
6.2.4	Error Descriptor Address Register (EDA) L, H.....	346

6.2.5	Receive Buffer Length (BFL) L, H .....	346
6.2.6	Byte Count Register (BCR) L, H .....	347
6.2.7	DMA Status Register (DSR) .....	348
6.2.8	DMA Mode Register A (DMRA) .....	352
6.2.9	DMA Mode Register B (DMRB) .....	355
6.2.10	Frame-End Interrupt-Counter (FCT) .....	358
6.2.11	DMA Interrupt Enable Register (DIR) .....	360
6.2.12	DMA Command Register (DCR) .....	362
6.2.13	DMA Priority Control Register (PCR) .....	363
6.2.14	DMA Master Enable Register (DMER) .....	365
6.3	Descriptor .....	368
6.3.1	Chained-block Transfers from Memory to the MSCI (Transmit) .....	368
6.3.2	Chained-block Transfers from the MSCI to Memory (Receive) .....	371
6.4	Operating Modes .....	372
6.4.1	Overview .....	372
6.4.2	Single-block Transfers Between Memory and Memory (Dual Address) .....	374
6.4.3	Single-block Transfers Between Memory and I/O (Memory-Mapped I/O) (Dual Address) .....	377
6.4.4	Single-block Transfers Between Memory and the MSCI (Single Address) .....	382
6.4.5	Chained-block Transfers from Memory to the MSCI .....	387
6.4.6	Chained-block Transfers from the MSCI to Memory .....	400
6.4.7	Characteristics .....	413
6.5	Connections Between the DMAC and MSCI .....	413
6.6	Internal Interrupts .....	415
6.7	Reset Operation .....	417
6.8	Precautions .....	418
Section 7. Timers .....		419
7.1	Overview .....	419
7.1.1	Functions .....	419
7.1.2	Configuration and Operation .....	420
7.1.3	Registers .....	421
7.2	Registers .....	421
7.2.1	Timer Up-counter (TCNT) .....	421
7.2.2	Timer Constant Register (TCNR) .....	422
7.2.3	Timer Control/Status Register (TCSR) .....	422

7.2.4	Timer Expand Prescale Register (TEPR) .....	425
7.3	Operation Timing .....	426
7.3.1	Timer Count-up Timing.....	426
7.3.2	Output Timing .....	429
7.4	Internal Interrupt .....	429
7.5	Low Power Dissipation Mode.....	430
7.6	Reset Operation.....	430
7.7	Precautions .....	431
 Section 8. Refresh Controller .....		 432
8.1	Overview .....	432
8.1.1	Functions .....	432
8.1.2	Configuration and Operation.....	433
8.1.3	Register.....	433
8.2	Register .....	434
8.2.1	Refresh Control Register (RCR) .....	434
8.3	Operation.....	435
8.4	Refresh Controller Operation in Low Power Dissipation Mode.....	436
8.5	Reset Operation .....	436
8.6	Precautions .....	437
 Section 9. Wait Controller .....		 438
9.1	Overview .....	438
9.1.1	Functions .....	438
9.1.2	Configuration and Operation.....	439
9.1.3	Registers .....	440
9.2	Registers .....	440
9.2.1	Physical Address Boundary Registers 0 and 1 (PABR0 and PABR1).....	440
9.2.2	Wait Control Registers L, M, and H (WCRL, WCRM, and WCRH) .....	445
9.2.3	I/O Wait Control Register (IOWCR).....	447
9.2.4	Interrupt Wait Control Register (INTWR) .....	449
9.2.5	Refresh Wait Control Register (RWCR) .....	451
9.3	Operation.....	452
9.3.1	Wait State Insertion Using $\overline{\text{WAIT}}$ Line Control .....	452
9.3.2	Wait State Insertion Using Register Control .....	454
9.3.3	Wait State Controls.....	456

9.4	Operation in Low Power Dissipation Mode.....	456
9.5	Reset Operation.....	456
9.6	Precautions .....	457
Section 10. Chip Select Control.....		458
10.1	Chip Select Line Operation.....	458
10.2	Operation in Low Power Dissipation Mode.....	459
10.3	Reset Operation.....	459
10.4	Precautions .....	459
Section 11. Low Power Dissipation Modes.....		460
11.1	Sleep Mode.....	460
11.2	System Stop Mode .....	460
Section 12. Oscillator Circuit .....		461
12.1	Crystal Resonator and Oscillator Circuit .....	461
12.2	Oscillator Circuit Board Design.....	462
12.3	Operation Using an External Clock .....	464
Section 13. Electrical Specifications .....		466
13.1	Absolute Maximum Ratings .....	466
13.2	DC Characteristics.....	466
13.3	AC Characteristics.....	467
13.3.1	Bus Timing .....	467
13.3.2	MSCI Timing.....	471
13.3.3	ASCI/CSIO Timing .....	474
13.3.4	DMAC Timing .....	476
13.3.5	Timer Timing.....	476
13.3.6	EXTAL Input Clock Signal Timing .....	477
13.3.7	Miscellaneous .....	477
13.4	Timing Diagrams.....	478
13.4.1	Bus Timing .....	478
13.4.2	MSCI Timing.....	481
13.4.3	ASCI/CSIO Timing .....	484
13.4.4	DMAC Timing .....	487
13.4.5	Timer Timing.....	488
13.4.6	EXTAL Input Clock Signal Timing .....	488
13.4.7	Miscellaneous .....	489

Section 14. Package Dimensions .....	490
14.1 Package Dimensions .....	490

## Appendices

A. Instruction Set .....	493
B. Alphabetical List of Instructions .....	510
C. Opcode Maps.....	517
D. Bus Cycle States.....	521
E-1. Requests in Each Operating Mode .....	541
E-2. Request Priorities .....	545
E-3. State Transition Diagrams .....	546
F-1. Status Signals .....	547
F-2. Pin States in Reset and Low Power Dissipation Modes.....	549
G. Built-in Registers.....	551



# Figures

Figure No.	Description .....	Page
1-1.	Block Diagram of the HD64180S .....	3
1-2.	Allocating CPU Processing Capability .....	4
1-3.	Example Configured with a Data Communications Subsystem .....	5
1-4.	Example of Data Communications Subsystem (minimum configuration) .....	6
1-5.	Example of Data Communications Subsystem (extended configuration) .....	6
1-6.	The HD64180S in a Distributed Control System .....	7
1-7.	Internal Configuration of a Distributed Control Device Using the HD64180S .....	8
2-1(a).	Pin Assignments (CP-84) .....	9
2-1(b).	Pin Assignments (FP-80A) .....	9
2-2.	Example of Crystal Resonator Connection .....	11
2-3.	Example of External Clock Connection .....	11
3-1.	Configuration of CPU Internal Registers .....	21
3-2.	When JP NZ, 6000H Instruction is at 5000H .....	29
3-3.	Conditional Branch Execution Timing .....	29
3-4.	I/O Space Configuration .....	30
3-5.	Opcode Fetch Timing .....	33
3-6.	Opcode Fetch Timing with Wait States .....	34
3-7.	Memory Read/Write Timing .....	36
3-8.	Memory Read/Write Timing with Tw States .....	37
3-9.	External I/O Read/Write Timing with Wait States .....	38
3-10.	Basic Instruction Execution Timing Example (LD (IX + d), g instruction) .....	39
3-11.	Example of Program for Clearing the $\overline{\text{LIRTE}}$ Bit .....	42
3-12.	Timing When 0 is Written to $\overline{\text{LIRTE}}$ Bit (with LIRE=0) .....	43
3-13 (a).	I/O Read and Write Cycle Timing for $\overline{\text{IOC}}=0$ .....	44
3-13 (b).	I/O Read and Write Cycle Timing for $\overline{\text{IOC}}=1$ .....	45
3-14.	RETI Instruction Timing .....	46
3-15.	Operation Mode Transitions .....	49
3-16.	Reset Mode Timing .....	51
3-17.	Timing for Entering Halt Mode and Leaving after an Interrupt .....	54
3-18.	Sleep and System Stop Mode Timing .....	57
3-20.	Bus Arbiter and Masters .....	59

3-21 (a).	Bus Release Mode (1).....	63
3-21 (b).	Bus Release Mode (2).....	64
3-22.	Bus Control Passing.....	65
3-23.	Interrupt Block Diagram.....	66
3-24 (a).	TRAP Cycle (Second Opcode is Undefined).....	81
3-24 (b).	TRAP Cycle (Third Opcode is Undefined).....	82
3-25.	$\overline{\text{NMI}}$ Processing Flowchart.....	83
3-26.	$\overline{\text{NMI}}$ Processing Timing.....	85
3-27.	Timing of $\overline{\text{INT0}}$ Mode 0 Interrupt (with an RST instruction on the data bus).....	87
3-28.	Flow of Interrupt Processing in $\overline{\text{INT0}}$ Mode 1.....	88
3-29.	Timing of $\overline{\text{INT0}}$ Mode 1 Interrupt.....	89
3-30.	Start Address Generation in $\overline{\text{INT0}}$ Mode 2.....	90
3-31.	Timing of $\overline{\text{INT0}}$ Mode 2 Interrupt.....	91
3-32.	Circuit of $\overline{\text{INT1}}$ , $\overline{\text{INT2}}$ , and Internal Interrupts Except TRAP.....	93
3-33.	Start Address Generation for $\overline{\text{INT1}}$ , $\overline{\text{INT2}}$ , and Internal Interrupts Except TRAP.....	95
3-34.	Timing for $\overline{\text{INT1}}$ , $\overline{\text{INT2}}$ , and Internal Interrupts Except TRAP.....	96
3-35.	MMU Block Diagram.....	98
3-36.	Relationship Between Physical and Logical Addresser for I/O Accesses.....	101
3-37.	Example of Logical Address Space Division.....	102
3-38.	Physical Address Generation.....	103
3-39.	Relationship Between Logical and Physical Spaces.....	104
4-1.	MSCI Block Diagram.....	108
4-2.	Block Diagram of the MSCI Receiver.....	111
4-3.	Block Diagram of the MSCI Transmitter.....	113
4-4 (a).	Modem Control Signal Timing (auto-enable, 5 bits/character, no parity and 1/1 clock mode).....	118
4-4 (b).	Modem Control Signal Timing.....	118
4-5.	Residue Bit Frame Reception Operation.....	155
4-6.	Abort End Frame Reception Operation.....	156
4-7.	MSCI Frame Status Register.....	161
4-8.	Interrupt Conditions.....	163
4-9.	Character Format for Asynchronous Mode.....	177
4-10.	Bit Rate Selection.....	178
4-11.	Timing of Data Sampling (1/1 clock mode).....	178
4-12.	State Transition Diagram for Asynchronous Mode Transmission.....	180
4-13 (a).	Transmit Operation in 1/1 Clock Mode.....	181

4-13 (b).	Transmit Operation in 1/16, 1/32, or 1/64 Clock Mode.....	181
4-14.	State Transition Diagram for Asynchronous Mode Reception.....	183
4-15 (a).	Receive Data Sampling Timing (1/1 Clock Mode) .....	184
4-15 (b).	Receive Data Sampling Timing (1/16, 1/32, or 1/64 Clock Mode).....	184
4-16.	Character Assembly in the Receiver Shift Register.....	185
4-17 (a).	Start Bit Sampling (normal start bit is detected).....	186
4-17 (b).	Start Bit Sampling (false start bit (noise) is detected) .....	186
4-18.	Noise Suppressor Function .....	187
4-19.	Receive Character Format.....	188
4-20.	Break Detection by the Receiver .....	190
4-21.	Sample MP Bit Operation .....	192
4-22.	Character Format for Byte Synchronous Mode .....	193
4-23.	State Transition Diagram for Byte Synchronous Transmission.....	195
4-24.	State Transition Diagram for Byte Synchronous Reception .....	197
4-25.	Message Format for Bit Synchronous Mode .....	200
4-26.	State Transition Diagram for Transmission in Bit Synchronous HDLC Mode .....	202
4-27.	State Transition Diagram for Bit Synchronous Reception.....	204
4-28.	Relationship between Primary and Secondary Stations in Bit Synchronous Loop Mode.....	209
4-29.	GA Pattern.....	210
4-30.	GA Pattern Generation at the End of Data Transmission in the Secondary Station .....	211
4-31 (a).	Normal Receive Format .....	212
4-31 (b).	Abort End Frame Format .....	212
4-32.	State Transition Diagram for Transmission in Bit Synchronous Loop Mode.....	214
4-33.	Selecting Transmit and Receive Clocks.....	216
4-34.	Transmit Clock Sources .....	217
4-35 (a).	Receive Clock Source (Receive BRG output or RXCM line input used as receive clock) .....	218
4-35 (b).	Receive Clock Source (Clock extracted by ADPLL used as receive clock).....	218
4-35 (c).	Receive Clock Source (Receive clock noise suppressed).....	219
4-36.	ADPLL Block Diagram .....	220
4-37.	Transmission Codes Supported by the ADPLL and Their Waveforms .....	223
4-38.	Data Flow and Clocks when Extracting Clock from Receive Data.....	224
4-39.	NRZ Receive Data Phase Compensation in Operating Mode $\times 8$ .....	225

4-40.	FM0 Receive Data Phase Compensation in Operating Mode $\times 8$ . . . . .	226
4-41.	Noise Suppression in the Receive Data Noise Suppressor in Operating Mode $\times 8$ . . . . .	226
4-42.	Data and Clock Signal Flow When Suppressing Receive Clock Noise. . . . .	227
4-43.	Noise Suppression in the Receive Clock Noise Suppressor. . . . .	228
4-44.	The Enter Search Mode Command Timing between Continuous Frames . . . . .	229
4-45.	Block Diagram of the Baud Rate Generator. . . . .	232
4-46.	Reload Timer Output . . . . .	233
4-47.	Logic Flow for Interrupt Requests, Status and Enable Bits . . . . .	245
4-48.	Transmission/Reception Between HD64180S Chips . . . . .	248
4-49.	Input Clock and Transmit Data. . . . .	248
5-1.	ASCI/CSIO Block Diagram . . . . .	262
5-2 (a).	Modem Control Signal Timing (Auto-enable, 7 bits/character, no parity, 1/1 clock mode).....	266
5-2 (b).	Modem Control Signal Timing.....	266
5-3.	Character Format in Asynchronous Mode.....	300
5-4.	Bit Rate Selection.....	301
5-5.	State Transition Diagram for Asynchronous Mode Transmission.....	302
5-6 (a).	Transmit Operation Using 1/1 Clock.....	303
5-6 (b).	Transmit Operation Using 1/16, 1/32, or 1/64 Clock.....	303
5-7.	State Transition Diagram for Asynchronous Mode Reception.....	305
5-8 (a).	Receive Data Sampling Using 1/1 Clock.....	306
5-8 (b).	Receive Data Sampling Using 1/16, 1/32, or 1/64 Clock.....	306
5-9.	Receive Character Assembly by Shift Register.....	307
5-10-(a).	Start Bit Sampling (normal start bit).....	307
5-10 (b).	Start Bit Sampling (false start bit).....	308
5-11.	Noise Suppressor Function.....	309
5-12.	Receive Character.....	309
5-13.	Break Detection.....	311
5-14.	Sample MP Bit Operation.....	312
5-15.	Communications Character Format for Clocked Serial Mode.....	314
5-16.	Sample Transmissions in Clocked Serial Mode.....	316
5-17.	Selecting Transmit and Receive Clocks.....	318
5-18.	BRG Block Diagram.....	320
5-19.	Reload Timer Output.....	321
5-20.	Relationship Between Interrupts, Status, and Enable Bits.....	332
5-21.	Input Clock and Transmit Data.....	334
6-1.	DMAC Block Diagram (one channel).....	339
6-2.	Software Abort and DMAC Operation.....	363

6-3.	DME Bit State Transitions .....	366
6-4.	DMA Enable Logic .....	366
6-5.	Effect of an $\overline{\text{NMI}}$ on DMAC Operation .....	367
6-6.	Descriptors and Buffers for Chained-block Transfers from Memory to the MSCI .....	368
6-7.	Descriptor Format .....	369
6-8.	Descriptors and Buffers for Chained-block Transfers from the MSCI to Memory .....	371
6-9.	Single-block Transfers Between Memory and Memory (Dual Address) .....	374
6-10.	Memory-to Memory Single-block Transfer (Dual Address) (Cycle Steal Mode).....	376
6-11.	Single-block Transfers between Memory and I/O (Memory-mapped I/O) Using Dual Address Type .....	377
6-12.	CPU and DMA Operation When the $\overline{\text{DREQ}}$ Line is Level Sensitive.....	378
6-13.	CPU and DMA Operation When the $\overline{\text{DRER}}$ Line is Edge Sensitive.....	379
6-14 (a).	Memory-to-I/O Single-block Transfer Mode (Dual Address) .....	380
6-14 (b).	Single-block Transfer between Memory and Memory-mapped I/O (Dual Address).....	381
6-15.	$\overline{\text{TEND}}$ Output Timing .....	382
6-16.	Single-block Transfers between Memory and MSCI (Single Address).....	383
6-17 (a).	Memory-to-MSCI Single-block Transfer Mode (Single Address) .....	385
6-17 (b).	MSCI-to Memory Single-block Transfer Mode (Single Address) .....	386
6-18.	Chained-block Transfer from Memory to the MSCI .....	388
6-19.	Chained-block Transfer from Memory to the MSCI .....	390
6-20.	Memory-to-MSCI Chained-block Transfer Timing (for the Start of Transfer and Buffer Switching).....	399
6-21.	Chained-block Transfer from the MSCI to Memory .....	401
6-22.	Chained-block Transfer from the MSCI to Memory .....	403
6-23.	MSCI-to Memory Chained-block Transfer Timing (for Transfer Start and Buffer Switching).....	412
6-24.	Connections Between the DMAC and MSCI .....	414
6-25.	Relationship between Interrupt Status Bits and Enable Bits.....	417
7-1.	Timer Block Diagram .....	420
7-2.	Timer Count-up Timing (Example 1) .....	427
7-3.	Timer Count-up Timing (Example 2) .....	427
7-4.	Timer Count-up Timing (Example 3) .....	428
7-5.	Timer Output Timing .....	429

7-6.	Internal Interrupt Circuit .....	429
7-7.	Internal Interrupt Timing (when the counter operating rate is BC and internally synchronized) .....	430
8-1.	Refresh Controller Block Diagram .....	433
8-2.	Refresh Timing .....	436
8-3.	Refresh Requests in Bus Release Mode.....	437
9-1.	Wait Controller Block Diagram.....	439
9-2.	Memory Space Partitioned by PABR0 and PABR1 .....	442
9-3 (a).	Setting Example when the Physical Address Space is not Partitioned .....	443
9-3 (b).	Setting Example when the Physical Address Space is Partitioned into PAM and PAL.....	443
9-3 (c).	Setting Example when the Physical Address Space is Partitioned into PAH and PAL.....	444
9-3 (d).	Setting Example when the Physical Address Space is Partitioned into PAH, PAM and PAL.....	444
9-4.	Example of Incorrect Boundary Specification.....	445
9-5.	Internal/External I/O Space Partition .....	448
9-6.	Wait State Insertion Timing in an $\overline{INT0}$ Acknowledge Cycle .....	451
9-7.	Wait State Insertion Timing Using $\overline{WAIT}$ Line Control .....	453
9-8.	Memory Space Division and Wait State Insertion.....	454
9-9.	I/O Space Division and Wait State Insertion .....	455
10-1.	Chip Select Timing .....	458
12-1.	$\emptyset$ Clock Generation and Supplies to Each Function Block .....	461
12-2.	Crystal Connection Circuit.....	462
12-3 (a).	Undesirable Oscillator Circuit Layout (1) .....	463
12-3 (b).	Undesirable Oscillator Circuit Layout (2) .....	463
12-4.	Example of Oscillator Circuit Layout.....	464
12-5.	Example of Connection for External Clock.....	465
13-1.	Bus Timing (1).....	478
13-2.	Bus Timing (2).....	479
13-3.	Bus Timing (3) (sleep or system stop mode).....	480
13-4.	Bus Timing (4).....	480
13-5.	MSCI Transmit Timing (TXCM input) .....	481
13-6.	MSCI Transmit Timing (TXCM output) .....	481
13-7.	MSCI Receive Timing (RXCM input).....	481
13-8.	MSCI Receive Timing (RXCM output).....	482
13-9.	MSCI ADPLL Operating Clock Timing.....	482
13-10.	MSCI Band Rate Generator Output Timing ( $f_{BRG} \neq f_{\emptyset}$ ).....	482

13-11.	MSCI <u>SYNC</u> Timing .....	483
13-12.	MSCI <u>CTSM</u> and <u>DCDM</u> Timing .....	483
13-13.	MSCI <u>RTSM</u> Timing .....	483
13-14.	ASCI/CSIO Transmit Timing (TXCA input) .....	484
13-15.	ASCI/CSIO Transmit Timing (TXCA output) .....	484
13-16.	ASCI/CSIO Receive Timing (RXCA input).....	484
13-17.	ASCI/CSIO Receive Timing (RXCA output).....	485
13-18.	ASCI/CSIO <u>Baud Rate Generator</u> Timing.....	485
13-19.	ASCI/CSIO <u>CTSA</u> and <u>DCDA</u> Timing .....	485
13-20.	ASCI/CSIO <u>RTSA</u> Timing .....	486
13-21.	DMAC Timing .....	487
13-22.	Timer Timing .....	488
13-23.	EXTAL Input Clock Signal Timing.....	488
13-24.	Rise and Fall Times of Input Signals with No Characteristics Specified .....	489
13-25.	Reference Levels .....	489
13-26.	Bus Timing Load .....	489
14-1.	Package Dimensions .....	490



# Tables

Table No.	Description .....	Page
1-1.	Major Functions of the HD64180S .....	1
3-1.	Register for Interfacing with Z80-based Peripherals .....	32
3-2.	OMCR Setting.....	41
3-3.	Bus Cycle States for RETI Instruction Execution .....	47
3-4.	The Status of the Functional Blocks in the Various Operation Modes.....	49
3-5.	Low Power Dissipation Mode Specification Register .....	50
3-6.	Interrupt Types, Priorities, and Sources.....	67
3-7.	Interrupt Control Registers.....	68
3-8.	IEF <sub>1</sub> and IEF <sub>2</sub> Values.....	79
3-9.	Interrupt Sources and Vectors.....	94
3-10.	Initial Values of Flags and Registers Associated with Interrupts .....	97
3-11.	MMU Registers.....	98
4-1.	MSCI Registers .....	114
4-2.	Transmit Commands .....	138
4-3.	Receive Commands.....	139
4-4.	Other Commands .....	141
4-5.	SYN Pattern Length in Byte Synchronous Mode .....	193
4-6.	Address Field Check .....	207
4-7.	Reactions to Short Frame Detection .....	207
4-8.	4-bit Address Mode Field Check Function .....	215
4-9.	Relationship Between the ADPLL Operating Clock and Bit Rates.....	221
4-10.	ADPLL Specifications .....	222
4-11.	Maximum of Level Transitions .....	230
4-12.	Phase Adjustment of the ADPLL .....	230
4-13.	BGR Output Waveform and Register Set Values .....	234
4-14.	Register Values and Bit Rates in Asynchronous Mode .....	235
4-15.	Register Set Values and Bit Rates (Byte/Bit Synchronous Mode) .....	239
4-16.	Internal Interrupts, Their Sources and Clearing Methods .....	242
4-17.	Maximum Bit Rates .....	247
5-1.	ASCII Registers .....	263
5-2.	Transmit Commands .....	279
5-3.	Receive Commands.....	280
5-4.	Other Commands .....	280
5-5.	Availability of Master/Slave Combinations.....	314
5-6.	Transmission/Reception Operation in Clocked Serial Mode.....	317
5-7.	BRG Output Waveform and Register Set Values.....	322
5-8.	Register Values and Corresponding Bit Rates in Asynchronous Mode .....	323

5-9.	Register Set Values and Bit Rates (clocked serial mode).....	327
5-10.	Internal Interrupts, Sources, and Clear Procedures.....	330
5-11.	Expressions to Calculate Maximum Bit Rates.....	334
6-1.	Registers.....	341
6-2.	Registers Shared by Channels 0 and 1.....	342
6-3.	Status Configuration (transmit).....	370
6-4.	Status Configuration (receive).....	372
6-5.	DMAC Operating Modes.....	373
6-6.	Control Registers Used for Chained-block Transfer from Memory to the MSCI (Transmit).....	391
6-7.	Chained-block Transfer from Memory to MSCI (single frame transfer).....	393
6-8.	Chained-block Transfer from Memory-to-MSCI (multi-frame transfer).....	396
6-9.	Control Registers Used for Chained-block Transfers from the MSCI to Memory (Receive).....	404
6-10.	Chained-block Transfer from MSCI to Memory (multi-frame transfer) (Normal receive operation).....	407
6-11.	Chained-block transfer from MSCI to Memory (multi-frame transfer) (Releasing part of a buffer during a receive operation).....	409
6-12.	Characteristics.....	413
6-13.	Lines between DMAC Channel 0 and MSCI Receiver.....	414
6-14.	Lines between DMAC Channel 1 and MSCI Transmitter.....	415
6-15.	Internal Interrupts.....	415
7-1.	Timer Registers.....	421
8-1.	Refresh Controller Register.....	433
8-2.	Refresh Cycles and their Intervals.....	435
9-1.	Registers.....	440
9-2.	Wait State Insertion Conditions.....	456
10-1.	$\overline{CS}$ Lines and Associated Physical Address Spaces.....	456
12-1.	Recommended Characteristics of Crystal Resonator and Load Capacitor.....	462
13-1.	Absolute Maximum Ratings.....	466
13-2.	DC Characteristics.....	466
13-3.	Bus Timing.....	467
13-4.	MSCI Timing.....	471
13-5.	ASCI/CSIO Timing.....	474
13-6.	DMAC Timing.....	476
13-7.	Timer Timing.....	476
13-8.	EXTAL Input Clock Signal Timing.....	477
13-9.	Rise and Fall Times of Input Signals with No Characteristics Specified.....	477

# Section 1. Overview

## 1.1 Overview

The HD64180S network processing unit (NPU) contains a 2-channel serial interface, 8-bit CPU, 2-channel direct memory access controller (DMAC) with a proprietary chained-block transfer function, timers, etc., all integrated on a single LSI chip. The HD64180S is thus well suited to multiprotocol communications processing.

The multiprotocol serial communications interface (MSCI) and the asynchronous serial communications interface/clocked serial I/O port (ASCI/CSIO) allow high speed data transfer using various communications protocols.

In particular, the MSCI is capable of handling asynchronous, byte synchronous, and bit synchronous communications protocols. Since the MSCI is connected to the on-chip DMAC, it is possible to realize high speed single-address DMA transfer (chained-block transfer) in frame units during bit synchronous communications. Furthermore, the flexible processing capability of the HD64180S's CPU ensures compatibility with a wide range of communications protocols.

Table 1-1 lists the major functions of the HD64180S and figure 1-1 shows the block diagram.

**Table 1-1. Major Functions of the HD64180S**

<b>Item</b>	<b>Specifications</b>
CPU	<ul style="list-style-type: none"><li>• Software-compatible with HD64180Z</li><li>• 80 type bus interface</li><li>• On-chip MMU (1 Mbyte physical address space)</li></ul>
DMAC	<ul style="list-style-type: none"><li>• 2 channels</li><li>• DMA transfer between memory and memory, memory and I/O (memory-mapped I/O), and memory and MSCI</li><li>• Chained-block transfer between memory and MSCI</li><li>• Internal interrupt requests available</li></ul>

**Table 1-1. Major Functions of the HD64180S (cont.)**

<b>Item</b>	<b>Specifications</b>
Multiprotocol serial communications interface (MSCI)	<ul style="list-style-type: none"> <li>• Full duplex channel</li> <li>• Asynchronous, byte synchronous (mono-, bi-, or external synchronous), or bit synchronous (HDLC or loop) selectable</li> <li>• Transmit/receive control using modem control signals (<math>\overline{\text{RTSM}}</math>, <math>\overline{\text{CTS}}</math>, and <math>\overline{\text{DCDM}}</math>)</li> <li>• Internal Advanced Digital PLL (ADPLL) clock extraction receive data and/or receive clock noise suppression</li> <li>• On-chip baud rate generator</li> <li>• Internal interrupt requests available</li> <li>• Maximum transfer rate 7.1 Mbps (with 10 MHz clock)</li> </ul>
Asynchronous serial communications interface/clocked serial I/O port (ASCI/CSIO)	<ul style="list-style-type: none"> <li>• Full duplex channel</li> <li>• Asynchronous or clocked serial mode (selectable)</li> <li>• Transmit/receive control using modem control signals (<math>\overline{\text{RTSA}}</math>, <math>\overline{\text{CTSA}}</math>, and <math>\overline{\text{DCDA}}</math>)</li> <li>• On-chip baud rate generator</li> <li>• Internal interrupt requests available</li> </ul>
Timers	<ul style="list-style-type: none"> <li>• 2 channels</li> <li>• 8-bit reloadable up-counter</li> <li>• Output waveform generator and external event count functions</li> <li>• Internal interrupt requests available</li> </ul>
Interrupt controller	<ul style="list-style-type: none"> <li>• Four external interrupt lines (<math>\overline{\text{NMI}}</math>, <math>\overline{\text{INT0}}</math>, <math>\overline{\text{INT1}}</math>, and <math>\overline{\text{INT2}}</math>)</li> <li>• Fifteen internal interrupt sources</li> </ul>
Memory access support function	<ul style="list-style-type: none"> <li>• Internal refresh controller</li> <li>• Internal wait state controller</li> <li>• Internal chip-select controller</li> </ul>
Other functions	<ul style="list-style-type: none"> <li>• On-chip clock oscillator circuit</li> <li>• Low power dissipation modes (sleep and system stop)</li> </ul>

**Type of Products**

<b>Product Name</b>	<b>Max. Operating Frequency</b>	<b>Package</b>
HD64180SCP6	6.17 MHz	CP-84 (84-pin PLCC)
HD64180SCP8	8 MHz	
HD64180SCP10	10 MHz	
HD64180SH6	6.17 MHz	FP-80A (80-pin QFP)
HD64180SH8	8 MHz	
HD64180SH10	10 MHz	

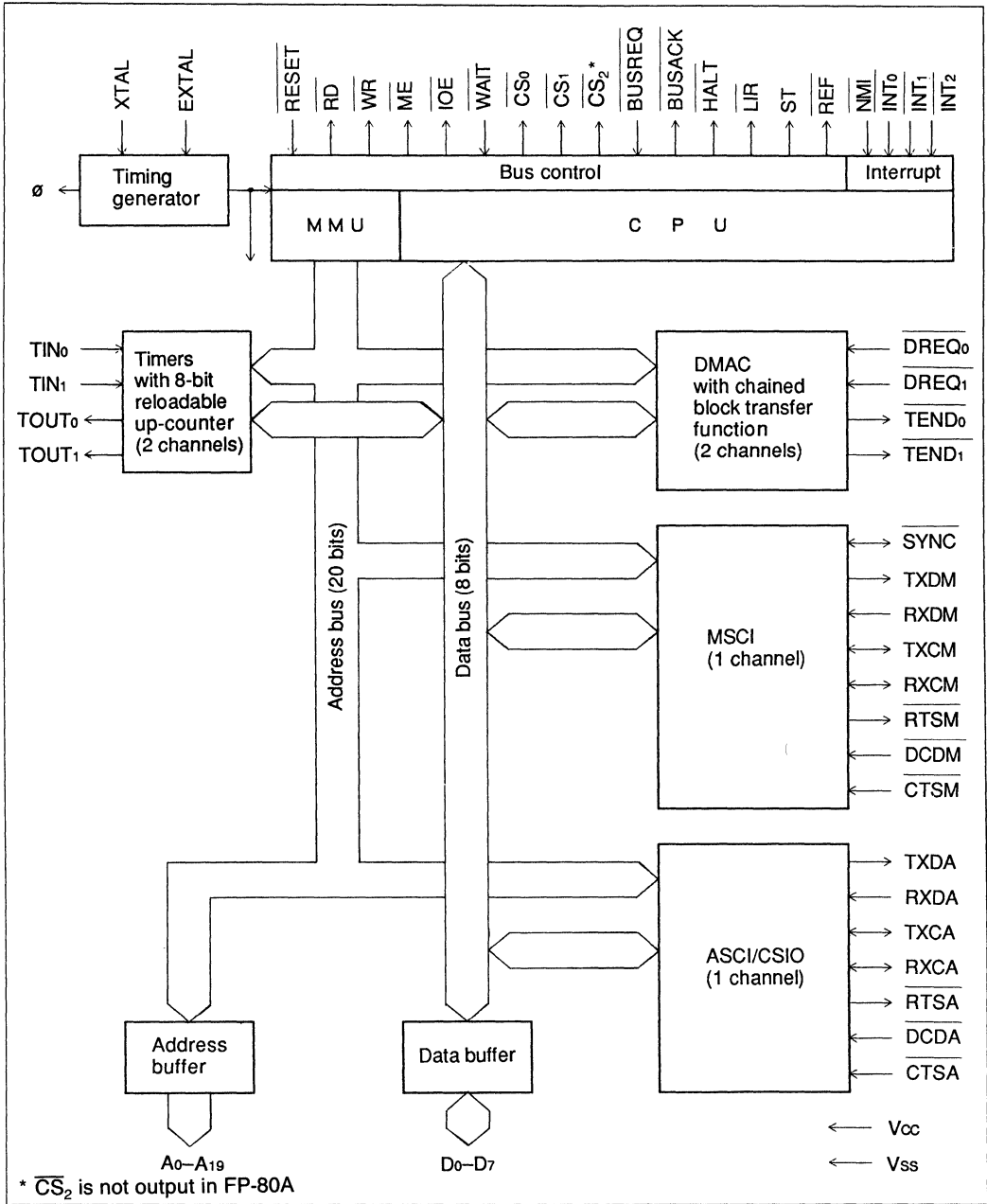
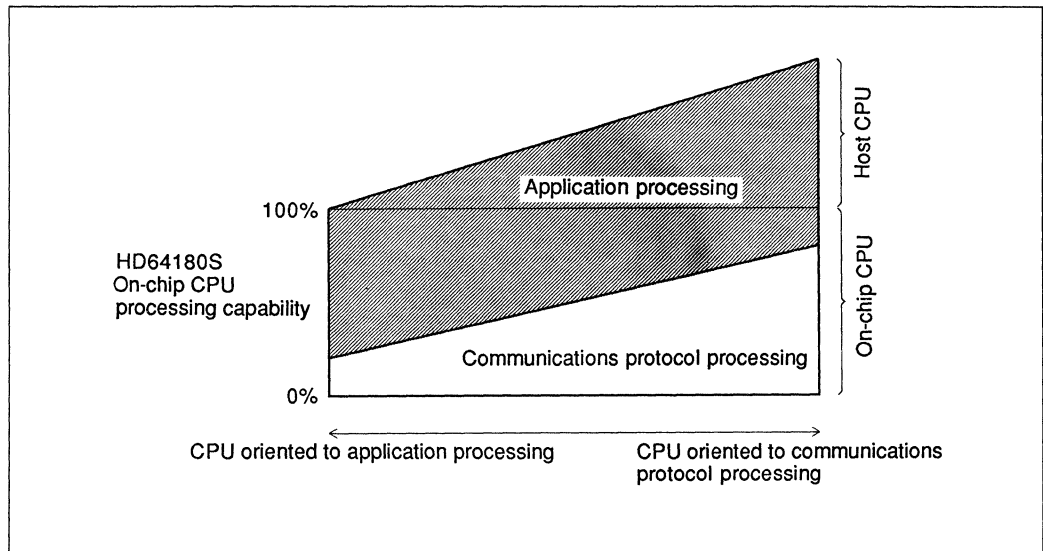


Figure 1-1. Block Diagram of the HD64180S

## 1.2 Applications

### 1.2.1 Position in Product Line

The HD64180S's on-chip CPU (software-compatible with the HD64180Z) is capable of processing both communications protocols and user application programs. If the on-chip CPU is programmed for use mainly as a communications processor, application processing can be carried out by another CPU. Figure 1-2 illustrates this concept.



**Figure 1-2. Allocating CPU Processing Capability**

For example, the HD64180S's CPU can be used mainly for communications protocol processing to provide various communications functions for a host CPU. This is suitable in situations requiring high-speed data transfer and/or complicated protocol processing. In this case, a flexible interface can be configured with the host CPU by selecting appropriate software and I/O devices.

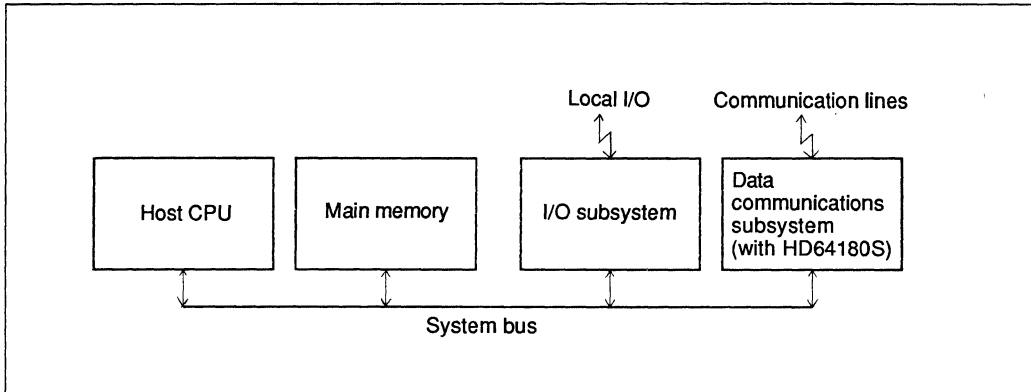
On the other hand, the HD64180S's CPU can be used for application processing (i.e., when data transfer occurs infrequently and/or at low speeds). In this case, the MSCI, ASCI/CSIO, and DMAC in the HD64180S can process the communications data so as to reduce CPU overhead.

Thus the HD64180S can be used in a wide range of applications—from small-scale configurations containing two or three chips to large-scale configurations containing mass memory and numerous I/O devices.

## 1.2.2 Examples of System Configuration

### (1) Data communications system

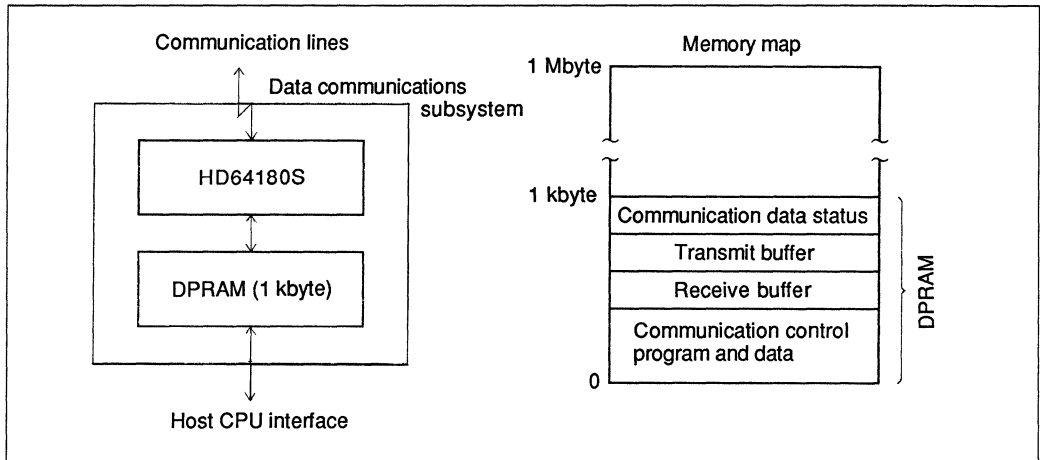
Figure 1-3 shows a system configured with a data communications subsystem. This system can be used for communications between computers in a public network or in an office automation (OA) system.



**Figure 1-3. Example Configured with a Data Communications Subsystem**

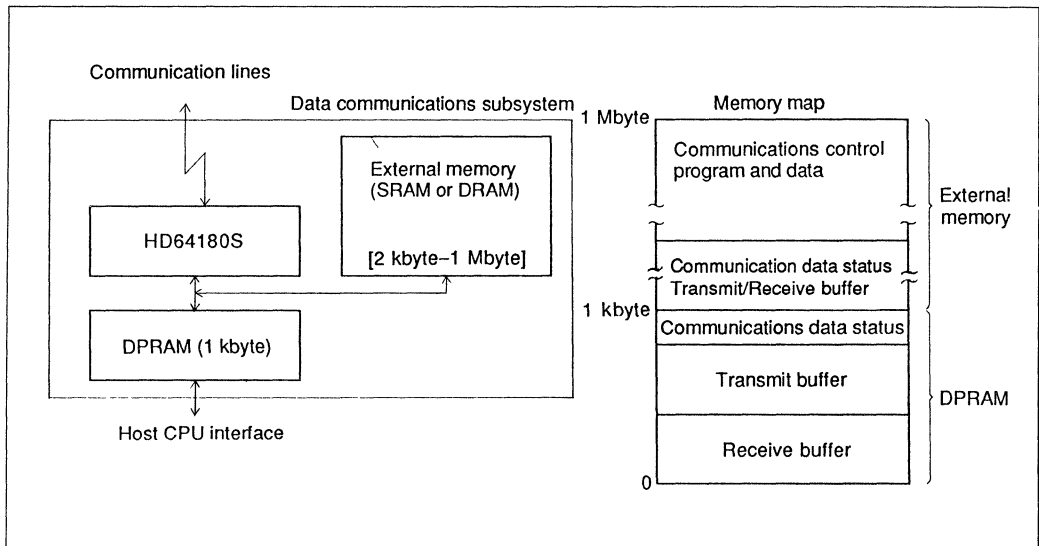
Figure 1-4 shows a minimum configuration example for the data communications subsystem shown in figure 1-3. In this configuration, the host CPU loads the HD64180S control program from main memory into the dual port RAM (DPRAM). The DPRAM has a transmit buffer, receive buffer, and communications data status area for interfacing between the host CPU and the HD64180S. Since the memory area allocated to this subsystem's communications program and transmit/receive buffers is relatively small, the subsystem is well suited for low-speed, simple communications protocols.





**Figure 1-4. Example of Data Communications Subsystem (minimum configuration)**

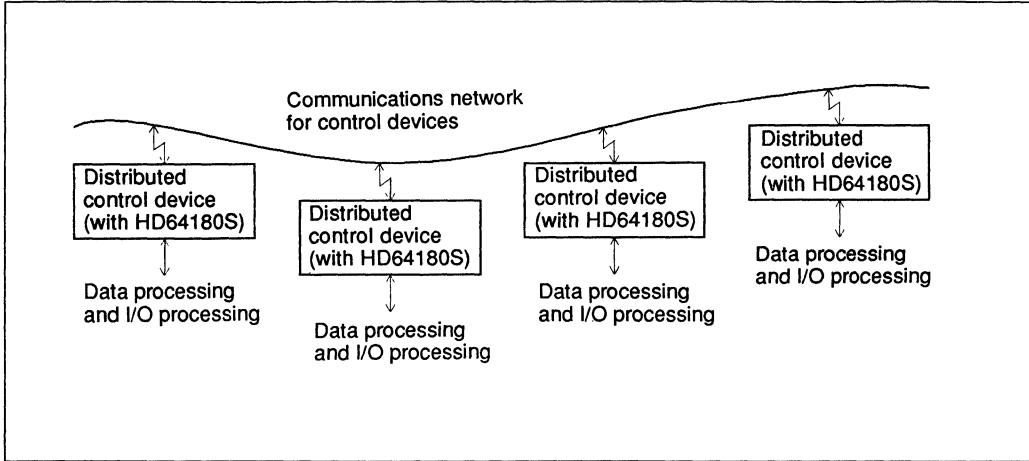
Figure 1-5 shows an extended communications subsystem for complex protocol processing and high-speed data transfer. This subsystem incorporates external memory and two stages of transmit/receive buffers. The HD64180S control program is loaded into external memory. This subsystem is easily realized because the HD64180S can directly access up to 1 Mbyte of memory using its 20-bit address bus.



**Figure 1-5. Example of Data Communications Subsystem (extended configuration)**

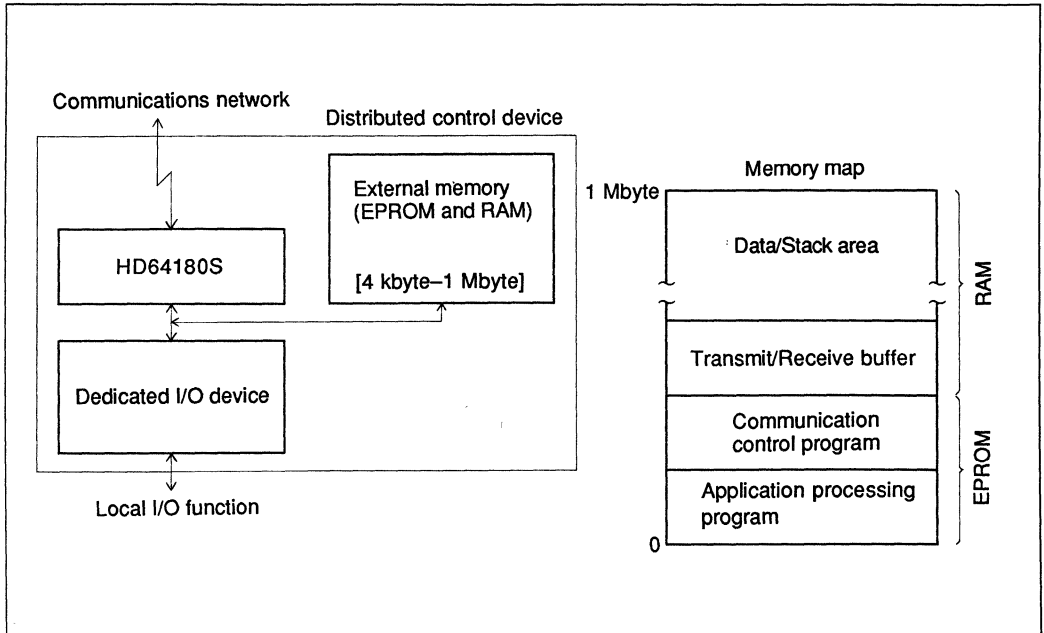
## (2) Distributed control system

Figure 1-6 shows an example in which the HD64180S is used as a distributed control device. This configuration can be used for controlling industrial machinery or for communicating between control devices of automobiles, OA systems, point-of-sales (POS) terminals, etc.



**Figure 1-6. The HD64180S in a Distributed Control System**

Figure 1-7 shows the internal configuration of the distributed control devices shown in figure 1-6. In this configuration, the HD64180S is directly connected to an I/O device, and the external memory (EPROM and RAM) contains the HD64180S control program and application programs. This simple system also allows high-speed data processing by providing direct access to up to 1 Mbyte of memory space including the data/stack and transmit/receive buffer areas.



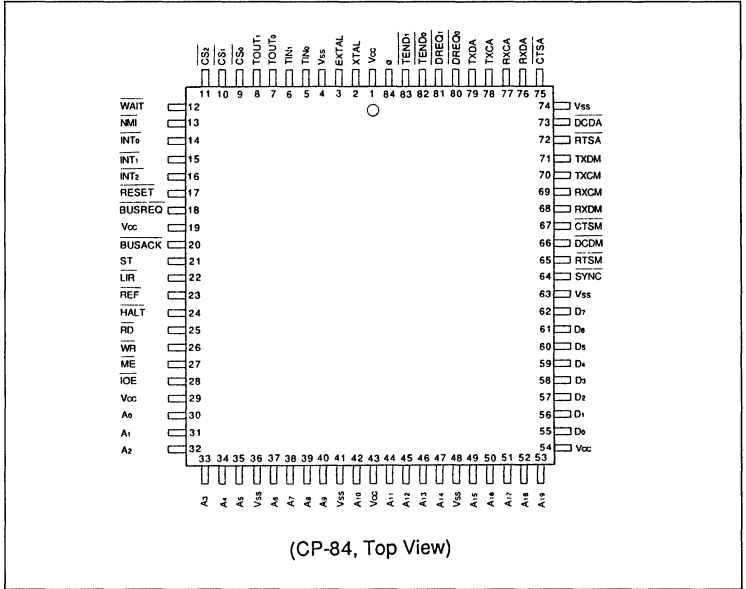
**Figure 1-7. Internal Configuration of a Distributed Control Device Using the HD64180S**

In the two configuration examples given above, the HD64180S is used either as a part of a data communications subsystem or as a distributed control device. In addition, the HD64180S can be used with various kinds of communications equipment.

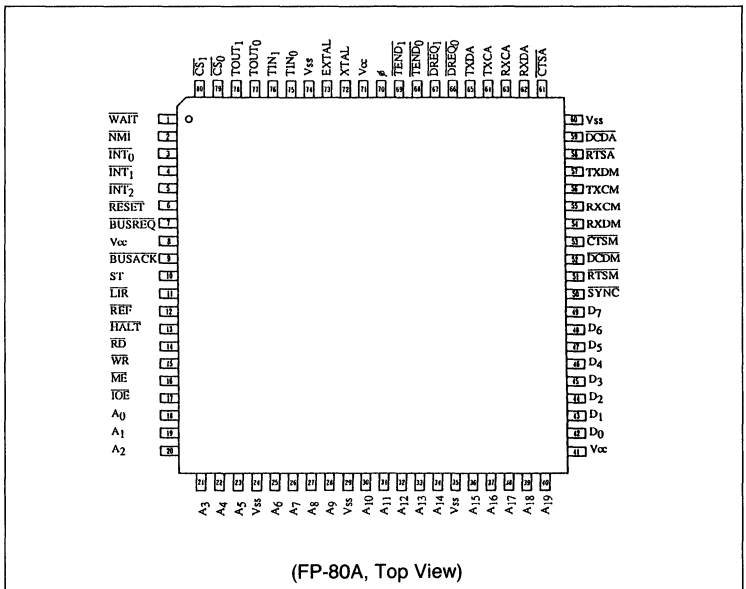
# Section 2. Pin Assignments and Signal Descriptions

## 2.1 Pin Assignments

Figure 2-1 shows the pin assignments for the HD64180S.



**Figure 2-1(a). Pin Assignments (CP-84)**



**Figure 2-1(b). Pin Assignments (FP-80A)**

## 2.2 Signal Descriptions

### 2.2.1 Power Supply

Symbol	Pin Number		Input/ Output	Remarks
	CP-84	FP-80A		
V <sub>CC</sub>	1, 19, 29, 43, 54	8, 41, 71	Input	+5V power supply: All V <sub>CC</sub> pins must be connected to the +5V system power supply.
V <sub>SS</sub>	4, 36, 41, 48, 63, 74	24, 29, 35, 60, 74	Input	Ground: All V <sub>SS</sub> pins must be connected to the system ground.

Note: To minimize potential difference in the chip, use the shortest possible lead length to the V<sub>CC</sub> and V<sub>SS</sub> pins.

### 2.2.2 Clock

Symbol	Pin Number		Input/ Output	Remarks
	CP-84	FP-80A		
XTAL	2	72	Input	Crystal resonator input: The input frequency must be double that of the $\phi$ clock. When the EXTAL pin is connected to an external clock, the XTAL pin should be left floating.
EXTAL	3	73	Input	Crystal resonator or external clock input: The input frequency must be double that of the $\phi$ clock. Figures 2-2 and 2-3 show crystal resonator and external clock connection diagrams, respectively.
$\phi$	84	70	Output	System clock: Supplies the $\phi$ clock to peripheral devices.

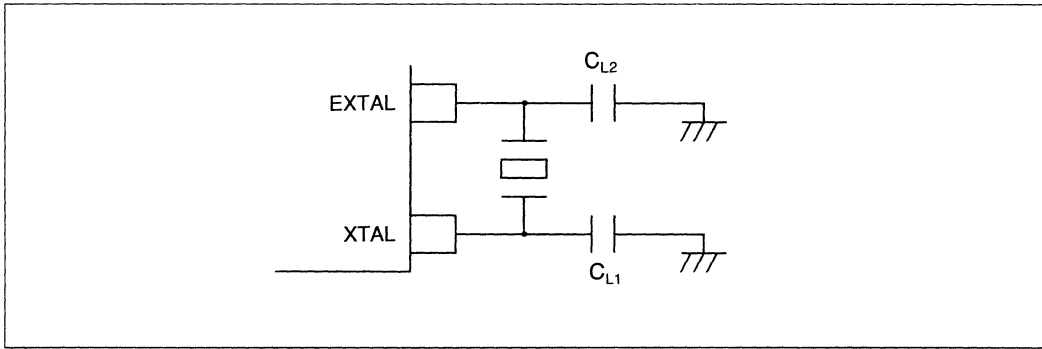


Figure 2-2. Example of Crystal Resonator Connection

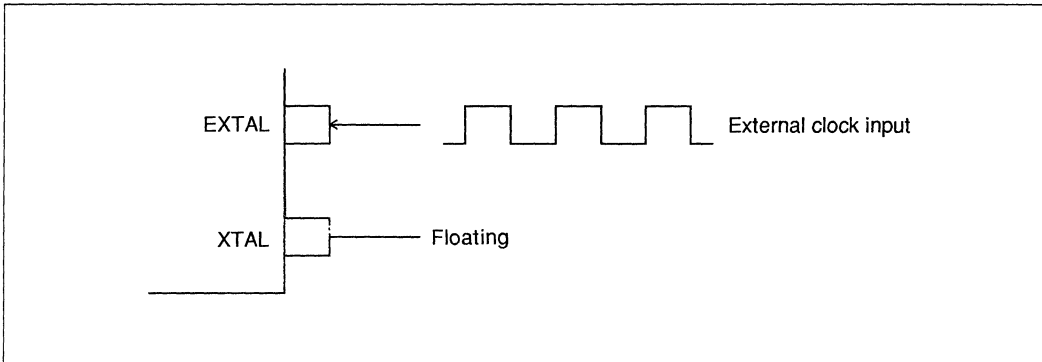


Figure 2-3. Example of External Clock Connection

### 2.2.3 Reset Line

Symbol	Pin		Input/ Output	Remarks
	CP-84	FP-80A		
$\overline{\text{RESET}}$	17	6	Input	Reset: When this line is driven active low for 6 or more clock cycles, the HD64180S enters the reset mode and all functions are reset.

## 2.2.4 Address Lines

Symbol	Pin Number		Input/ Output	Remarks
	CP-84	FP-80A		
A <sub>0</sub> -A <sub>19</sub>	30-35, 37-40, 42, 44-47, 49-53	18-23, 25-28, 30-34, 36-40	Output (Three State)	Address bus: This 20-bit address bus supports 1Mbyte of memory and a 64kbyte (16-bit address width) I/O space. The address bus goes to high impedance during: <ul style="list-style-type: none"> <li>• Reset mode</li> <li>• Passing control of the bus to another device (the HD64180S is placed in the bus release mode when the <math>\overline{\text{BUSREQ}}</math> line is asserted).</li> </ul>

## 2.2.5 Data Lines

Symbol	Pin Number		Input/ Output	Remarks
	CP-84	FP-80A		
D <sub>0</sub> -D <sub>7</sub>	55-62	42-49	Input/ Output (Three State)	Data bus: The 8-bit handles bi-directional data passing (input and output of data.)

## 2.2.6 Memory and I/O Interface Lines

Symbol	Pin Number		Input/ Output	Remarks
	CP-84	FP-80A		
$\overline{\text{RD}}$	25	14	Output (Three State)	Read: This line is asserted during read cycles. When this line is driven active low, the data lines are used as inputs.
$\overline{\text{WR}}$	26	15	Output (Three State)	Write: This line is asserted during write cycles. When this line is driven active low, the data lines are used as outputs.

Symbol	Pin Number		Input/ Output	Remarks												
	CP-84	FP-80A														
$\overline{\text{ME}}$	27	16	Output (Three State)	<p>Memory enable: This line is used to indicate a memory read or write operation. It is asserted in the following cases:</p> <ul style="list-style-type: none"> <li>• Instruction fetch, operand read, and memory read/write instructions</li> <li>• Memory access during DMA cycles</li> <li>• Refresh cycles</li> </ul>												
$\overline{\text{IOE}}$	28	17	Output (Three State)	<p>I/O enable: This line is used to indicate an I/O read/write operation. It is asserted in the following cases:</p> <ul style="list-style-type: none"> <li>• I/O read/write instructions</li> <li>• I/O access during DMA cycles</li> <li>• <math>\overline{\text{INT0}}</math> acknowledge cycles</li> </ul>												
$\overline{\text{WAIT}}$	12	1	Input	<p>Wait: This line is used to extend either memory or I/O read/write cycles. If this line is low at the falling edge of a T2 state, a Tw state is inserted. If the line is still low at the falling edge of the inserted Tw state, an additional Tw state is inserted. This process is repeated until the signal level on this line is high at the falling edge.</p>												
$\overline{\text{CS0}}$	9	79	Output	<p>Chip select: These lines are used to access one of the three physical address areas: PAL, PAM, and PAH. The partition of the physical address space is the same as that of wait controllers.</p> <table border="1"> <thead> <tr> <th></th> <th>Physical address area accessed</th> <th>Signal asserted</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>PAL area (lower physical address area)</td> <td><math>\overline{\text{CS0}}</math></td> </tr> <tr> <td>2</td> <td>PAM area (middle physical address area)</td> <td><math>\overline{\text{CS1}}</math></td> </tr> <tr> <td>3</td> <td>PAH area (upper physical address area)</td> <td><math>\overline{\text{CS2}}</math></td> </tr> </tbody> </table>		Physical address area accessed	Signal asserted	1	PAL area (lower physical address area)	$\overline{\text{CS0}}$	2	PAM area (middle physical address area)	$\overline{\text{CS1}}$	3	PAH area (upper physical address area)	$\overline{\text{CS2}}$
	Physical address area accessed	Signal asserted														
1	PAL area (lower physical address area)	$\overline{\text{CS0}}$														
2	PAM area (middle physical address area)	$\overline{\text{CS1}}$														
3	PAH area (upper physical address area)	$\overline{\text{CS2}}$														
$\overline{\text{CS1}}$	10	80	Output													
$\overline{\text{CS2}}$	11	—	Output													



## 2.2.7 System Control Lines

Symbol	Pin		Input/ Output	Remarks															
	CP-84	FP-80A																	
$\overline{\text{BUSREQ}}$	18	7	Input	Bus request: This line is asserted by an external device to request control of the bus. When this line is driven active low, the internal bus master waits until the end of the current machine cycle, then places the address lines, the data lines, and some of the memory I/O interface lines ( $\overline{\text{RD}}$ , $\overline{\text{WR}}$ , $\overline{\text{ME}}$ , and $\overline{\text{IOE}}$ ) into the high impedance state.															
$\overline{\text{BUSACK}}$	20	9	Output	Bus acknowledge: This line is used by the internal bus master to notify an external device by sending a $\overline{\text{BUSACK}}$ signal that a $\overline{\text{BUSREQ}}$ signal has been received and the bus has been released.															
$\overline{\text{HALT}}$	24	13	Output	HALT: This line is asserted whenever a HALT or SLP instruction is executed. It indicates that the HD64180S is in the halt, sleep, or system stop mode. This line is also used in conjunction with the $\overline{\text{LIR}}$ and ST lines to indicate the status of the CPU and internal DMAC.															
$\overline{\text{LIR}}$	22	11	Output	Load instruction register: This line is asserted during opcode fetch cycles. This line can also be used to output the Z80 peripheral LSI interface signal.															
ST	21	10	Output	Status: This line is used, together with $\overline{\text{LIR}}$ and $\overline{\text{HALT}}$ , to indicate the internal status of the HD64180S (see table).															
				<table border="1"> <thead> <tr> <th></th> <th><math>\overline{\text{HALT}}</math></th> <th><math>\overline{\text{LIR}}</math></th> <th>ST</th> <th>Status</th> </tr> </thead> <tbody> <tr> <td>(1)</td> <td>1</td> <td>0*1</td> <td>0</td> <td>CPU active (first byte of an opcode fetch)</td> </tr> <tr> <td></td> <td></td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>		$\overline{\text{HALT}}$	$\overline{\text{LIR}}$	ST	Status	(1)	1	0*1	0	CPU active (first byte of an opcode fetch)			1		
	$\overline{\text{HALT}}$	$\overline{\text{LIR}}$	ST	Status															
(1)	1	0*1	0	CPU active (first byte of an opcode fetch)															
		1																	

\*1 The upper value shows the  $\overline{\text{LIR}}$  pin status when the LIRE bit of the operation mode control register is 1, and the lower value shows the  $\overline{\text{LIR}}$  pin status when the LIRE bit is 0.

Symbol	Pin Number		Input/ Output	Remarks
	CP-84	FP-80A		
				(2) 1      0*2    1      CPU active (second or third byte of an opcode fetch)
				1
				(3) X*1      1      0      DMAC operation
				(4) 1      1      1      Normal operating mode (other than (1), (2), or (3)) Reset mode
				(5) 0      0*2    0      Opcode fetch during halt mode (no instructions are executed)
				1
				(6) 0      1      1      Halt mode (other than (3) or (5)) Sleep mode (other than (3)) System stop mode
$\overline{\text{REF}}$	23	12	Output	Refresh: This line is asserted during the DRAM refresh cycle. During this cycle, the refresh address is output on the 12 low-order lines (A0 – A11) of the address bus.

\*1 X: Don't care

\*2 The upper value shows the  $\overline{\text{LIR}}$  pin status when the LIRE bit of the operation mode control register is 1, and the lower value shows the  $\overline{\text{LIR}}$  pin status when the LIRE bit is 0.

## 2.2.8 Interrupt Lines

Symbol	Pin Number		Input/ Output	Remarks
	CP-84	FP-80A		
$\overline{\text{NMI}}$	13	2	Input	Non-maskable interrupt: This line is used to request a non-maskable interrupt.

Symbol	Pin Number		Input/ Output	Remarks								
	CP-84	FP-80A										
$\overline{\text{INT0}}$	14	3	Input	Interrupt 0: This line is used to request a level-0 maskable interrupt. There are three different modes for level-0 interrupts (see table).								
				<table border="1"> <thead> <tr> <th>Mode</th> <th>Function</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Executing the instruction on the data bus</td> </tr> <tr> <td>1</td> <td>Executing the instruction at address 0038H</td> </tr> <tr> <td>2</td> <td>Vector mode</td> </tr> </tbody> </table>	Mode	Function	0	Executing the instruction on the data bus	1	Executing the instruction at address 0038H	2	Vector mode
Mode	Function											
0	Executing the instruction on the data bus											
1	Executing the instruction at address 0038H											
2	Vector mode											
$\overline{\text{INT1}}$	15	4	Input	Interrupt 1 and 2: These lines are used respectively to request level-1 and level-2 maskable interrupts (vector mode).								
$\overline{\text{INT2}}$	16	5	Input									

### 2.2.9 DMA Lines

Symbol	Pin Number		Input/ Output	Remarks
	CP-84	FP-80A		
$\overline{\text{DREQ0}}$	80	66	Input	DMA request for channel 0: This line is used to request a DMA transfer using internal DMAC channel 0.
$\overline{\text{DREQ1}}$	81	67	Input	DMA request for channel 1: This line is used to request a DMA transfer using internal DMAC channel 1.
$\overline{\text{TEND0}}$	82	68	Output	Transfer end for channel 0: This line is used to indicate the end of a DMA transfer using internal DMAC channel 0. It is asserted synchronously with the read cycle upon the last data transfer.
$\overline{\text{TEND1}}$	83	69	Output	Transfer end for channel 1: This line is used to indicate the end of a DMA transfer using internal DMAC channel 1. It is asserted synchronously with the read cycle upon the last data transfer.

## 2.2.10 Serial I/O (MSCI) Lines

Symbol	Pin		Input/ Output	Remarks
	CP-84	FP-80A		
TXDM	71	57	Output	Transmit data from the MSCI: This line is used to output transmit data from the MSCI.
RXDM	68	54	Input	Receive data to the MSCI: This line is used to input receive data to the MSCI.
TXCM	70	56	Input/ Output	<p>Transmit clock for the MSCI: This line is used to input/output the MSCI transmit clock.</p> <p>Three programmable modes:</p> <p>Input: • External transmit clock</p> <p>Output: • Transmit clock from the on-chip baud rate generator</p> <p>• Receive clock (used as the transmit clock)</p>
RXCM	69	55	Input/ Output	<p>Receive clock for the MSCI: This line is used to input/output the MSCI receive clock. This line can also be used to input the ADPLL operating clock.</p> <p>Four programmable modes:</p> <p>Input: • External receive clock</p> <p>• ADPLL operating clock</p> <p>Output: • Receive clock extracted by the ADPLL (when the on-chip baud rate generator is used as the ADPLL operating clock )</p> <p>• Receive clock from the on-chip baud rate generator</p>
$\overline{\text{RTSM}}$	65	51	Output	Request to send for the MSCI: Indicates that the HD64180S has data to be output to a communications device such as modem. The output level can be automatically controlled by MSCI operation (auto-enable function). This line can also be used as a general purpose output port.
$\overline{\text{DCDM}}$	66	52	Input	Data carrier detect for the MSCI: Indicates that a communications device such as modem is receiving valid data from the communications line. MSCI receive operation can be automatically controlled by this input (auto-enable function). This line can also be used as a general purpose input port.

Symbol	Pin Number		Input/ Output	Remarks
	CP-84	FP-80A		
$\overline{\text{CTSM}}$	67	53	Input	Clear to send for the MSCI: Indicates that a communications device such as modem is ready to send data to the communications line. MSCI transmit operation can be automatically controlled by this input (auto-enable function). This line can also be used as a general purpose input port.
$\overline{\text{SYNC}}$	64	50	Input/ Output	Synchronization for the MSCI: This line is used as an input in the external byte synchronous mode. Synchronization is established at the falling edge of $\overline{\text{SYNC}}$ . This line is used as an output in the byte sync (mono- or bi-) or HDLC mode. It indicates the inverse of the SYNCD/FLGD bit in MSCI status register 1 (MST1)*. In the asynchronous mode, this line is used as an input. The input value does not affect operation.

\* For details concerning MSCI status register 1 (MST1), see section 4.2.10 "MSCI Status Register 1."

### 2.2.11 Serial I/O (ASCI/CSIO) Lines

Symbol	Pin Number		Input/ Output	Remarks
	CP-84	FP-80A		
TXDA	79	65	Output	Transmit data from the ASCI/CSIO: This line is used to output transmit data from the ASCI/CSIO.
RXDA	76	62	Input	Receive data to the ASCI/CSIO: This line is used to input receive data to the ASCI/CSIO.
TXCA	78	64	Input/ Output	Transmit clock for the ASCI/CSIO: This line is used to input/output the ASCI/CSIO transmit clock. Two programmable modes: Input: • External transmit clock Output: • Transmit clock from the on-chip baud rate generator
RXCA	77	63	Input/ Output	Receive clock for the ASCI/CSIO: This line is used to input/output the ASCI/CSIO receive clock. Two programmable modes: Input: • External receive clock Output: • Receive clock from the on-chip baud rate generator

Symbol	Pin Number		Input/ Output	Remarks
	CP-84	FP-80A		
$\overline{\text{RTSA}}$	72	58	Output	Request to send for ASCII/CSIO: Indicates that the HD64180S has data to be output to a communications device such as a modem. The output level can be automatically controlled by the ASCII/CSIO operation (auto-enable function). This line can also be used as a general purpose output port.
$\overline{\text{DCDA}}$	73	59	Input	Data carrier detect for ASCII/CSIO: Indicates that a communications device such as a modem is receiving valid signals from the communications line. ASCII/CSIO receive operation can be automatically controlled by this input (auto-enable function). This line can also be used as a general purpose input port.
$\overline{\text{CTSA}}$	75	61	Input	Clear to send for ASCII/CSIO: Indicates that a communications device such as modem is ready to send data to the communications line. ASCII/CSIO transmit operation can be controlled automatically by this input (auto-enable function). This line can also be used as a general purpose input port.

### 2.2.12 Timer Lines

Symbol	Pin Number		Input/ Output	Remarks
	CP-84	FP-80A		
TIN <sub>0</sub>	5	75	Input	Timer inputs for channels 0 and 1: Event counter signals are input via these lines.
TIN <sub>1</sub>	6	76	Input	
TOUT <sub>0</sub>	7	77	Output	Timer outputs for channels 0 and 1: Timer signals are output via these lines.
TOUT <sub>1</sub>	8	78	Output	

## Section 3. Central Processing Unit (CPU)

### 3.1 Overview

The HD64180S's on-chip CPU features the following:

- Software-compatibility with HD64180Z's CPU
- Internal memory management unit (MMU) supports 1 Mbyte of physical address space (memory only)
- Four external interrupt lines ( $\overline{\text{NMI}}$ ,  $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ , and  $\overline{\text{INT2}}$ ) and 15 internal interrupt lines which can be software enabled/disabled
- Internal I/O (MSCI, ASCI/CSIO, DMAC, etc.) controlled by I/O instructions
- Three special operation modes (halt, sleep, and system stop)

### 3.2 Basic CPU Architecture

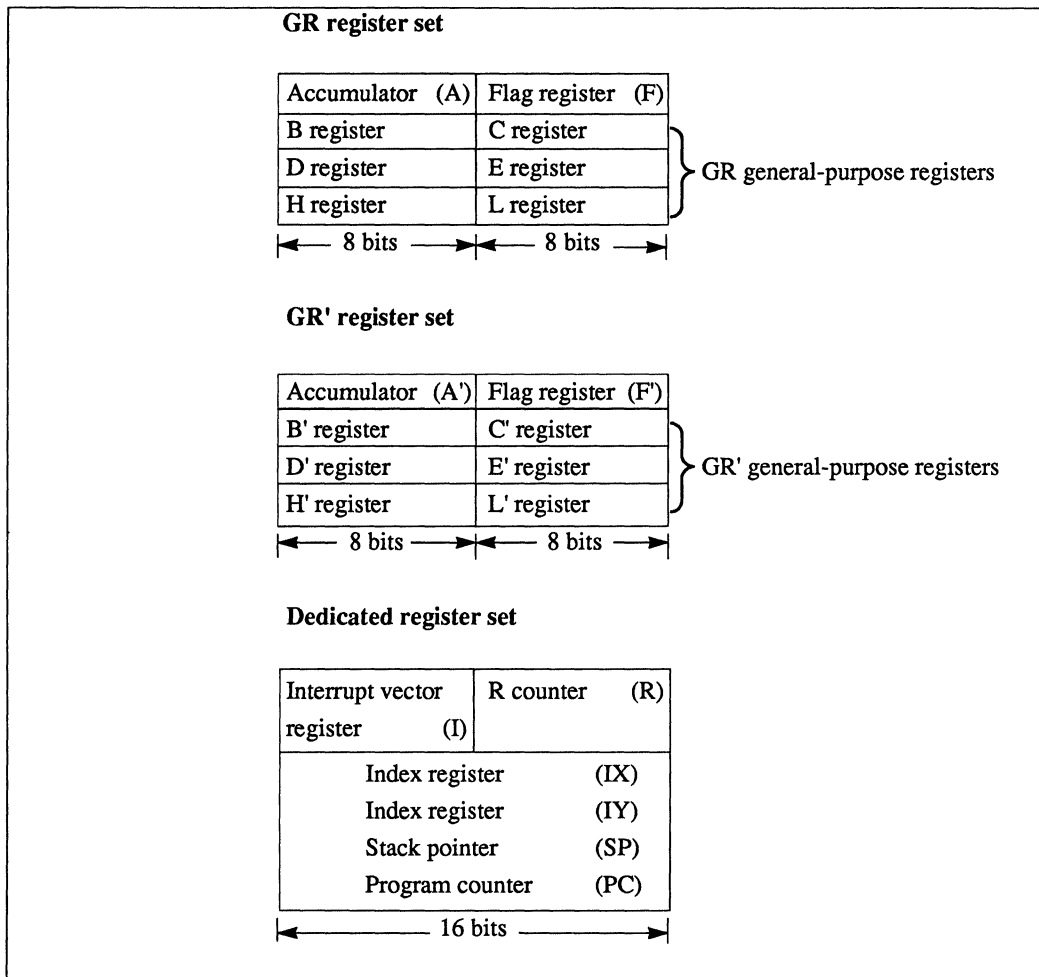
This section provides details about the CPU internal registers, addressing modes, instruction set, and the I/O space.

#### 3.2.1 CPU Internal Registers

The CPU has two general register sets (GR and GR') and one dedicated register set. GR and GR' each consist of one 8-bit accumulator, one 8-bit flag register, and six 8-bit general-purpose registers.

The dedicated register set consists of the interrupt vector register (I), the R counter (R), two index registers (IX and IY), the stack pointer (SP), and the program counter (PC).

Figure 3-1 shows the configuration of the CPU internal registers.



**Figure 3-1. Configuration of CPU Internal Registers**

Functions of the various registers are explained below.

**Accumulators (A and A'):** The accumulators are operational registers used for 8-bit arithmetic, logical, and shift operations. The contents of accumulator A can be replaced with the contents of A' by executing an EX AF, AF' instruction. Following a reset operation, the values of the A and A' accumulators are undefined.

**Flag registers (F and F'):** Flag registers indicate the status of the operation result. The contents of flag register F can be replaced with the contents of F' by executing an EX AF, AF' instruction. Following a reset operation, the values of the F and F' registers are undefined.



	7	6	5	4	3	2	1	0
Bit Name	S	Z	-*1	H	-*1	P/V	N	C
Initial Value	X*2	X*2	X*2	X*2	X*2	X*2	X*2	X*2

\*1 Reserved. These bits can be read/written by a PUSH AF or POP AF instruction, respectively.

\*2 Undefined.

### Bit 7: S (Sign) Flag

#### S Functions

---

0 The result of an operation is positive (MSB = 0)

---

1 The result of an operation is negative (MSB = 1)

---

### Bit 6: Z (Zero) Flag

#### Z Functions

---

0 The result of an operation is not 0

---

1 The result of an operation is 0

---

**Bit 5:** Reserved. This bit can be read/written by a PUSH AF or POP AF instruction, respectively.

### Bit 4: H (Half Carry) Flag

H is used for compensation in binary coded decimal (BCD) operations (DAA instruction).

#### H Functions

---

0 Neither carry at the fourth bit from the LSB, nor borrow at the fourth bit from the MSB has occurred.

---

1 A carry has occurred at the fourth bit from the LSB, or a borrow has occurred at the fourth bit from the MSB.

---

**Bit 3:** Reserved. This bit can be read/written by a PUSH AF or POP AF instruction, respectively.

### Bit 2: P/V (Parity/Overflow) Flag

The P/V bit functions as either a parity or overflow bit. As a parity bit, it shows whether the number of bits set to 1 in the accumulator after a logical operation is even or odd. As an overflow bit, it shows whether the result of a signed 8-bit arithmetic operation is between -128 and +127 or whether the result of a signed 16-bit arithmetic operation is between -32768 and +32767.

P/V	Parity	Overflow (8 bit)	Overflow (16 bit)
0	Odd	Result lies between -128 and +127	Result lies between -32768 and +32767
1	Even	Result lies outside of -128 to +127	Result lies outside of -32768 to +32767

### Bit 1: N (Negate) Flag

#### N Functions

0	Addition instruction (ADD, INC, etc.) has been executed
1	Subtraction instruction (SUB, DEC, CP, etc.) has been executed

### Bit 0: C (Carry) Flag

The C bit is set when a carry or borrow at the MSB has been generated by an operation. When neither carry nor borrow has occurred, this bit is reset. Carries and borrows are categorized as follows:

- Carry generated by addition
- Borrow generated by subtraction
- Carry generated by shift or rotation

#### C Functions

0	Neither carry nor borrow has been generated at the MSB
1	Carry or borrow has been generated at the MSB

**General-Purpose Registers B, C, D, E, H, and L:** The six 8-bit general-purpose registers in register set GR are used for operations and addressing. Registers B and C, D and E, or H and L can be used together as 16-bit registers. The reset values are undefined.

**General-Purpose Registers B', C', D', E', H', and L':** The six 8-bit general-purpose registers in register set GR' function in the same way as registers B, C, D, E, H, and L. They can be used in place of register set GR. An EXX instruction is used to swap the contents of the GR and GR' general-purpose registers. The reset values are undefined.

**Interrupt Vector Register (I):** The interrupt vector register specifies the high order byte of a 16-bit interrupt vector. This register is used for  $\overline{\text{INT}}_0$  mode 2,  $\overline{\text{INT}}_1$ ,  $\overline{\text{INT}}_2$ , and internal interrupts except TRAP. This register is read or written using LD A, I or LD I, A instructions. By a reset, this register is initialized to 00H. For details about this register, see "Interrupt Vector Register (I)" in section 3.6.2 "Interrupt Control Registers and Interrupt Enable Flags."

**R Counter (R):** The 7-bit R counter indicates the number of executed opcode fetch cycles. The most significant bit is reserved. This register is read or written using an LD A, R or LD R, A instructions. The reserved bit can also be read and written. Upon reset, this counter is initialized to 00H.

**Note:** The content of the R counter has no relationship to the refresh address. The refresh address is generated by another internal counter, which cannot be accessed by the user.

**Index Registers (IX and IY):** The 16-bit index registers are used for index addressing and 16-bit operations.

For index addressing, the base address is loaded into the index register. The effective address of the data in memory to be accessed is generated by adding a signed 8-bit displacement to the base address.

General-purpose registers (BC, DE), index registers (IX, IY) or the stack pointer (SP) can be used for the 16-bit operands (xx or yy) of instructions, such as ADD IX, xx or ADD IY, yy, that use index registers IX and IY. The contents of index register IX and IY are undefined by a reset.

**Stack Pointer (SP):** The 16-bit stack pointer register holds the address of the top of the stack. It is initialized to 0000H by a reset.

**Program Counter (PC):** The 16-bit program counter register holds the logical address of the next instruction to be executed.

The contents of this register are normally incremented by one each time a 1-byte opcode or operand is accessed. When a jump instruction is executed, the jump destination address is loaded into this register. It is initialized to 0000H by a reset.

### 3.2.2 Addressing Modes

The CPU supports eight addressing modes: implied, register direct, register indirect, indexed, extended, immediate, relative and I/O.

**Implied (IMP) Addressing:** In the implied addressing mode, an address location is implied by the opcode, rather than being stated explicitly within the instruction. Such instructions operate on the accumulator (A), index registers (IX and IY), stack pointer (SP), general-purpose register HL, and bit positions specified by opcodes.

**Register Direct (REG) Addressing:** In the register direct addressing mode, an 8- or 16-bit register is addressed directly. The g, g', ww, xx, yy, and zz opcode fields indicate the register. The field codes and corresponding registers are listed below.

## 8-Bit Register Specification

g or g' field			Specified Register
0	0	0	B
0	0	1	C
0	1	0	D
0	1	1	E
1	0	0	H
1	0	1	L
1	1	0	–
1	1	1	A

## 16-Bit Register Specification

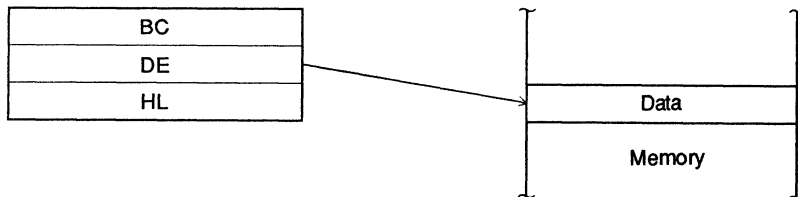
ww field		Specified Register
0	0	BC
0	1	DE
1	0	HL
1	1	SP

yy field		Specified Register
0	0	BC
0	1	DE
1	0	IY
1	1	SP

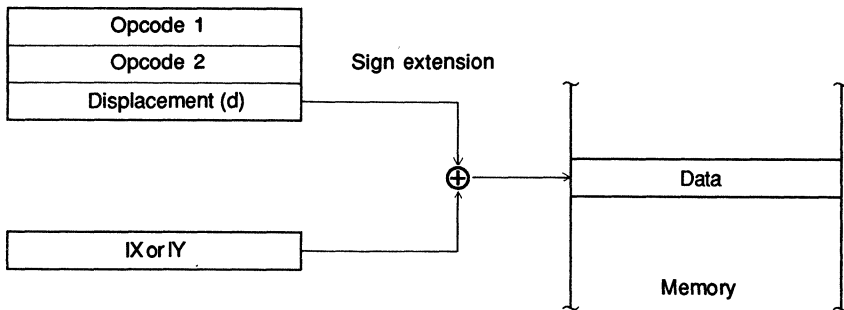
xx field		Specified Register
0	0	BC
0	1	DE
1	0	IX
1	1	SP

zz field		Specified Register
0	0	BC
0	1	DE
1	0	HL
1	1	AF

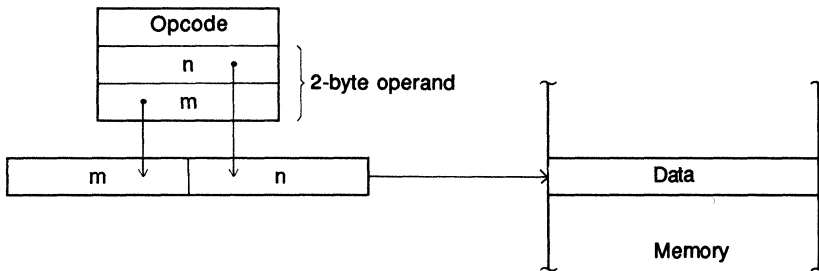
**Register Indirect (REGI) Addressing:** In the register indirect addressing mode, the contents of two general-purpose registers indicate the 16-bit memory address.



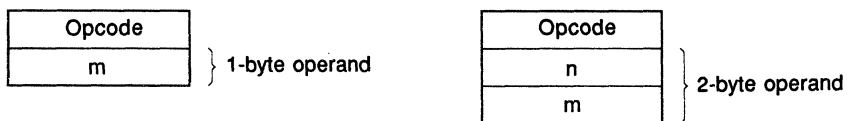
**Indexed (INDX) Addressing:** In the indexed addressing mode, the effective address of the data in memory is generated by adding a signed 8-bit displacement (d) to the contents of an index register (IX or IY).



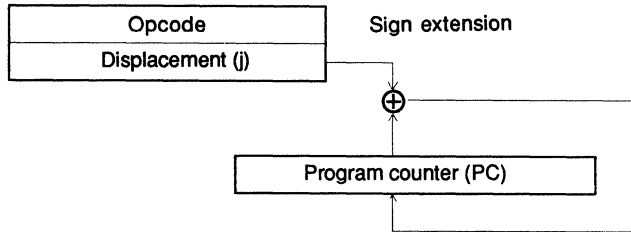
**Extended (EXT) Addressing:** In the extended addressing mode, the 16-bit data address is specified by the 2-byte operand (m, n) following the opcode.



**Immediate (IMMED) Addressing:** In the immediate addressing mode, a 1-byte operand (m) or 2-byte operand (m, n) following the opcode is used as data.



**Relative (REL) Addressing:** The relative addressing mode is only used for jump instructions. A jump address is generated by adding a signed 8-bit displacement (j) to the contents of the program counter (PC). For conditional jump instructions, the jump address is only generated when the specified jump condition is satisfied.



**I/O Addressing:** The I/O addressing mode is only used for I/O instructions. The specified address is handled as an I/O address ( $\overline{\text{IOE}} = 0$ ). An address is output in one of the following ways:

- (1) The operand contents are output to address bus lines A0 – A7, and the accumulator contents are output to A8 – A15.
- (2) The C register contents are output to address bus lines A0 – A7, and the B register contents are output to A8 – A15.
- (3) The operand contents are output to address bus lines A0 – A7 and 00H is output to A8 – A15. (This can be used when accessing an internal I/O register.)
- (4) The C register contents are output to address bus lines A0 – A7 and 00H is output to A8 – A15 as an address. (This can be used when accessing an internal I/O register.)

### 3.2.3 Instruction Set

The CPU instruction set can be divided into five groups:

- Data manipulation instructions
- Data transfer instructions
- Program control instructions
- I/O instructions
- Special control instructions

Instruction length varies from one to four bytes. Typical formats are shown below.

1-byte instruction	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 2px 5px;">7</td> <td style="padding: 2px 5px;">6</td> <td style="padding: 2px 5px;">5</td> <td style="padding: 2px 5px;">4</td> <td style="padding: 2px 5px;">3</td> <td style="padding: 2px 5px;">2</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> </tr> <tr> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">1</td> <td colspan="3" style="padding: 2px 5px;">g</td> <td colspan="3" style="padding: 2px 5px;">g'</td> </tr> </table>	7	6	5	4	3	2	1	0	0	1	g			g'			LD g, g'																								
7	6	5	4	3	2	1	0																																			
0	1	g			g'																																					
2-byte instruction	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 2px 5px;">7</td> <td style="padding: 2px 5px;">6</td> <td style="padding: 2px 5px;">5</td> <td style="padding: 2px 5px;">4</td> <td style="padding: 2px 5px;">3</td> <td style="padding: 2px 5px;">2</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> </tr> <tr> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">0</td> <td colspan="3" style="padding: 2px 5px;">g</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> </tr> <tr> <td colspan="8" style="padding: 2px 5px;">m</td> </tr> </table>	7	6	5	4	3	2	1	0	0	0	g			1	1	0	m								LD g, m Immediate data																
7	6	5	4	3	2	1	0																																			
0	0	g			1	1	0																																			
m																																										
3-byte instruction	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 2px 5px;">7</td> <td style="padding: 2px 5px;">6</td> <td style="padding: 2px 5px;">5</td> <td style="padding: 2px 5px;">4</td> <td style="padding: 2px 5px;">3</td> <td style="padding: 2px 5px;">2</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> </tr> <tr> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">1</td> </tr> <tr> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">1</td> <td colspan="3" style="padding: 2px 5px;">g</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> </tr> <tr> <td colspan="8" style="padding: 2px 5px;">d</td> </tr> </table>	7	6	5	4	3	2	1	0	1	1	0	1	1	1	0	1	0	1	g			1	1	0	d								LD g, (IX+d) Displacement								
7	6	5	4	3	2	1	0																																			
1	1	0	1	1	1	0	1																																			
0	1	g			1	1	0																																			
d																																										
4-byte instruction	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 2px 5px;">7</td> <td style="padding: 2px 5px;">6</td> <td style="padding: 2px 5px;">5</td> <td style="padding: 2px 5px;">4</td> <td style="padding: 2px 5px;">3</td> <td style="padding: 2px 5px;">2</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> </tr> <tr> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">1</td> </tr> <tr> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> </tr> <tr> <td colspan="8" style="padding: 2px 5px;">d</td> </tr> <tr> <td colspan="8" style="padding: 2px 5px;">m</td> </tr> </table>	7	6	5	4	3	2	1	0	1	1	0	1	1	1	0	1	0	0	1	1	0	1	1	0	d								m								LD (IX+d), m Displacement Immediate data
7	6	5	4	3	2	1	0																																			
1	1	0	1	1	1	0	1																																			
0	0	1	1	0	1	1	0																																			
d																																										
m																																										

For details concerning the instruction set, see Appendix A.

#### Supplemental Explanation: Conditional jump and call instructions

The execution of a conditional jump instruction (JP f, mn) varies according to whether the jump condition is satisfied or not.

Consider the following example in which the JP NZ, 6000H instruction is executed (figures 3-2 and 3-3).

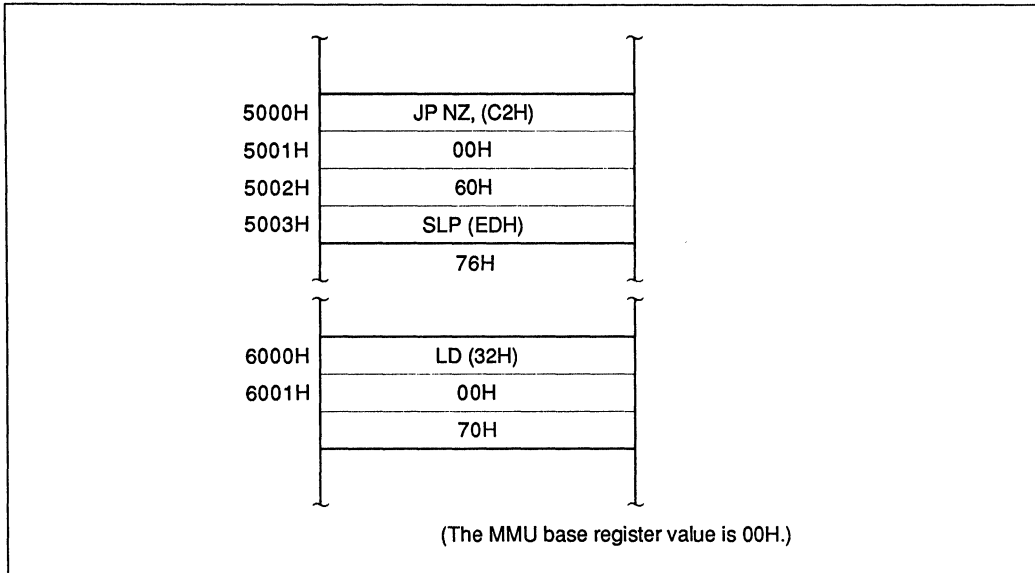


Figure 3-2. When JP NZ, 6000H Instruction is at 5000H

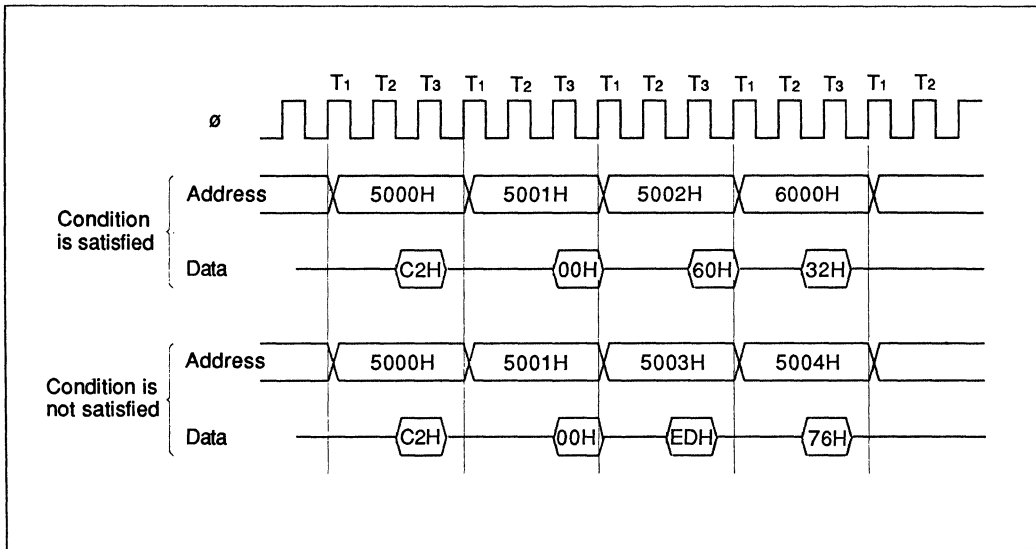


Figure 3-3. Conditional Branch Execution Timing



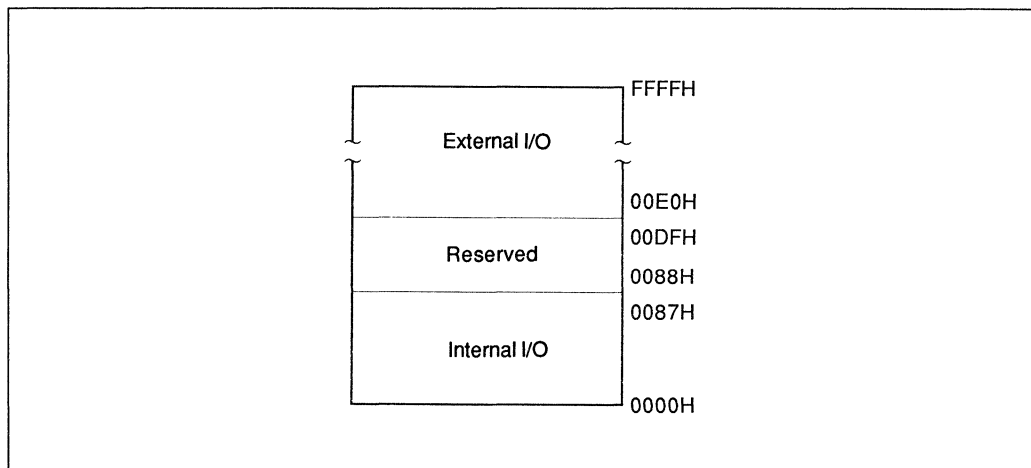
If the jump condition is not satisfied in this case, the second byte in memory (m) of the JP f, mn instruction operand is not read. The number of executed states in this case is 6. Similarly, for conditional call instructions (CALL f, mn), if the condition is not satisfied the second byte (m) is not read.

### 3.2.4 I/O Space

All registers except the accumulator, flag, general purpose, and dedicated registers which are CPU registers, exist in the I/O space and can be accessed by I/O instructions. These registers occupy 136 bytes of the I/O space (addresses 0000H to 0087H). Addresses 00E0H to FFFFH in the I/O space are allocated to external I/O.

Addresses 0088H to 00DFH are reserved and cannot be used.

Figure 3-4 shows the I/O space configuration.



**Figure 3-4. I/O Space Configuration**

#### • Precautions for internal I/O registers

(1) The internal I/O registers are located addresses 0000H to 0087H in the 64 kbyte I/O space. The high-order byte of the I/O address must therefore be 00H. For example, an OUT (m), A instruction causes the A register contents to be directly output to the high-order byte of the I/O address. Therefore, this instruction cannot be used to write to the internal I/O registers. For the same reason, OTIR, OTDR, INIR, and INDR must not be used. When other I/O instructions are used to access the internal I/O registers, the high-order byte of the I/O address must be 00H.

Thus, for example, the A register must be initialized to 00H before executing an IN A, (m) instruction.

IN0 g, (m), OUT0 (m), g, OTIM, OTIMR, OTDM, OTDMR, and TSTIO m instructions are convenient for accessing the internal I/O registers. These instructions automatically set the high-order address to 00H.

(2) If an external I/O is referenced by the same I/O address as an internal I/O register, the following operations are performed:

- Write operation: The value written in the internal I/O register is also written to the external I/O.
- Read operation: The external I/O can be read, but the read data cannot be sent to the CPU. Rather, the internal I/O value is sent.

In either case, the number of wait states is 0.

(3) Note that the I/O addresses for internal I/O registers (the MMU register, etc.) that are functionally identical for the HD64180R1 and HD64180Z have been changed in the HD64180S.

### 3.3 CPU Basic Operation Timing

#### 3.3.1 Outline

This section explains the timing of the following CPU basic operations:

- Opcode fetch
- Memory read/write
- I/O read/write
- Basic instruction execution
- Interfacing with Z80\*-based peripheral LSIs

Basic operations consist of one or more machine cycles (MCs). For memory or I/O accesses, a machine cycle consists of three states, T<sub>1</sub>, T<sub>2</sub>, T<sub>3</sub>. For an internal cycle, a machine cycle consists of a single cycle state (T<sub>i</sub>).

\* Z80 is a trademark of Zilog, Inc.

For slow memory or I/O subsystems, a Tw (wait) state can be inserted between the T2 and T3 states. Insertion of the Tw state is controlled either by hardware (using the  $\overline{\text{WAIT}}$  line) or by software (using internal registers). For details, see section 9 "Wait Controller."

Table 3-1 shows the internal register used for interfacing with Z80-based peripherals.

**Table 3-1. Register for Interfacing with Z80-based Peripherals**

Register Name	Symbol	I/O Address	Initial Value*	Read/Write
			MSB ↔ LSB	
Operation mode control register	OMCR	0004H	11100000	R/W

\* "Initial value" means the value after a hardware reset.

### 3.3.2 Opcode Fetch Timing

Figure 3-5 shows the timing of an opcode fetch with no wait states. In the first half of the T1 state, the contents of the PC are output to the address bus (A0 – A19). When the MMU is used, address information is extended to 20 bits. In the second half of the T1 state, the  $\overline{\text{ME}}$  and  $\overline{\text{RD}}$  signals are asserted to enable memory access.

The opcode is read from the data bus at the rising edge of the  $\phi$  clock of the T3 state. The  $\overline{\text{LIR}}$  signal remains active (low) from the first half of the T1 state to the first half of the T3 state, indicating that an opcode fetch cycle is in progress\*1.

If this cycle is the first opcode fetch cycle, the ST signal remains low from the middle of the T1 state to the end of the T3 state. For the second or third opcode fetch cycle, the ST signal remains high throughout this period. Thus, by checking the level of the ST and  $\overline{\text{LIR}}$  signals, it can be determined whether this cycle is the first or the other opcode fetch cycle.

\*1 Asserting the  $\overline{\text{LIR}}$  signal can be inhibited by clearing the LIRE bit in the operation mode control register. For details, see section 3.3.6 "Interfacing Z80-based Peripheral LSIs."

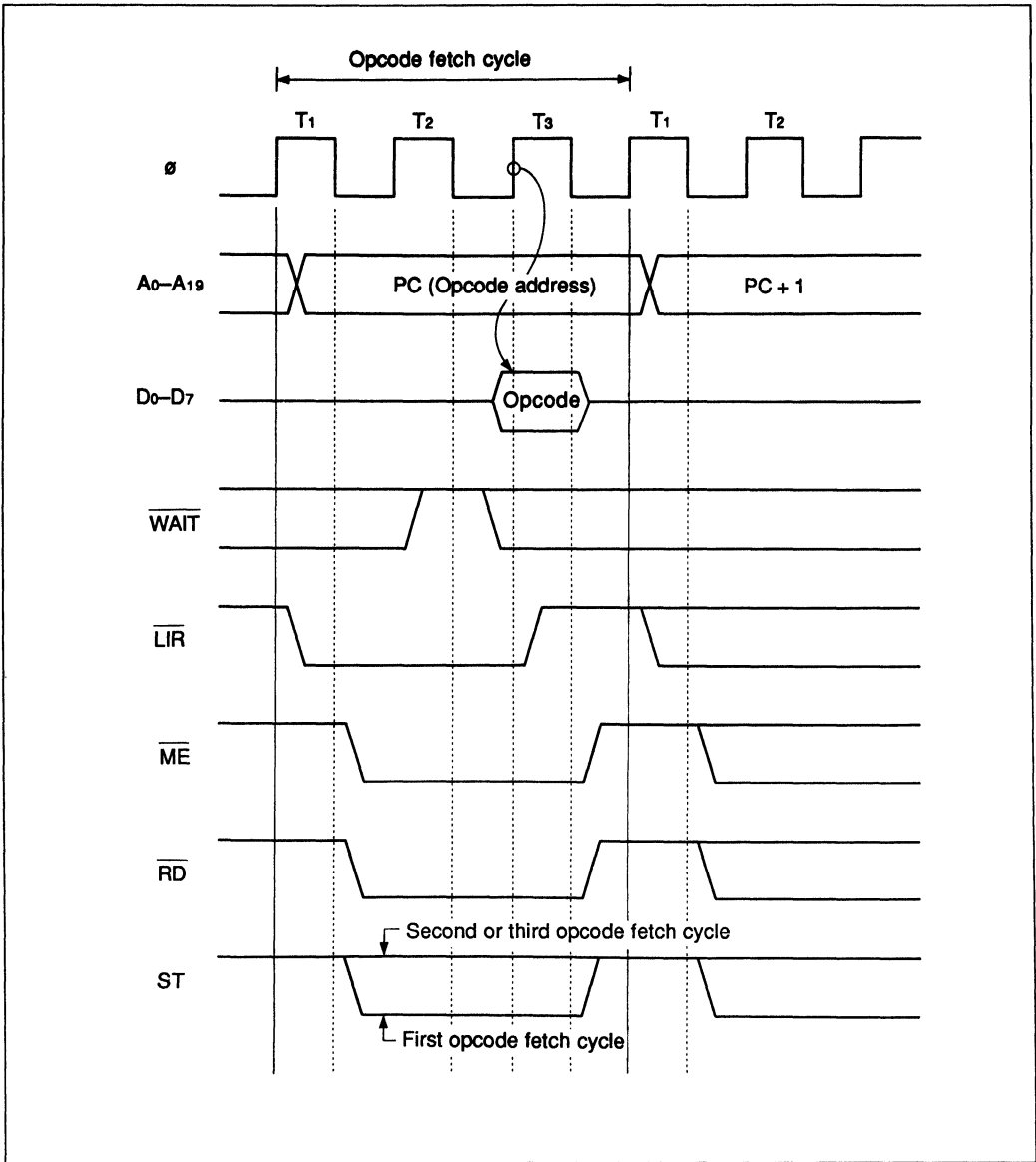


Figure 3-5. Opcode Fetch Timing

Figure 3-6 shows the timing of an opcode fetch with wait states. If the  $\overline{\text{WAIT}}$  signal is active at the falling edge of the T2 state, a Tw state is inserted in the next cycle. If the  $\overline{\text{WAIT}}$  signal is still active at the falling edge of the Tw state, another Tw state is inserted. If the  $\overline{\text{WAIT}}$  signal is not active, the bus cycle enters the T3 state and then completes the opcode fetch cycle. The  $\overline{\text{ME}}$ ,  $\overline{\text{RD}}$ , and  $\overline{\text{LIR}}$  signals maintain their current levels while wait states are being inserted in this way.

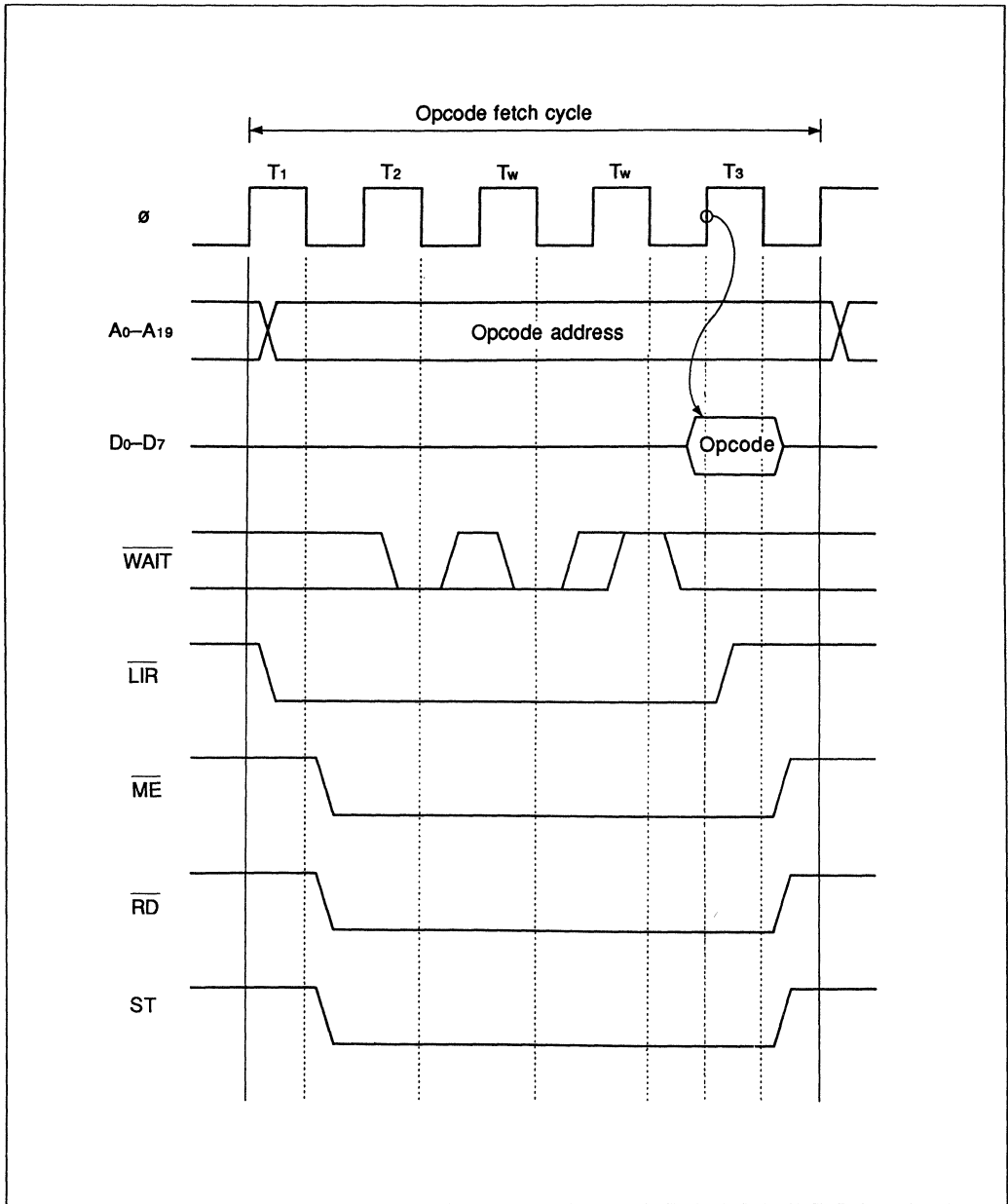


Figure 3-6. Opcode Fetch Timing with Wait States

### 3.3.3. Memory Data Read/Write Timing

Memory data read/write timing differs from opcode fetch timing in the following ways:

- The  $\overline{\text{LIR}}$  signal remains inactive.
- In memory read cycles, the data latch timing is delayed by a half clock cycle (data is read at the falling edge of the T3 state).

The output timing of addresses, and the  $\overline{\text{ME}}$  and the  $\overline{\text{RD}}$  signal timings are the same as for the opcode fetch cycle.

Immediate data, displacement, and extended address data is latched using the same timing scheme as memory data.

For memory write operations, the  $\overline{\text{ME}}$  signal becomes active during the second half of the T1 state and the  $\overline{\text{WR}}$  signal becomes active during the first half of the T2 state. Valid data is output on the data bus (D0 – D7) during the second half of the T1 state. The  $\overline{\text{ME}}$  and  $\overline{\text{WR}}$  signals become inactive during the second half of the T3 state. Data on the data bus is valid until the end of the (T3) state.

Figures 3-7 and 3-8 show memory read/write timing without/with wait states, respectively.

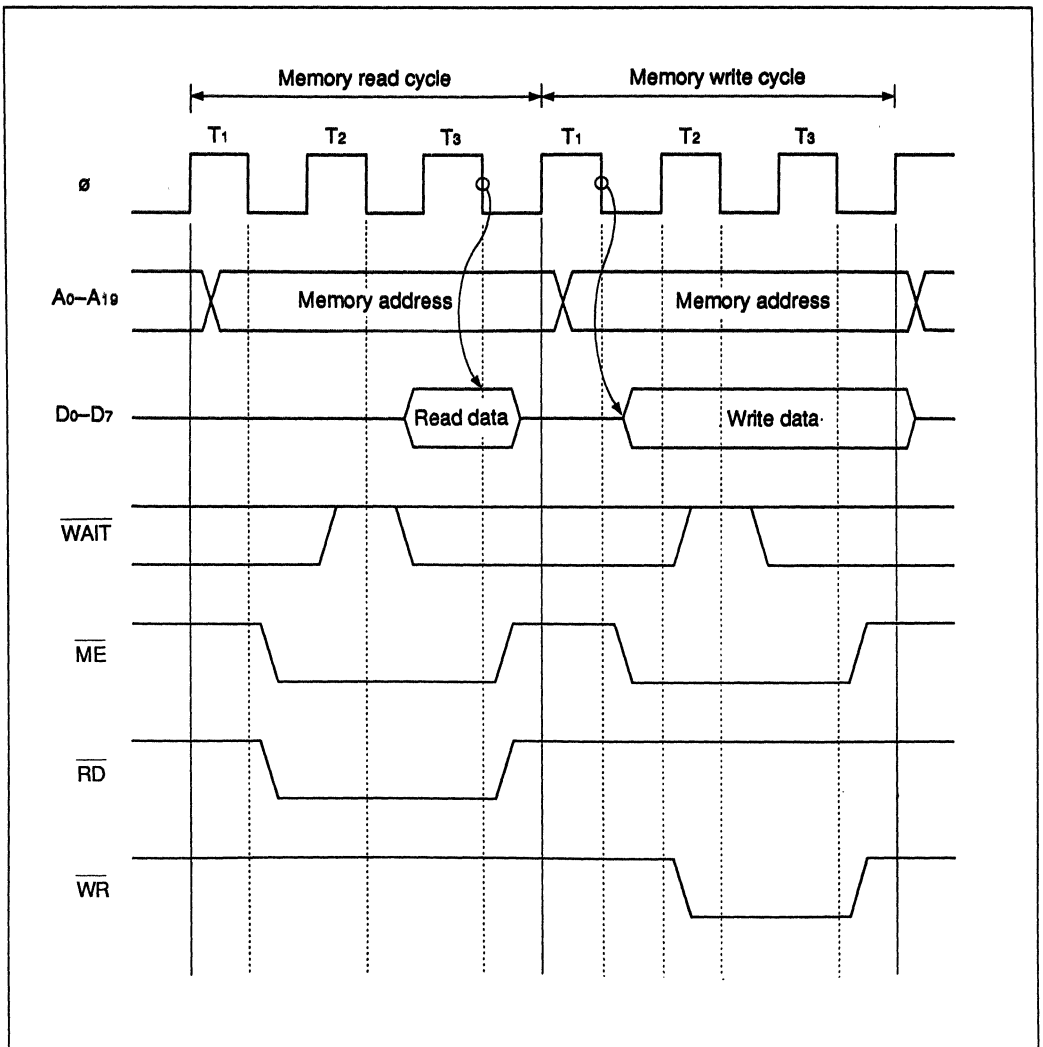


Figure 3-7. Memory Read/Write Timing

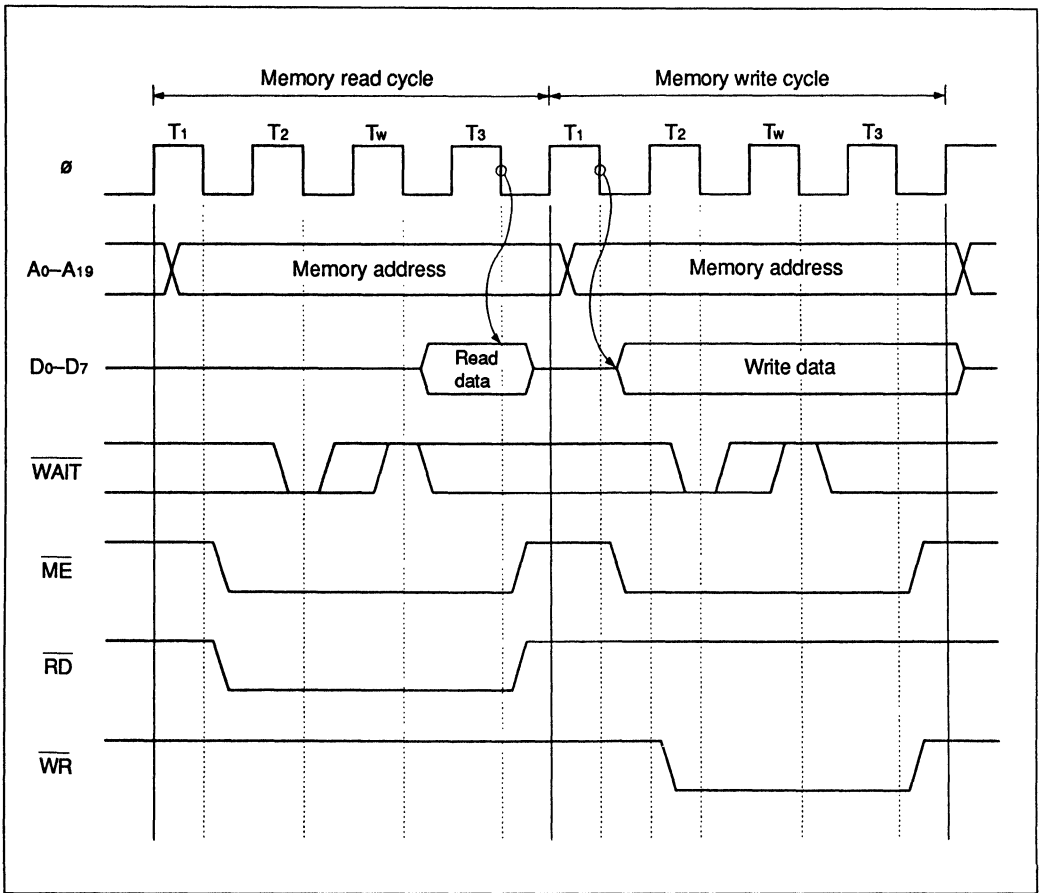


Figure 3-8. Memory Read/Write Timing with  $T_w$  States

### 3.3.4 I/O Read/Write Timing

I/O read/write timing is essentially the same as memory read/write timing (see section 3.3.3 "Memory Data Read/Write Timing"). Note, however, the following differences:

- The  $\overline{IOE}$  signal is used instead of the  $\overline{ME}$  signal.
- The I/O address is output to the address bus ( $A_0-A_{19}$ ). I/O addresses are not translated by the MMU and consequently lines ( $A_{16}-A_{19}$ ) always remain low.



Figure 3-9 shows external I/O read/write timing with wait states. Wait states are not inserted during internal I/O accesses.

The falling edge of the T1 clock or the rising edge of the T2 clock can be selected as the  $\overline{\text{IOE}}$  and  $\overline{\text{RD}}$  signal activation events. For details, see section 3.3.6 "Interfacing Z80-based Peripheral LSIs."

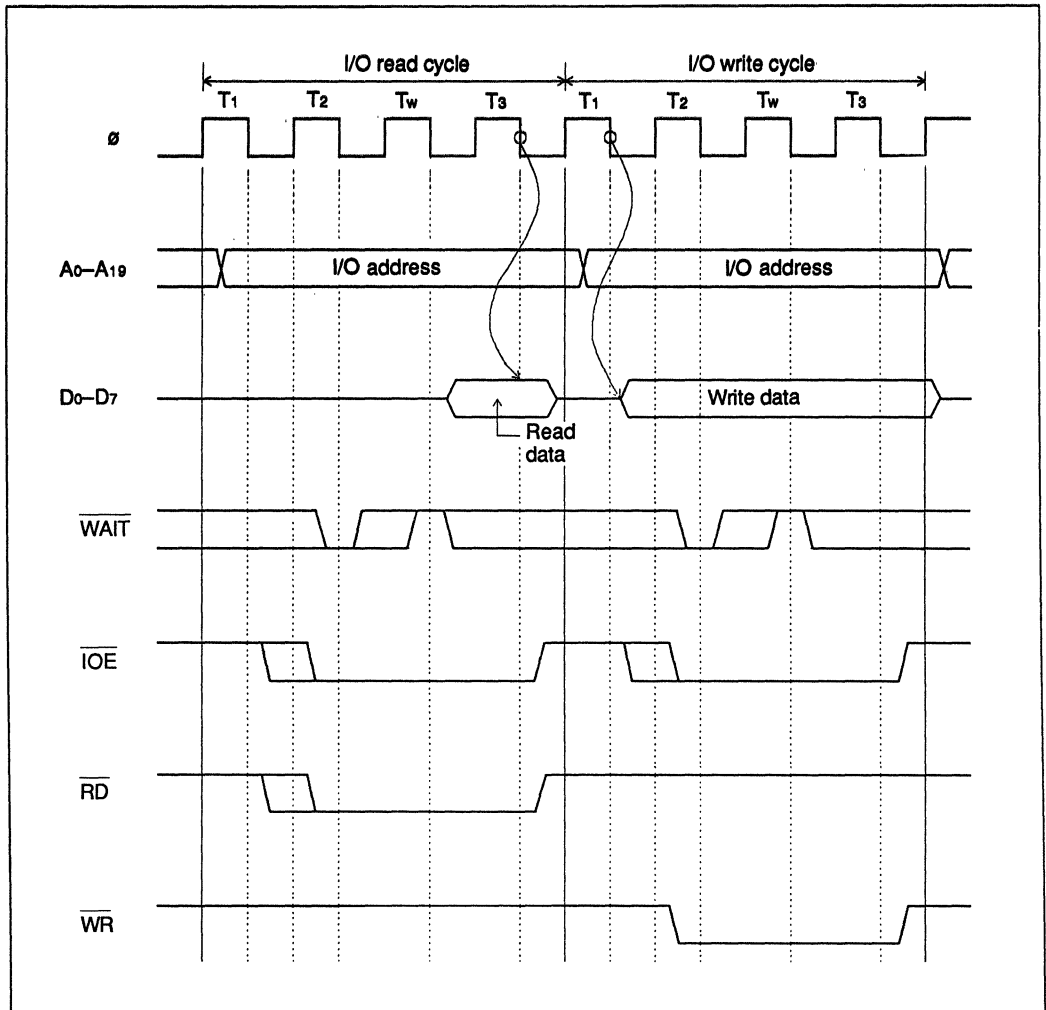
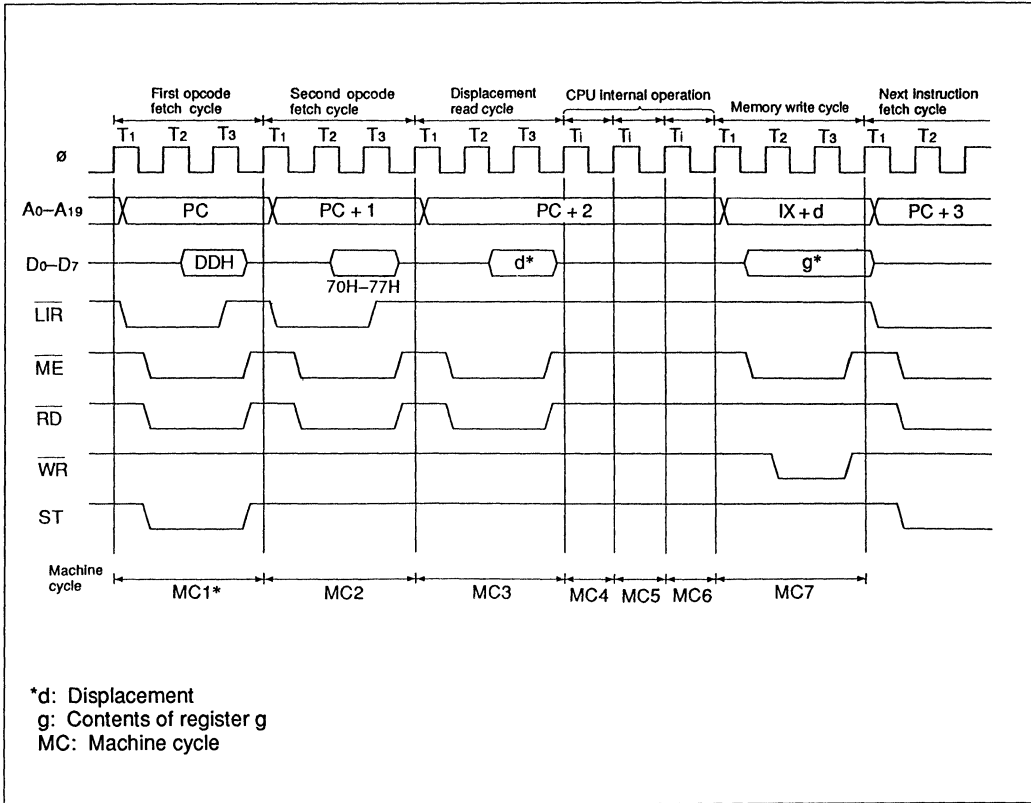


Figure 3-9. External I/O Read/Write Timing with Wait States

### 3.3.5 Basic Instruction Execution Timing

Figure 3-10 shows the timing of a data transfer instruction, LD (IX + d), g as a typical example. This 3-byte instruction involves two opcode fetch cycles and one displacement read cycle. In addition, during a memory write cycle the contents of register (g) are transferred to the address location determined by adding displacement (d) to the contents of index register (IX). In the example shown in figure 3-10, wait states are not inserted.



**Figure 3-10. Basic Instruction Execution Timing Example  
(LD (IX + d), g instruction)**

### 3.3.6 Interfacing Z80-based Peripheral LSIs

A special function allows the HD64180S to interface with Z80-based peripheral chips. The operation mode control register (OMCR) is used to select this function.

To permit interfacing with Z80-based peripheral LSIs, note the following differences from normal operation:

- $\overline{\text{LIR}}$  signal timing
- $\overline{\text{IOE}}$  and  $\overline{\text{RD}}$  signal timing
- RETI instruction (return from  $\overline{\text{INT0}}$  interrupt) timing

**Operation Mode Control Register (OMCR):** The OMCR register controls the  $\overline{\text{LIR}}$ ,  $\overline{\text{IOE}}$ , and  $\overline{\text{RD}}$  line outputs, and is used for interfacing with Z80-based peripheral chips. For more information about the  $\overline{\text{LIR}}$ ,  $\overline{\text{IOE}}$ , and  $\overline{\text{RD}}$  lines, see section 2 "Pin Assignments and Signal Descriptions."

Bit Name	7	6	5	4	3	2	1	0
	LIRE	$\overline{\text{LIRTE}}$	$\overline{\text{IOC}}$	—*	—*	—*	—*	—*
Read/Write	R/W	W	R/W	—	—	—	—	—
Initial Value	1	1	1	0	0	0	0	0

$\overline{\text{LIR}}$  Enable  
0: The  $\overline{\text{LIR}}$  output is low only during the opcode fetch cycle 2 of the RETI instruction and the first machine cycle of the  $\overline{\text{INT0}}$  interrupt acknowledge cycle.  
1: Normal operation

I/O Compatibility  
0: Output of the  $\overline{\text{IOE}}$  and  $\overline{\text{RD}}$  lines is compatible with that of the Z80-based peripheral LSIs.  
1: Normal operation

$\overline{\text{LIR}}$  Temporary Enable  
0: When the LIRE bit is 0, the  $\overline{\text{LIR}}$  output is low only for the opcode fetch cycle immediately after 0 is written to the  $\overline{\text{LIRTE}}$  bit.  
1: Normal operation

\* Reserved. These bits always read 0 and should be set to 0.

To interface with Z80-based peripheral LSIs, set the bits of the OMCR as shown in table 3-2.

**Table 3-2. OMCR Setting**

**Daisy-chained Interrupt from**

<b>Z80-based Peripheral LSIs</b>	<b>LIRE</b>
Used	0
Unused	0 or 1

<b>Z80 PIO</b>	<b><math>\overline{\text{LIRTE}}</math></b>
Used	0 enables interrupts from the Z80 PIO
Unused	Setting is unnecessary

<b>Z80 CTC</b>	<b><math>\overline{\text{IOC}}</math></b>
Used	0
Unused	0 or 1

**Bit 7: LIRE ( $\overline{\text{LIR}}$  enable)**

LIRE controls the  $\overline{\text{LIR}}$  pin output. This bit is set to 1 by a reset.

<b>LIRE</b>	<b>Function</b>
0	$\overline{\text{LIR}}$ disabled
1	$\overline{\text{LIR}}$ enabled

When disabled,  $\overline{\text{LIR}}$  is active (low) only during:

- The opcode fetch cycle 2 of a RETI instruction (see "RETI Instruction" at the end of this section)
- The first machine cycle of an  $\overline{\text{INT0}}$  interrupt acknowledge cycle

LIRE = 0 is used for interfacing with Z80-based peripherals that support daisy-chained interrupts.

When enabled,  $\overline{\text{LIR}}$  is active (low) during:

- Opcode fetch cycles
- The first machine cycle of an  $\overline{\text{NMI}}$  interrupt acknowledge cycle
- The first machine cycle of an  $\overline{\text{INT0}}$  interrupt acknowledge cycle

LIRE can be set 1 when daisy-chained interrupts are not supported.

**Bit 6:  $\overline{\text{LIRTE}}$  ( $\overline{\text{LIR}}$  Temporary Enable)**

The write-only  $\overline{\text{LIRTE}}$  bit is used LIRE can be set 1 when daisy-chained interrupts are not supported

to activate the  $\overline{\text{LIR}}$  pin output. This bit is used to enable interrupts from the Z80 PIO after internal control registers have been set. At this time the Z80 PIOs support daisy-chained interrupts. This bit always reads 1 and is set to 1 by a reset.

When  $\overline{\text{LIRTE}}$  is 0 (active),  $\overline{\text{LIR}}$  output depends on the state of the LIRE bit. When LIRE is 1,  $\overline{\text{LIR}}$  output is not affected. When LIRE is 0,  $\overline{\text{LIR}}$  output goes low only during the opcode fetch cycle immediately after 0 is written to the  $\overline{\text{LIRTE}}$  bit (see figure 3-12).

$\overline{\text{LIRTE}}$	Function
0	$\overline{\text{LIR}}$ goes low in next opcode fetch if LIRE = 0
1	$\overline{\text{LIR}}$ output enabled/disabled by LIRE

When clearing the  $\overline{\text{LIRTE}}$  bit, all interrupts must be disabled (interrupt enable flag IEF1 must be set to 0 by a DI instruction). The instruction that clears the  $\overline{\text{LIRTE}}$  bit must be followed by an instruction with a 1-byte opcode. Figure 3-11 shows an example.

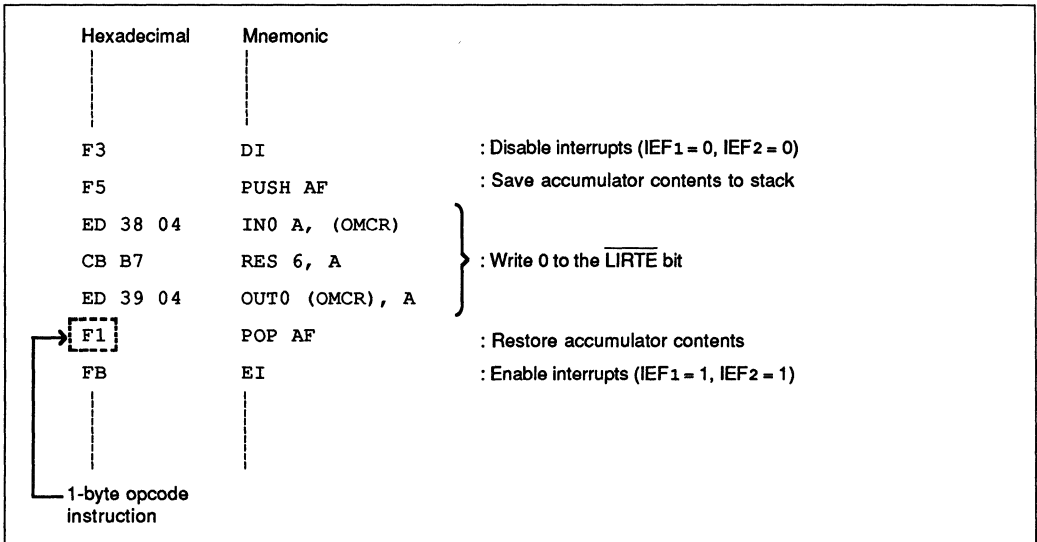


Figure 3-11. Example of Program for Clearing the  $\overline{\text{LIRTE}}$  Bit

Figure 3-12 shows the timing when the LIRE bit remains 0 and the  $\overline{\text{LIRTE}}$  bit is cleared.

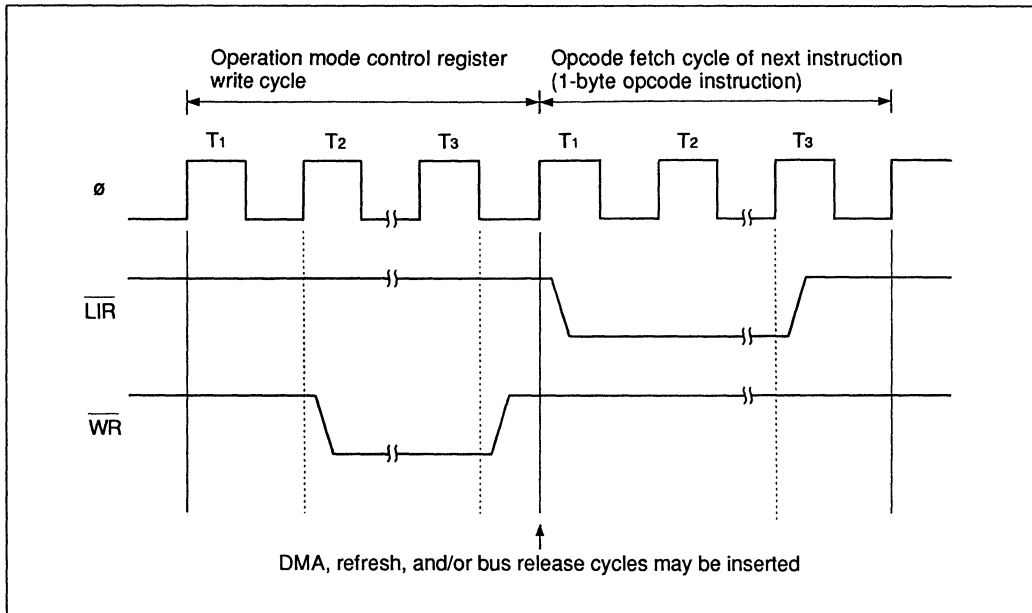


Figure 3-12. Timing When 0 is Written to  $\overline{\text{LIRTE}}$  Bit (with LIRE = 0)

### Bit 5: $\overline{\text{IOC}}$ (I/O Compatibility)

The  $\overline{\text{IOC}}$  bit controls the  $\overline{\text{IOE}}$  and  $\overline{\text{RD}}$  line outputs. When this bit is 0 during an I/O read or write cycle, the  $\overline{\text{IOE}}$  and  $\overline{\text{RD}}$  line outputs are compatible with those of Z80-based peripherals. The  $\overline{\text{IOC}}$  bit has no effect on the  $\overline{\text{RD}}$  line output during a memory read cycle or the  $\overline{\text{IOE}}$  line output during an  $\overline{\text{INT0}}$  interrupt acknowledge cycle. When  $\overline{\text{IOC}}$  is 0, the  $\overline{\text{IOE}}$  line output goes low at the rising edge of the T2 state during an I/O read or write cycle and the  $\overline{\text{RD}}$  line output also goes low at the rising edge of the T2 state during an I/O read cycle (see figure 3-13(a)).

The  $\overline{\text{IOC}}$  bit goes to 1 after a reset.

When  $\overline{\text{IOC}}$  is 1, the  $\overline{\text{IOE}}$  line output during an I/O read or write cycle and the  $\overline{\text{RD}}$  line output during an I/O read cycle go low at the falling edge of the T1 state (see figure 3-13(b)).

$\overline{\text{IOC}}$	Function
0	$\overline{\text{IOE}}$ and $\overline{\text{RD}}$ timing are Z80-compatible
1	$\overline{\text{IOE}}$ and $\overline{\text{RD}}$ timing are not Z80-compatible

Figure 3-13 shows the I/O read and write cycle timing.

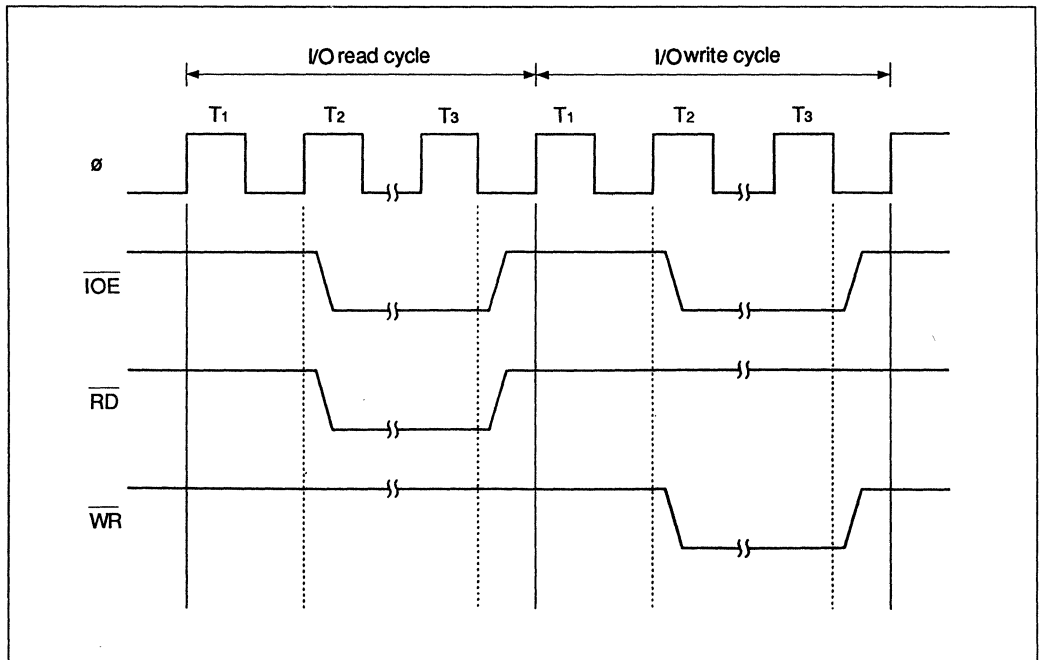


Figure 3-13.(a) I/O Read and Write Cycle Timing for  $\overline{\text{IOC}} = 0$

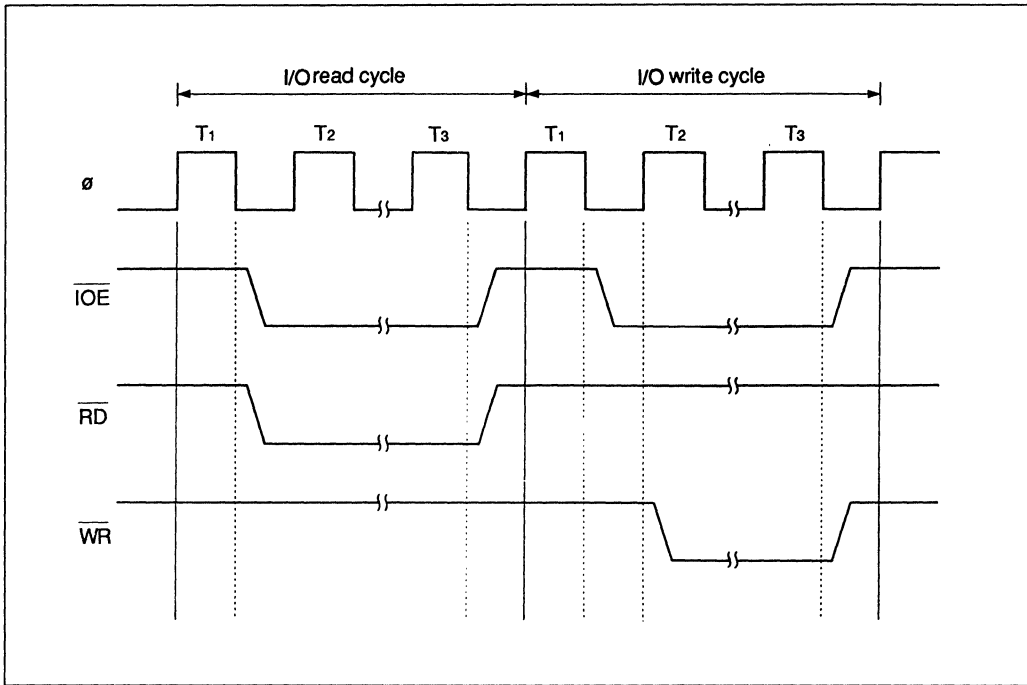


Figure 3-13.(b) I/O Read and Write Cycle Timing for  $\overline{IOC} = 1$

**Bits 4-0 :** Reserved. These bits always read 0 and should be set to 0.

**RETI Instruction:** In order to interface with Z80-based peripherals, the operation of the RETI instruction (return from  $\overline{INT0}$  interrupt) differs from other instructions as follows:

- The CPU fetches the 2-byte opcode (EDH, 4DH) twice. During the opcode fetch cycle 2, an internal operation ( $T_i$ ) cycle is inserted between the EDH and 4DH fetch cycles.
- If the LIRE bit is 1, the  $\overline{LIR}$  line goes low during both the opcode fetch cycle 1 and 2 in the RETI instruction.

If the LIRE bit is 0, the  $\overline{LIR}$  line goes low during the opcode fetch cycle 2, but remains high during the opcode fetch cycle 1.

The operations of the RETI and RET instructions are identical except that the RETI instruction, when placed at the end of an  $\overline{INT0}$  interrupt processing routine, allows a Z80-based peripheral to detect the end of the routine by decoding this instruction code.

Figure 3-14 shows the timing for a RETI instruction. Interrupt lines are not sampled at the end of the opcode fetch cycle 1 in the RETI instruction, rather, they are sampled 1.5 clock cycles before the end of the stack read cycle.



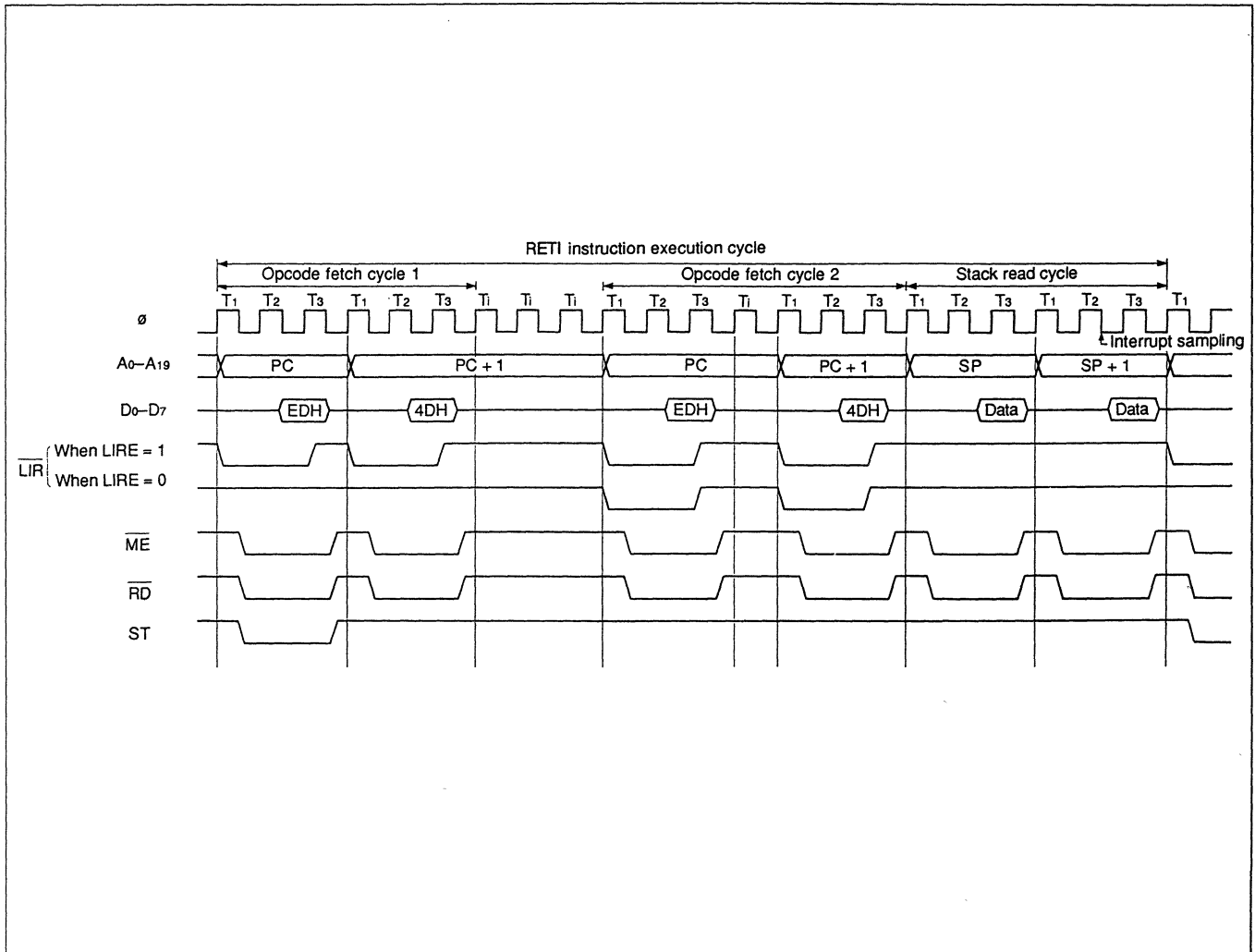


Figure 3-14. RETI Instruction Timing

Table 3-3 gives the bus cycle states for RETI instruction execution.

**Table 3-3. Bus Cycle States for RETI Instruction Execution**

	States			Address	Data	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
MC1	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	First opcode address	First opcode	0	1	0	1	0* 1	1	0
MC2	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	Second opcode address	Second opcode	0	1	0	1	0* 1	1	1
MC3 – MC5	T <sub>i</sub>	T <sub>i</sub>	T <sub>i</sub>	Undefined	Z	1	1	1	1	1	1	1
MC6	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	First opcode address	First opcode	0	1	0	1	0	1	1
MC7	T <sub>i</sub>			Undefined	Z	1	1	1	1	1	1	1
MC8	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	Second opcode address	Second opcode	0	1	0	1	0	1	1
MC9	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	SP	Data	0	1	0	1	1	1	1
MC10	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	SP + 1	Data	0	1	0	1	1	1	1

Z (data): High impedance

Note: \* The upper number indicates the  $\overline{LIR}$  line status when the LIRE bit = 1, and the lower number indicates the status when the LIRE bit = 0.

### 3.4 Chip Operation Modes

#### 3.4.1 Outline

The HD64180S supports five chip operation modes:

- Reset mode
  - Normal operation mode
  - Halt mode .....
  - Sleep mode.....
  - System stop mode.....
- } Low power dissipation modes
- } Special operation modes

The sleep and system stop modes are low power modes in which power dissipation is reduced.

The halt, sleep, and system stop modes are special operation modes in which the HD64180S's internal states differ from those in the normal operation mode.

The HD64180S can be placed into any other operation modes from the normal operation mode (see figure 3-15). The HD64180S can be returned to the normal operation mode from a special operation mode by an interrupt.

If the  $\overline{\text{RESET}}$  signal is held active (low) for 6 clock cycles or more in the normal or special operation mode, the HD64180S is placed in the reset mode. In this mode, the HD64180S is stopped completely. HD64180S operation is restarted in the normal operation mode when the  $\overline{\text{RESET}}$  signal is deactivated.

Table 3-4 lists the status of the functional blocks of the HD64180S in the various operation modes.

If the  $\overline{\text{BUSREQ}}$  signal is asserted in the normal or special operation modes, the HD64180S enters the bus release mode. When this mode is set, the HD64180S passes bus control to an external I/O device.

See section 3.5 "Bus Arbiter" for details concerning the bus release mode.

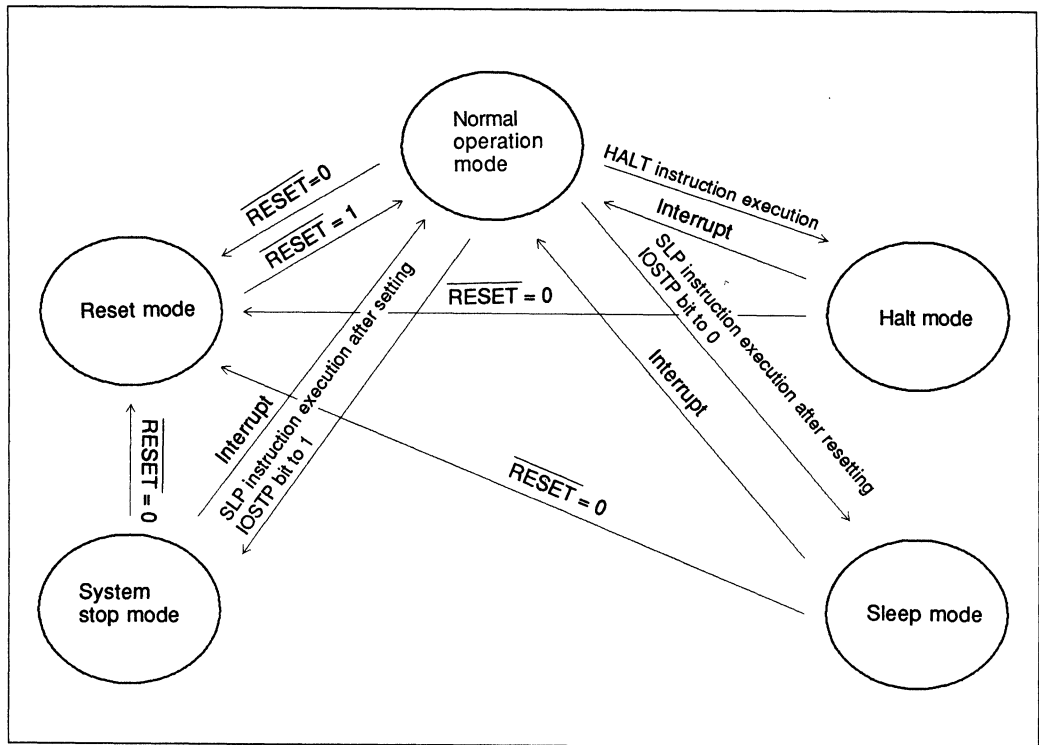


Figure 3-15. Operation Mode Transitions

Table 3-4. The Status of the Functional Blocks in the Various Operation Modes

Internal Function / Chip Operation Mode	CPU	Built-in DMAC	MSCI	ASCI/CSIO	Timer	Refresh Controller	Bus Release Mode
Reset mode	-----	-----	-----	-----	-----	-----	-----
Normal operation mode	○	○	○	○	○	○	○
Halt mode	○*	○	○	○	○	○	○
Sleep mode	-----	○	○	○	○	○	○
System stop mode	-----	-----	-----	-----	-----	-----	○

(○: Operation enabled, -----: Operation disabled)

\* Instructions are not executed (the CPU repeatedly fetches the instruction that follows the halt instruction).

Table 3-5 shows the register used to specify the low power dissipation mode.

**Table 3-5. Low Power Dissipation Mode Specification Register**

Register Name	Symbol	I/O Address	Initial Value*	Read/Write
			MSB ↔ LSB	
I/O control register	IOCR	0005H	00000000	R/W

\* The initial value is the value after a hardware reset.

### 3.4.2 Reset Mode

If the  $\overline{\text{RESET}}$  pin is held low for six or more clock cycles, all HD64180S functions are reset and the NPU enters the reset mode. In this mode, the HD64180S operates as follows.

- The CPU, MSC1, ASCI/CSIO, DMAC, refresh controller, and timer are halted and their internal states are reset.
- The A<sub>0</sub> – A<sub>19</sub> and D<sub>0</sub> – D<sub>7</sub> pins go to high impedance and all output pins are initialized to their predefined values.
- The on-chip oscillator continues to output the  $\phi$  clock.
- The CPU does not acknowledge external ( $\overline{\text{NMI}}$ ,  $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ ) or internal interrupts.
- The CPU does not acknowledge the  $\overline{\text{BUSREQ}}$  signal.

The  $\overline{\text{RESET}}$  line is sampled at the falling edge of every  $\phi$  clock. If the  $\overline{\text{RESET}}$  line is low for three successive cycles, the HD64180S enters the reset mode after a half clock cycle delay.

Note that although the HD64180S enters the reset mode after only three and a half clock cycles, normal reset operation is not guaranteed unless the  $\overline{\text{RESET}}$  is held low for six clock cycles (at the falling edge of the  $\phi$  clock).

The HD64180S will leave the reset mode once the  $\overline{\text{RESET}}$  line is deactivated. If the  $\overline{\text{RESET}}$  line remains high for three successive  $\phi$  clock falling edges, the HD64180S leaves the reset mode after a half clock cycle delay and resumes execution in the normal mode with an opcode fetch from logical address 0000H (physical address 00000H).

Figure 3-16 shows the timing for entering and leaving the reset mode.

Note: When the HD64180S enters the reset mode, the delay time for the A<sub>0</sub> – A<sub>19</sub> and D<sub>0</sub> – D<sub>7</sub> pins to go to high impedance from the  $\phi$  clock rising edge and the delay time for output pins  $\overline{\text{LIR}}$ ,  $\overline{\text{ME}}$ ,  $\overline{\text{IOE}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{WR}}$ ,  $\overline{\text{REF}}$ ,  $\overline{\text{HALT}}$ , and  $\overline{\text{BUSACK}}$  to go high from the  $\phi$  clock rising edge is 10  $\phi$  clock cycles maximum.

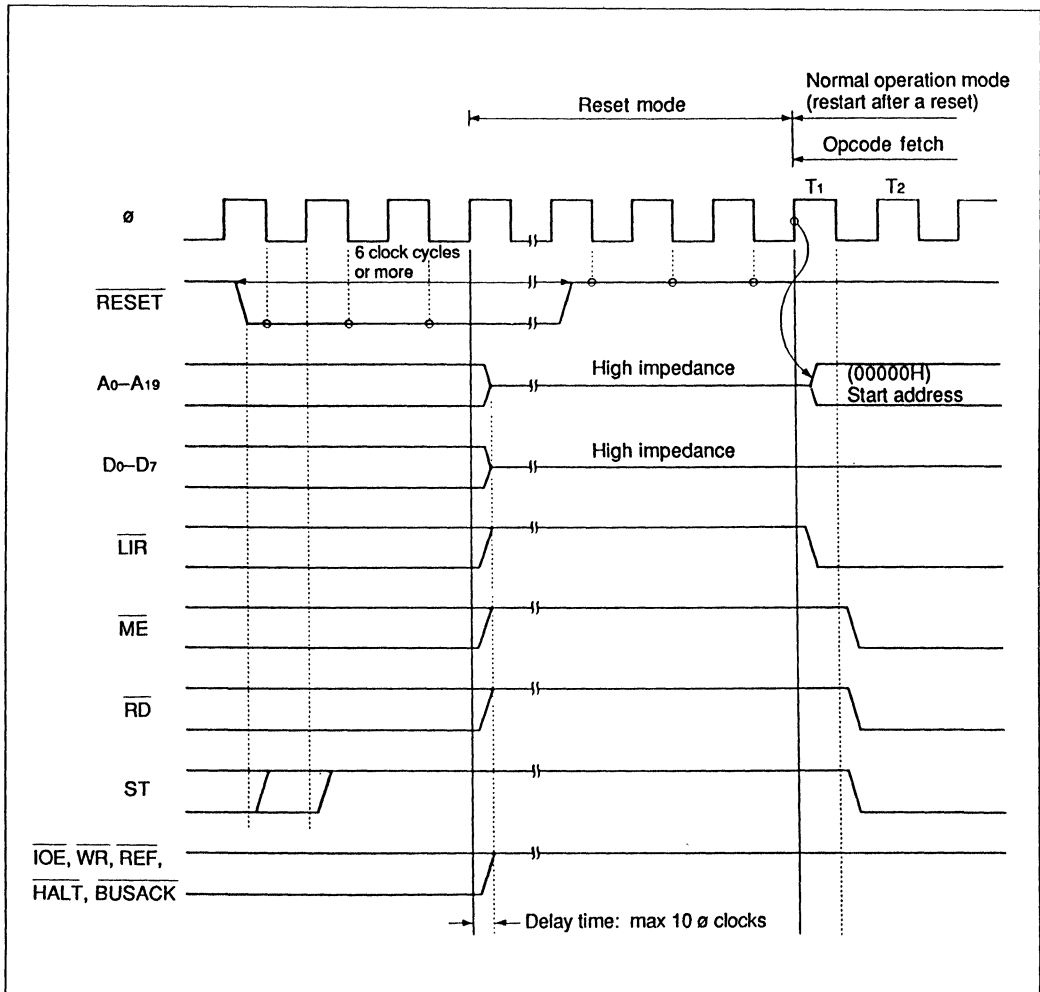


Figure 3-16. Reset Mode Timing

### 3.4.3 Normal Operation Mode

In the normal operation mode, the HD64180S operates as follows:

- The CPU fetches and executes instructions.
- The MSCI, ASCI/CSIO, DMAC, refresh controller, and timers are enabled.
- The on-chip oscillator continues to operate.
- The CPU accepts external ( $\overline{\text{NMI}}$ ,  $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ ) and internal interrupts.
- The CPU can be placed in the bus release mode by asserting the  $\overline{\text{BUSREQ}}$  signal.

The HD64180S can be placed into any of the other operation mode from the normal operation mode as follows:

- When the  $\overline{\text{RESET}}$  signal is held low for six or more clock cycles, the HD64180S enters the reset mode.
- When a HALT instruction is executed, the halt mode is entered.
- When an SLP instruction is executed while the IOSTP bit \* is 0, the sleep mode is entered.
- When an SLP instruction is executed while the IOSTP bit is 1, the system stop mode is entered.

\* Bit 7 of the I/O control register. This bit specifies the low power dissipation mode. For details, see section 3.4.5 "Sleep Mode."

From one of the special operation modes (halt, sleep, or system stop), the HD64180S can return to the normal operation mode following an interrupt. However, the type of interrupt that can be acknowledged and whether or not interrupt processing is performed depends on several conditions. For details, see sections 3.4.4 "Halt Mode," 3.4.5 "Sleep Mode," and 3.4.6 "System Stop Mode" below.

### 3.4.4 Halt Mode

Executing a HALT instruction (76H) in the normal operation mode causes the  $\overline{\text{HALT}}$  line to go low thus placing the HD64180S in the halt mode.

Internal operation in the halt mode is outlined below.

- The CPU internal clock does not stop. The CPU repeatedly fetches the instruction which follows the HALT instruction.
- The MSCI, ASCI/CSIO, DMAC, refresh controller, and timers continue operating.
- The on-chip oscillator continues to operate.

- The CPU accepts external ( $\overline{\text{NMI}}$ ,  $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ ) and internal interrupts.
- The CPU can be placed in the bus release mode by asserting the  $\overline{\text{BUSREQ}}$  signal (active low).

Operation in the halt mode is the same as operation in the normal mode, except that no instructions are executed. The HD64180S can leave the halt mode in two ways:

#### (1) Reset

If the  $\overline{\text{RESET}}$  signal is held low for six or more clock cycles, the HD64180S is reset and leaves the halt mode.

#### (2) Interrupt

If interrupt enable flag IEF1 is set to 1 (interrupt enabled) and either a maskable external ( $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ , or  $\overline{\text{INT2}}$ ) or internal interrupt occurs, the HD64180S leaves the halt mode and branches to the appropriate interrupt processing routine. If an  $\overline{\text{NMI}}$  interrupt occurs in the halt mode, the HD64180S leaves the halt mode and branches to the interrupt processing routine, regardless of the value of the IEF1 flag.

If an interrupt is requested 1.5 clock cycles before the end of the opcode fetch cycle for a HALT instruction, the HD64180S will not enter the halt mode. It branches to the interrupt processing routine after fetching the HALT instruction opcode.

Figure 3-17 shows the timing for entering the halt mode and leaving after an interrupt.



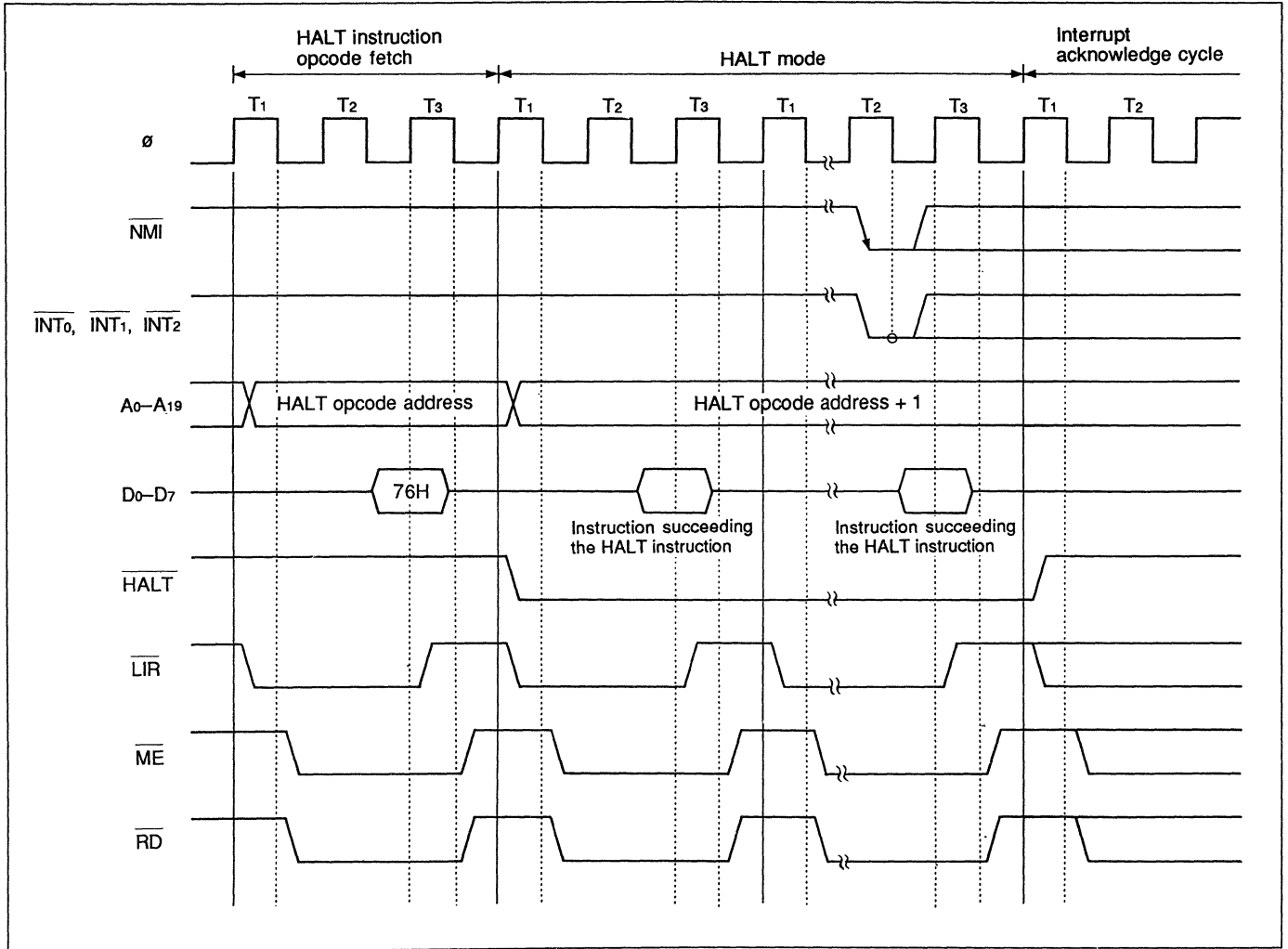


Figure 3-17. Timing for Entering Halt Mode and Leaving after an Interrupt

### 3.4.5 Sleep Mode

Executing an SLP instruction (EDH, 76H) while the IOSTP bit in the I/O control register is 0 causes the CPU internal clock to stop and the  $\overline{\text{HALT}}$  line to go low, thus placing the HD64180S in sleep mode. In this mode, power dissipation is reduced because the CPU internal clock is stopped.

#### I/O Control Register (IOCR)

The I/O control register is used in combination with a SLP instruction to specify the low power dissipation mode (sleep or system stop).

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

|  
I/O Stop  
 0: Sleep mode (SLP instruction execution)  
 1: System stop mode (SLP instruction execution)

\* Reserved. These bits always read 0 and should be set to 0.

#### Bit 7: IOSTP (I/O stop)

This bit specifies the low power dissipation mode.

IOSTP	Function
0	SLP instruction causes the HD64180S to enter the sleep mode
1	SLP instruction causes the HD64180S to enter the system stop mode

Setting value to IOSTP bit has no effect on the operation of HD64180S internal functions. The execution of SLP instruction causes the HD64180S to enter either the sleep mode or the system stop mode according to the value.

This bit is cleared by a reset.

**Bits 6-0:** Reserved. These bits always read 0 and should be set to 0.

Internal operations in sleep mode are as follows:

- The internal CPU clock stops and the CPU stops operating.
- The MSCI, ASCI/CSIO, DMAC, refresh controller, and timers continue operating.
- The on-chip oscillator continues operating.
- The CPU accepts external ( $\overline{\text{NMI}}$ ,  $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ ) and internal interrupts.
- Asserting the  $\overline{\text{BUSREQ}}$  line causes the CPU to enter bus release mode.

The HD64180S can leave sleep mode in the following two ways:

#### (1) Reset

If the  $\overline{\text{RESET}}$  signal is held active for six or more clock cycles, the HD64180S is reset and leaves the sleep mode.

#### (2) Interrupt

If either an external ( $\overline{\text{NMI}}$ ,  $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ ) or an internal interrupt occurs, the HD64180S leaves the sleep mode and enters normal mode. Unlike in the halt mode, the HD64180S leaves the sleep mode following an interrupt, even if the interrupt is disabled by  $\text{IEF}_1 = 0$ .

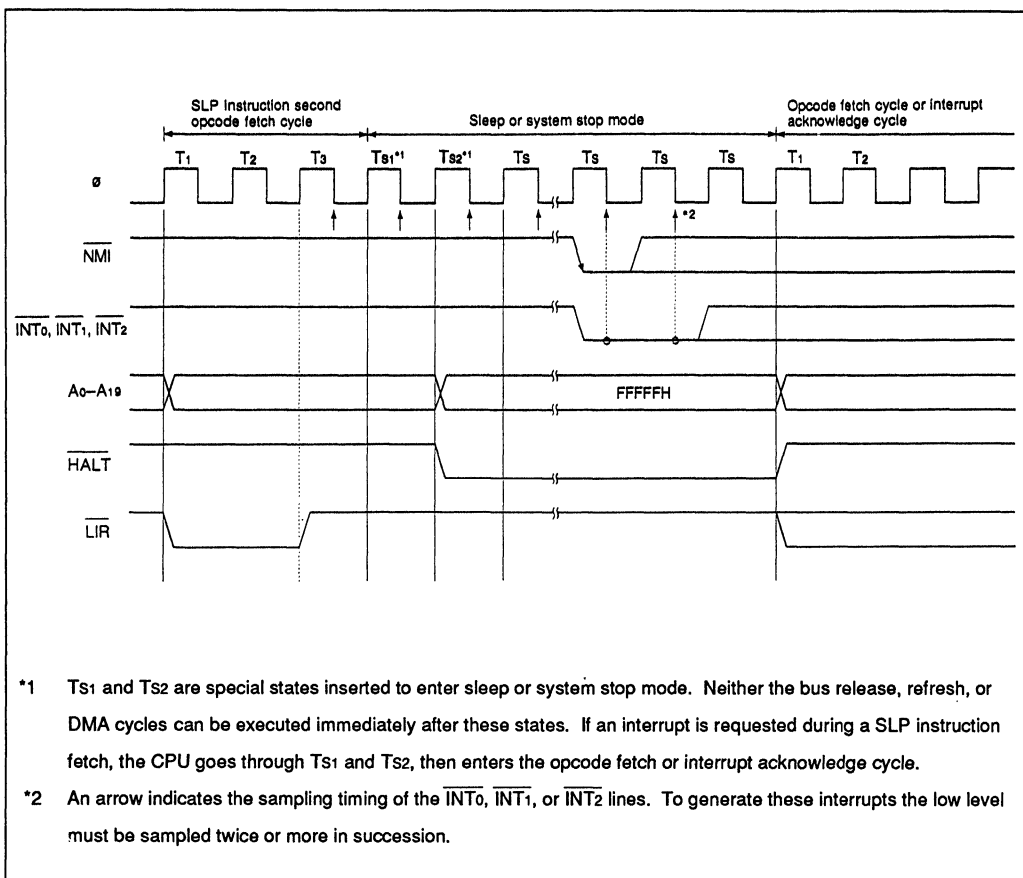
If interrupts are disabled by interrupt enable flag  $\text{IEF}_1$ , the CPU executes the instruction following the SLP instruction after leaving the sleep mode. If interrupts are enabled, the CPU branches to the corresponding interrupt processing routine.

In the sleep mode, level-sensitive external interrupts ( $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ , or  $\overline{\text{INT2}}$ ) are sampled at every falling edge of the  $\phi$  clock. If an external interrupt line is active when sampled, the HD64180S leaves the sleep mode after 2.5 clock cycles. Note that the external interrupt line must be active for at least two successive samples.

If an  $\overline{\text{NMI}}$  interrupt occurs in sleep mode, the HD64180S leaves the sleep mode and branches to the interrupt processing routine, regardless of the  $\text{IEF}_1$  flag.

If the falling edge of an  $\overline{\text{NMI}}$  signal is input before the falling edge of the  $\phi$  clock in the sleep mode, the HD64180S leaves the sleep mode 2.5 clock cycles after the  $\phi$  clock falling edge. After leaving the sleep mode, the CPU starts the  $\overline{\text{NMI}}$  acknowledge cycle.

Figure 3-18 shows the timing for entering the sleep mode and leaving after an interrupt.



**Figure 3-18. Sleep and System Stop Mode Timing**

### 3.4.6 System Stop Mode

Executing a SLP instruction while the IOSTP bit in the I/O control register is 1 causes the  $\overline{\text{HALT}}$  line to go low and the HD64180S to enter the system stop mode. In this mode less power is dissipated than in the sleep mode since the clock supplied to the CPU and other functional units is stopped. For details of the I/O control register, see section 3.4.5 "Sleep Mode."

Internal operation in system stop mode is as follows:

- The CPU, MSCI, ASCI/CSIO, DMAC, and timers stop.
- The refresh controller stops (the contents of DRAM are lost).
- The on-chip oscillator does not stop.
- The CPU accepts external interrupts ( $\overline{\text{NMI}}$ ,  $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ ).
- Asserting the  $\overline{\text{BUSREQ}}$  line causes the CPU to enter bus release mode.

The HD64180S can leave the system stop mode in the following two ways:

#### (1) Reset

If the  $\overline{\text{RESET}}$  line is held low for six or more clock cycles, the HD64180S is reset and leaves the system stop mode.

#### (2) Interrupt

An external interrupt ( $\overline{\text{NMI}}$ ,  $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ ) causes the HD64180S to leave the system stop mode and to enter the normal operation mode.

If the interrupt enable flag IEF1 is 0 and interrupts are disabled, the CPU resumes execution at the first instruction after the SLP instruction. If IEF1 is 1 and interrupts are enabled, the CPU branches to the corresponding interrupt processing routine.

If an  $\overline{\text{NMI}}$  interrupt occurs, the CPU branches to the interrupt processing routine regardless of the value of the IEF1 flag.

The timing for entering and leaving the system stop mode using an interrupt is the same as that in the sleep mode (see figure 3-18).

## 3.5 Bus Arbiter

### 3.5.1 Overview

The HD64180S is equipped with a bus arbiter which arbitrates bus contention between the on-chip CPU, DMAC, refresh controller (internal devices), and external I/O devices.

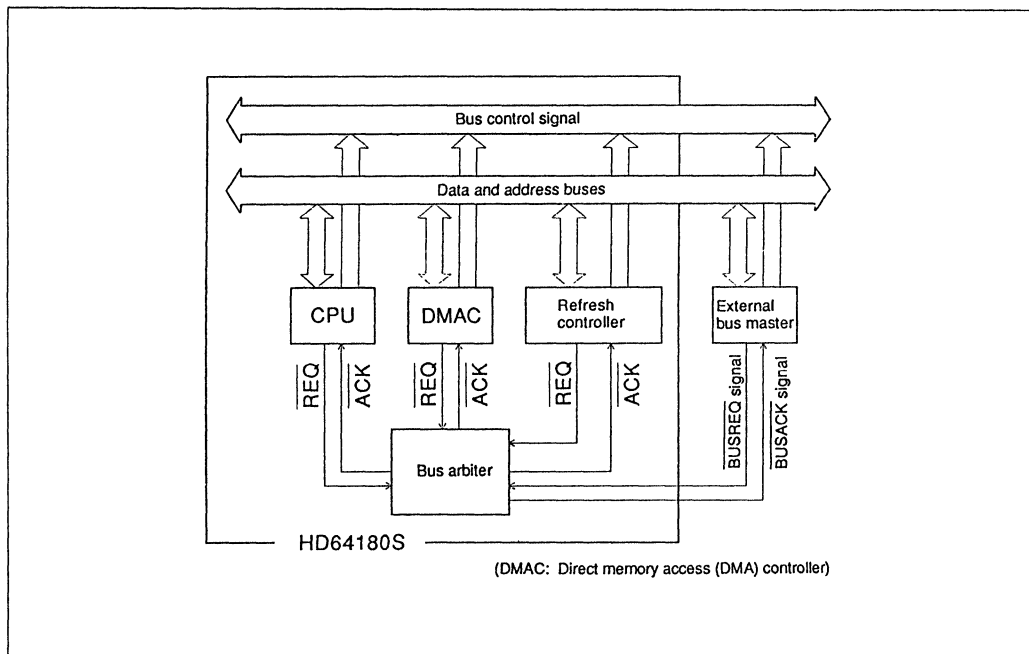


Figure 3-20. Bus Arbiter and Masters

As shown in figure 3-20, the on-chip CPU, DMAC, and refresh controller are internally connected to the arbiter via  $\overline{\text{REQ}}$  and  $\overline{\text{ACK}}$  lines. Internal devices use a request ( $\overline{\text{REQ}}$ ) signal to request bus control and the arbiter uses an acknowledge ( $\overline{\text{ACK}}$ ) line to acknowledge that a particular device has been given control of the bus.

External I/O devices are connected to the arbiter via the external  $\overline{\text{BUSREQ}}$  and  $\overline{\text{BUSACK}}$  signals.  $\overline{\text{BUSREQ}}$  and  $\overline{\text{BUSACK}}$  function in the same way as the  $\overline{\text{REQ}}$  and  $\overline{\text{ACK}}$  signals do for internal devices.

The bus arbiter constantly samples the bus request signals ( $\overline{\text{REQ}}$  and  $\overline{\text{BUSREQ}}$ ) looking for one of the lines to go low. Once the arbiter detects a bus request signal, it passes control of the bus to the appropriate device by asserting its acknowledge line ( $\overline{\text{ACK}}$  or  $\overline{\text{BUSACK}}$ ). The device has control until the acknowledge line is deactivated (goes high).

If bus control is requested from more than one device at the same time, the bus arbiter passes control to the device with the highest priority.

Devices are given the following priority:

External bus master > refresh controller > DMAC > CPU.

Note: The term "bus master" is used in this manual to refer to a CPU, internal I/O device, or external I/O device which already has or can request the bus control.

### 3.5.2 Timing for Passing Bus Control

The priority assignment is only used when several devices request control at the same time. Depending on the devices involved, bus control will not necessarily be passed immediately to a higher-priority device as soon as it requests bus control. Various contention situations are discussed in this section.

#### (1) When the CPU has control

When the CPU has control of the bus and a request is received from another device via internal  $\overline{\text{REQ}}$  lines or  $\overline{\text{BUSREQ}}$  line, the arbiter can pass control at the end of any machine cycle (immediately after a T3 or Ti state). An opcode fetch, memory read/write, or I/O read/write machine cycle (3  $\emptyset$  clock cycles without wait states) is equivalent to one machine cycle. For internal CPU operations (Ti states), one state (1  $\emptyset$  clock cycle) is equivalent to one machine cycle.

• The arbiter cannot pass bus control immediately after the following states:

- ① Second machine cycle (Ti state) of a DJNZ j instruction.
- ② TRP state of a TRAP acknowledge cycle when the second opcode is undefined.
- ③ Ts1 or Ts2 state at the start of the sleep or system stop mode.

#### (2) When the refresh controller has control

When the refresh controller has control of the bus, the arbiter can pass control at the end of each machine cycle (immediately after a Tr2 state). One machine cycle consists of two states if no wait states are inserted.

#### (3) When the DMAC has control

When the DMAC has control of the bus, the arbiter can pass control at the end of each machine cycle (immediately after a T3 or Ti state). For example, during a DMA transfer in burst mode, when a device having a priority higher than the DMAC requests control, the DMAC suspends the transfer at the end of the current machine cycle and releases control. When the higher priority device releases the bus (by placing the bus request signal high), the DMAC is given control and resumes the transfer.



- The arbiter cannot pass bus control immediately after the following states. See section 6 "Direct Memory Access Controller (DMAC)" for details.

- ①  $T_i$  state immediately before the DMA transfer start for the first byte.
- ② Immediately after a read cycle during dual-address DMA transfer.
- ③ Immediately after a part of  $T_3$  or  $T_i$  states during buffer chaining operations.

(4) When an external I/O device is the bus master.

For details, see section 3.5.3 "Bus Release Mode," below.

### 3.5.3 Bus Release Mode

When an external I/O device obtains control of the bus, the HD64180S enters the bus release mode. Since the  $\overline{\text{BUSREQ}}$  signal has the highest priority, when this signal is asserted, bus control is passed to the device requesting the bus. The address lines ( $A_0 - A_{19}$ ), data lines ( $D_0 - D_7$ ), and bus control lines ( $\overline{\text{ME}}$ ,  $\overline{\text{IOE}}$ ,  $\overline{\text{RD}}$ , and  $\overline{\text{WR}}$ ) go to the high impedance state.

In the bus release mode, the HD64180S does not perform refresh operations. These must be performed by the external I/O device. The CPU cannot accept external or internal interrupts in the bus release mode.

The HD64180S leaves the bus release mode when the external device releases the bus by deactivating the  $\overline{\text{BUSREQ}}$  signal or when the  $\overline{\text{RESET}}$  line is held low for six or more clock cycles. When the external device releases control, it is returned to the internal bus master. If a reset occurs, bus control is returned to the CPU when the  $\overline{\text{RESET}}$  signal goes high.

Figure 3-21 (a) shows the timing when bus control is requested by an external I/O device via the  $\overline{\text{BUSREQ}}$  line during a CPU memory read cycle. Figure 3-21 (b) shows the timing when the bus control is requested during a CPU internal operation.

The  $\overline{\text{BUSREQ}}$  signal is sampled at the falling edge of the  $\phi$  clock in the state preceding the  $T_3$ ,  $T_i$ ,  $T_X$  (bus release state),  $TR_2$  (refresh cycle end state), or  $T_s$  state (sleep or system stop mode). If the  $\overline{\text{BUSREQ}}$  pin is low at the falling edge of the  $\phi$  clock in the state preceding the  $T_X$  state, the  $T_X$  state is repeated.

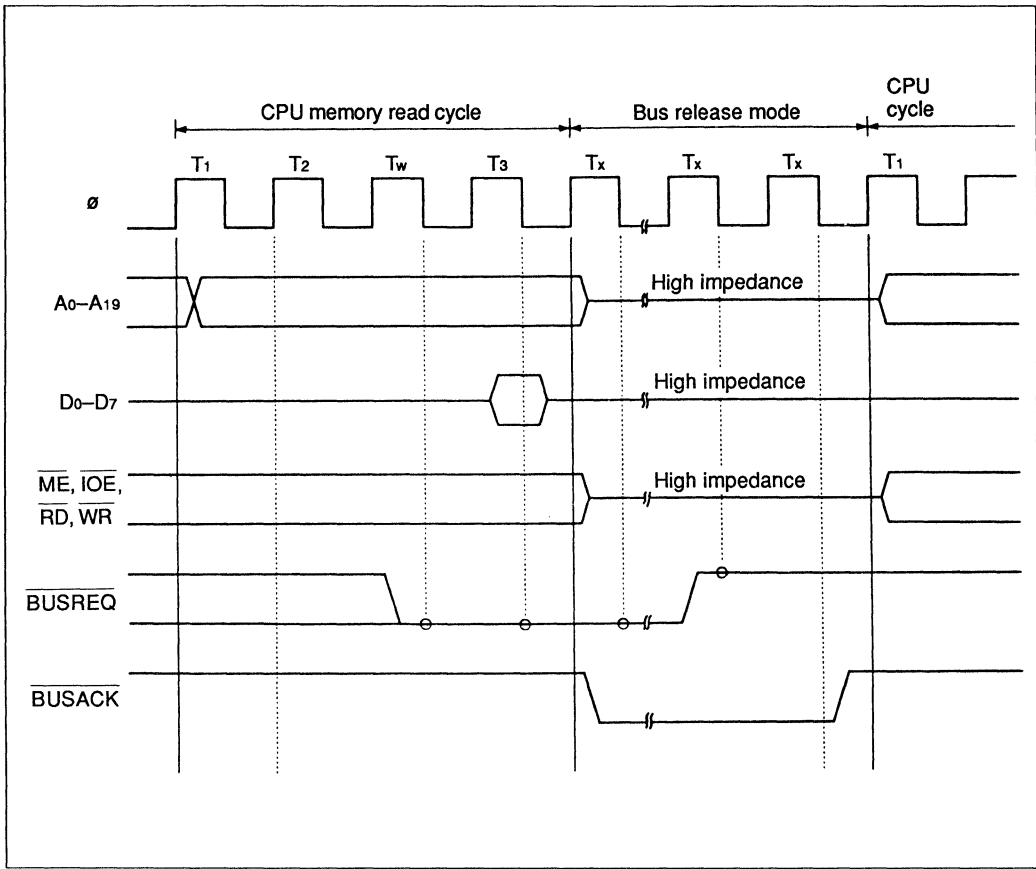


Figure 3-21. (a) Bus Release Mode (1)

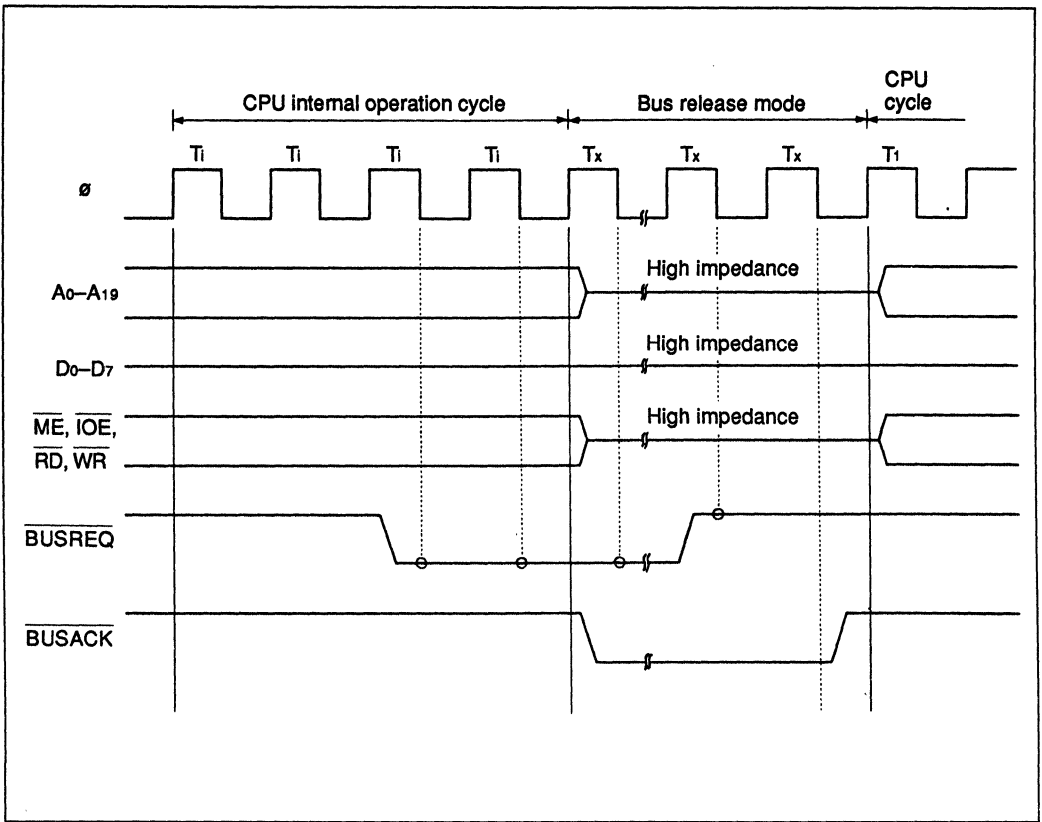


Figure 3-21. (b) Bus Release Mode (2)

### 3.5.4 Bus Control Passing

Figure 3-22 shows how bus control is passed. Bus control requests have the following priority:

$\overline{\text{BUSREQ}} = 0 > \text{refresh request} > \text{DMA request}$

When bus control is not requested by an external I/O device, refresh controller, or DMAC, it returns to the CPU.

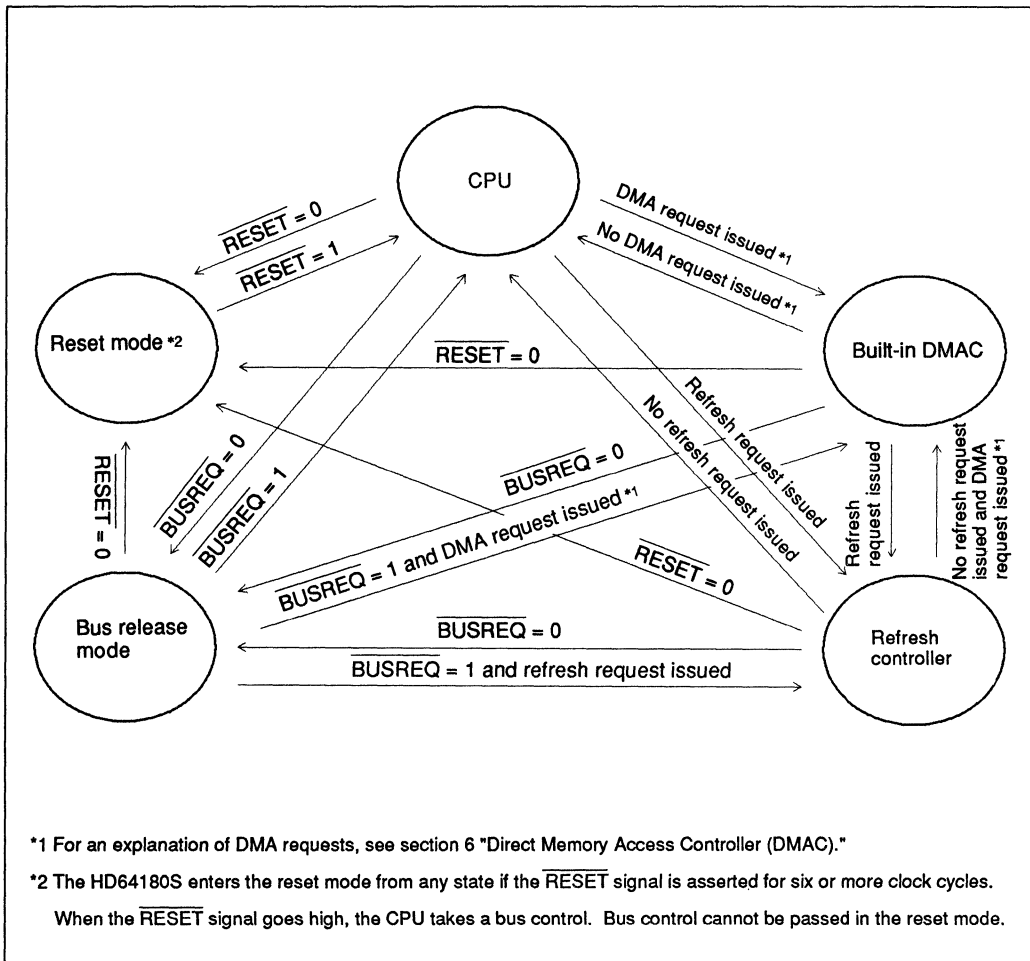


Figure 3-22. Bus Control Passing

## 3.6 Interrupts

### 3.6.1 Overview

The HD64180S supports 4 external and 15 internal interrupts. External interrupts have higher priority over internal interrupts except for TRAP. Figure 3-23 shows which interrupts are requested by various internal and external devices.

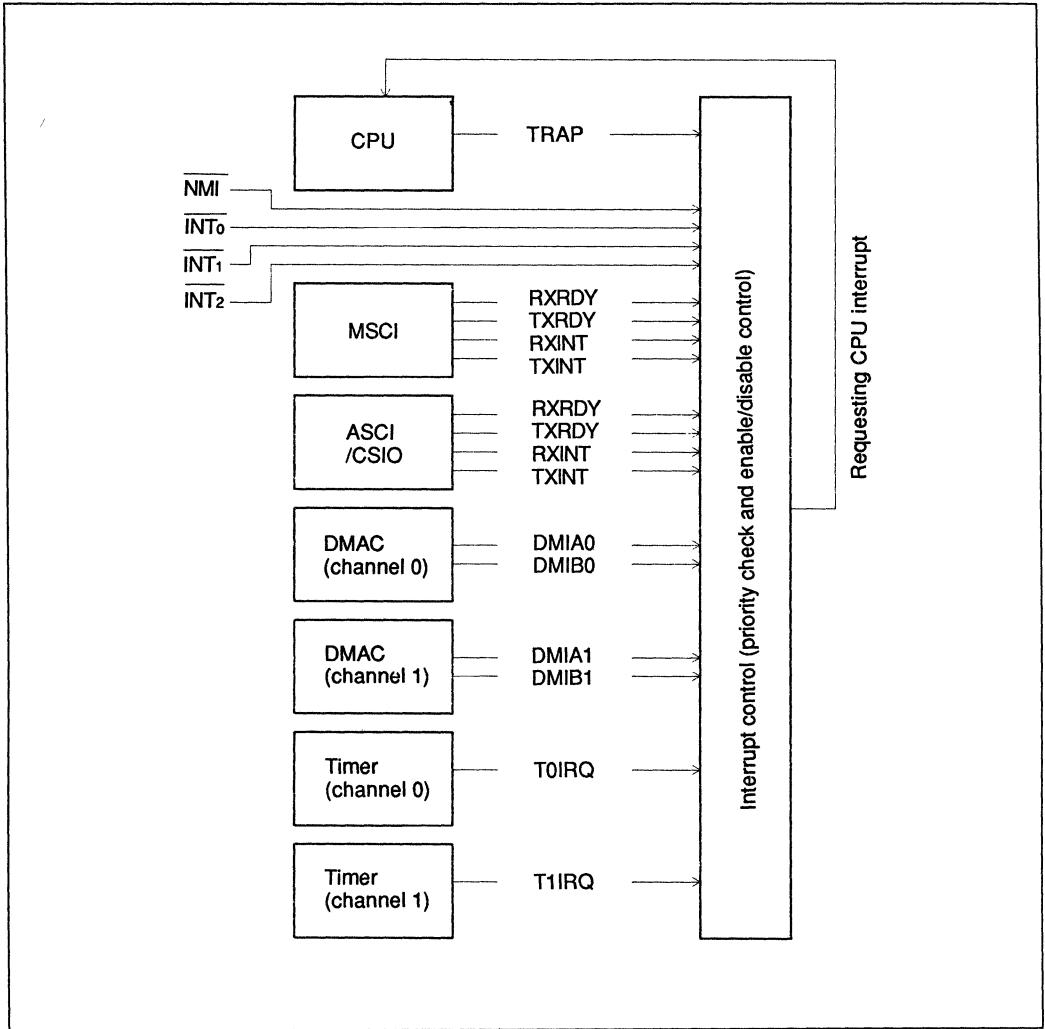




Figure 3-23. Interrupt Block Diagram

Table 3-6 lists interrupts by priority and indicates the source.

**Table 3-6. Interrupt Types, Priorities, and Sources**

Priority	Interrupts	Interrupt source	Internal/External
High                 Low 	TRAP	Undefined opcode fetch	Internal interrupt
	$\overline{\text{NMI}}$	} Line input	
	$\overline{\text{INT0}}$		
	$\overline{\text{INT1}}$		
	$\overline{\text{INT2}}$		
	MSCI RXRDY	Receive buffer contains data.	} Internal interrupts
	MSCI TXRDY	Transmit buffer is empty or not full.	
	MSCI RXINT	Receive status	
	MSCI TXINT	Transmit status	
	ASCI/CSIO RXRDY	Receive buffer contains data.	
	ASCI/CSIO TXRDY	Transmit buffer is empty.	
	ASCI/CSIO RXINT	Receive status	
	ASCI/CSIO TXINT	Transmit status	
	DMAC Channel 0 DMIA0	Error interrupt	
	DMAC Channel 0 DMIB0	Normal termination interrupt	
	DMAC Channel 1 DMIA1	Error interrupt	
DMAC Channel 1 DMIB1	Normal termination interrupt		
Timer Channel 0 TOIRQ	Count match		
Timer Channel 1 T1IRQ	Count match		

When an interrupt (except TRAP,  $\overline{\text{NMI}}$ , and  $\overline{\text{INT0}}$ ) is requested, the request status is indicated by interrupt status registers 0 and 1. If the requested interrupt has been enabled by interrupt enable registers 0 or 1, an interrupt request is issued to the CPU.

### 3.6.2 Interrupt Control Registers and Interrupt Enable Flags

The HD64180S has seven interrupt control registers (see table 3-7) and two interrupt enable flags. These registers are mapped in the internal I/O address space and can be accessed by CPU I/O instructions. This does not apply to the interrupt vector register (I), however. This register can be accessed by LD A, I and LD I, A instructions.

**Table 3-7. Interrupt Control Registers**

Register Name	Symbol	I/O Address	Initial Value*1	
			MSB ↔ LSB	Read/Write
Interrupt vector register	I	—	00000000	R/W
Interrupt vector low register	IL	0014H	00000000	R/W
Interrupt control register	ICR	0000H	00000001	R/W
Interrupt status register 0	ISR0	0010H	000000XX*2	R
Interrupt status register 1	ISR1	0011H	00000000	R
Interrupt enable register 0	IER0	0012H	00000000	R/W
Interrupt enable register 1	IER1	0013H	00000000	R/W

\*1 "Initial value" is the value after a hardware reset.

\*2 x: Undefined

**Interrupt Vector Register (I):** The interrupt vector register indicates the vector table position in memory that stores the starting address of the interrupt processing routine for  $\overline{INT}0$  (mode 2),  $\overline{INT}1$ ,  $\overline{INT}2$ , or internal interrupts (vector mode).

	7	6	5	4	3	2	1	0
Bit Name	17	16	15	14	13	12	11	10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

└───┬──────────────────────────────────┘

The eight high-order bits of the vector address

The 16-bit interrupt vector indicates the starting address of the interrupt processing routine in the vector table. The low order byte is the data placed on the bus, in  $\overline{INT}0$  mode 2, or the vector generated for each interrupt source. The high order byte is the I register contents. Vector tables can be generated at arbitrary memory locations at 256-byte intervals by setting the I register. Use LD A, I and LD I, A instructions to access this register.

**Interrupt Vector Low Register (IL):** The 8-bit interrupt vector low register specifies the low order byte of the 16-bit interrupt vector for vector interrupts ( $\overline{INT}1$ ,  $\overline{INT}2$ , and internal interrupts except TRAP) other than  $\overline{INT}0$  mode 2. The interrupt vector register (I) specifies the high-order byte.

The low order six bits of this register are automatically set depending on the source of the interrupt. Bits 7 and 6 (IL7, 6) can be set arbitrarily by software.

	7	6	5	4	3	2	1	0
Bit Name	IL7	IL6	-	-	-	-	-	-
Read/Write	R/W	R/W	-	-	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0

Fixed code\*

---

Low order byte of vector address

\* The fixed code (bits 5 – 0) cannot be read or written. For the fixed code, see table 3-9 "Interrupt Sources and Vectors."

This register allows interrupt vector tables to be placed at any 64-byte boundary in system memory. Bits 7 and 6 are set to 0 after a reset. Bits 5 – 0 always read 0 and should be set to 0.

Since the interrupt vector register (I) is also initialized to 00H by a reset, the 16-bit interrupt vector address becomes 0000 0000 00XX XXXX. For example, the  $\overline{INT}1$  fixed code is 00 0000, so its associated interrupt vector address after a reset is 0000 0000 0000 0000.

**Interrupt Control Register (ICR):** The interrupt control register tests whether a TRAP interrupt has occurred and specifies whether the  $\overline{INT}0$  interrupt is enabled or disabled.

	7	6	5	4	3	2	1	0
Bit Name	TRAP	UFO	-*	-*	-*	-*	-*	ITE0
Read/Write	R/W	R	-	-	-	-	-	R/W
Initial Value	0	0	0	0	0	0	0	1

TRAP Status

0: TRAP interrupt not generated

1: TRAP interrupt generated

Undefined Fetch Object Code

0: Second byte of opcode undefined

1: Third byte of opcode undefined

$\overline{INT}0$  Enable

0:  $\overline{INT}0$  disabled

1:  $\overline{INT}0$  enabled

\* Reserved. These bits always read 0 and should be set to 0.

### Bit 7: TRAP (TRAP status)

The TRAP bit indicates whether a TRAP interrupt has occurred. This bit is cleared by writing 0 to this bit position by CPU. A 1 cannot be written to this bit. It is cleared to 0 by a reset.



**Note:** A TRAP interrupt is an internal interrupt having the highest priority generated when an undefined opcode is fetched during an opcode fetch cycle.

<b>TRAP</b>	<b>Functions</b>
0	A TRAP interrupt has not occurred
1	A TRAP interrupt has occurred

#### **Bit 6: UFO (Undefined fetch object code)**

The UFO bit indicates which opcode is undefined after a TRAP interrupt occurs.

When the TRAP bit is 0, the UFO bit is cleared during the first byte of opcode fetch and set to 1 again during the third byte of the opcode fetch. When a TRAP interrupt is generated and the TRAP bit is set to 1, the UFO bit will not be updated by subsequent opcode fetches. Thus it is possible, by reading the UFO bit, to determine whether the undefined opcode occurred during the second or third byte. Note that there are no undefined opcodes for the first byte.

Writing to this bit position has no effect, thus either 0 or 1 can be written in this bit when it is necessary to write in the interrupt control register. This bit is cleared to 0 by a reset.

<b>UFO</b>	<b>Functions</b>
0	Second opcode is undefined
1	Third opcode is undefined

**Bits 5 – 1:** Reserved. These bits always read 0 and should be set to 0.

#### **Bit 0: ITE0 ( $\overline{\text{INT0}}$ enable)**

The ITE0 bit specifies whether  $\overline{\text{INT0}}$  interrupts are enabled or disabled. This bit is set to 1 by a reset.

ITE0	Functions
0	Disables $\overline{\text{INT0}}$ interrupts
1	Enables $\overline{\text{INT0}}$ interrupts

**Interrupt Status Register 0 (ISR0):** The read-only interrupt status register 0 indicates whether  $\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ , or internal interrupts (except TRAP) requests have been issued. Bits 1 and 0 of this register indicate the levels of the  $\overline{\text{INT1}}$  and  $\overline{\text{INT2}}$  pins even after a reset. Bits 7 – 2 are cleared to 0 by a reset.

	7	6	5	4	3	2	1	0
Bit Name	TXRDY1	RXRDY1	TXINT0	RXINT0	TXRDY0	RXRDY0	INT2	INT1
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	X*	X*
	<u>ASCI/CSIO TXRDY</u> 0: Request not issued 1: Request issued		<u>MSCI TXINT</u> 0: Request not issued 1: Request issued		<u>MSCI TXRDY</u> 0: Request not issued 1: Request issued		<u>External Interrupt INT2</u> 0: Request not issued 1: Request issued	
		<u>ASCI/CSIO RXRDY</u> 0: Request not issued 1: Request issued		<u>MSCI RXINT</u> 0: Request not issued 1: Request issued		<u>MSCI RXRDY</u> 0: Request not issued 1: Request issued		<u>External Interrupt INT1</u> 0: Request not issued 1: Request issued

\* Undefined.

#### Bit 7: TXRDY1 (ASCI/CSIO TXRDY)

TXRDY1	Functions
0	TXRDY internal interrupt has not been generated by the ASCI/CSIO
1	TXRDY internal interrupt has been generated by the ASCI/CSIO

**Bit 6: RXRDY1 (ASCI/CSIO RXRDY)**

<b>RXRDY1</b>	<b>Functions</b>
0	RXRDY internal interrupt has not been generated by the ASCI/CSIO
1	RXRDY internal interrupt has been generated by the ASCI/CSIO

**Bit 5: TXINT0 (MSCI TXINT)**

<b>TXINT0</b>	<b>Functions</b>
0	TXINT internal interrupt has not been generated by the MSCI
1	TXINT internal interrupt has been generated by the MSCI

**Bit 4: RXINT0 (MSCI RXINT)**

<b>RXINT0</b>	<b>Functions</b>
0	RXINT internal interrupt has not been generated by the MSCI
1	RXINT internal interrupt has been generated by the MSCI

**Bit 3: TXRDY0 (MSCI TXRDY)**

<b>TXRDY0</b>	<b>Functions</b>
0	TXRDY internal interrupt has not been generated by the MSCI
1	TXRDY internal interrupt has been generated by the MSCI

**Bit 2: RXRDY0 (MSCI RXRDY)**

<b>RXRDY0</b>	<b>Functions</b>
0	RXRDY internal interrupt has not been generated by the MSCI
1	RXRDY internal interrupt has been generated by the MSCI

**Bit 1: INT2 (External Interrupt  $\overline{\text{INT}}_2$ )**

<b>INT2</b>	<b>Functions</b>
0	$\overline{\text{INT}}_2$ external interrupt has not been generated
1	$\overline{\text{INT}}_2$ external interrupt has been generated

**Bit 0: INT1 (External Interrupt INT1)**

INT1	Functions
0	$\overline{\text{INT1}}$ external interrupt has not been generated
1	$\overline{\text{INT1}}$ external interrupt has been generated

**Interrupt Status Register 1 (ISR1):** The read-only interrupt status register 1 indicates the status of internal interrupt. This register is cleared to 0 by a reset.

	7	6	5	4	3	2	1	0
Bit Name	T1IRQ	T0IRQ	DMIB1	DMIA1	DMIB0	DMIA0	TXINT1	RXINT1
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0
	<u>Timer Channel 1 Interrupt Request</u> 0: Request not issued 1: Request issued	<u>DMA Interrupt B Channel 1</u> 0: Request not issued 1: Request issued	<u>DMA Interrupt B Channel 0</u> 0: Request not issued 1: Request issued	<u>DMA Interrupt A Channel 1</u> 0: Request not issued 1: Request issued	<u>DMA Interrupt A Channel 0</u> 0: Request not issued 1: Request issued	<u>ASCII/CSIO TXINT</u> 0: Request not issued 1: Request issued	<u>ASCII/CSIO RXINT</u> 0: Request not issued 1: Request issued	

**Bit 7: T1IRQ (Timer Channel 1 Interrupt Request)**

T1IRQ	Functions
0	Timer channel 1 internal interrupt (T1IRQ) has not been generated
1	Timer channel 1 internal interrupt (T1IRQ) has been generated

**Bit 6: TOIRQ (Timer Channel 0 Interrupt Request)**

<b>TOIRQ</b>	<b>Functions</b>
0	Timer channel 0 internal interrupt (TOIRQ) has not been generated
1	Timer channel 0 internal interrupt (TOIRQ) has been generated

**Bit 5: DMIB1 (DMA Interrupt B Channel 1)**

<b>DMIB1</b>	<b>Functions</b>
0	DMAC channel 1 internal interrupt (DMIB) has not been generated
1	DMAC channel 1 internal interrupt (DMIB) has been generated

**Bit 4: DMIA1 (DMA Interrupt A Channel 1)**

<b>DMIA1</b>	<b>Functions</b>
0	DMAC channel 1 internal interrupt (DMIA) has not been generated
1	DMAC channel 1 internal interrupt (DMIA) has been generated

**Bit 3: DMIB0 (DMA Interrupt B Channel 0)**

<b>DMIB0</b>	<b>Functions</b>
0	DMAC channel 0 internal interrupt (DMIB) has not been generated
1	DMAC channel 0 internal interrupt (DMIB) has been generated

**Bit 2: DMIA0 (DMA Interrupt A Channel 0)**

<b>DMIA0</b>	<b>Functions</b>
0	DMAC channel 0 internal interrupt (DMIA) has not been generated
1	DMAC channel 0 internal interrupt (DMIA) has been generated

**Bit 1: TXINT1 (ASCI/CSIO TXINT)**

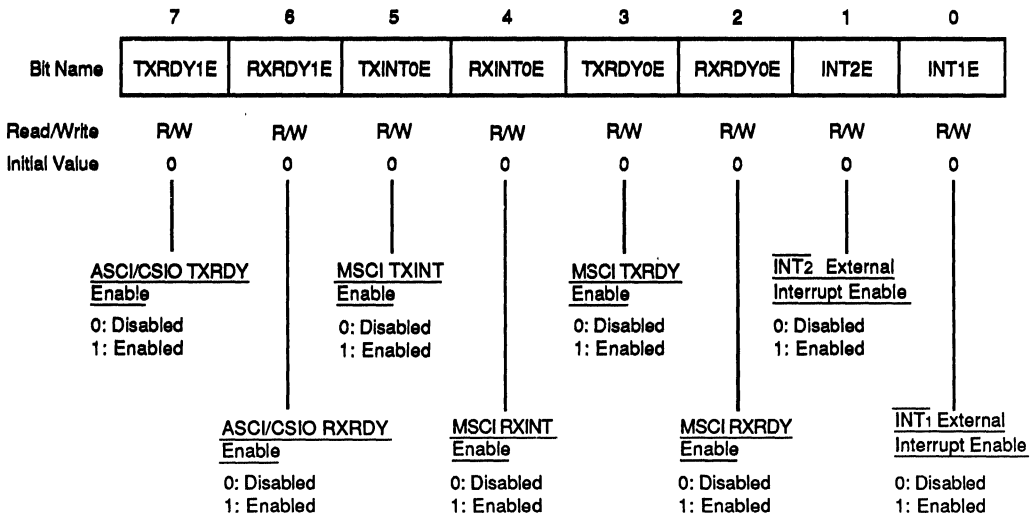
<b>TXINT1</b>	<b>Functions</b>
0	TXINT internal interrupt has not been generated by the ASCI/CSIO
1	TXINT internal interrupt has been generated by the ASCI/CSIO

## Bit 0: RXINT1 (ASCI/CSIO RXINT)

### RXINT1 Functions

0	RXINT internal interrupt has not been generated by the ASCI/CSIO
1	RXINT internal interrupt has been generated by the ASCI/CSIO

**Interrupt Enable Register 0 (IER0):** Interrupt enable register 0 specifies whether to enable the interrupt requests from interrupt status register 0 (ISRO). This register is cleared to 0 by a reset.



## Bit 7: TXRDY1E (ASCI/CSIO TXRDY Enable)

### TXRDY1E Functions

0	Disables TXRDY internal interrupts from the ASCI/CSIO
1	Enables TXRDY internal interrupts from the ASCI/CSIO

## Bit 6: RXRDY1E (ASCI/CSIO RXRDY Enable)

### RXRDY1E Functions

0	Disables RXRDY internal interrupts from the ASCI/CSIO
1	Enables RXRDY internal interrupts from the ASCI/CSIO

**Bit 5: TXINT0E (MSCI TXINT Enable)**

TXINT0E	Functions
0	Disables TXINT internal interrupts from the MSCI
1	Enables TXINT internal interrupts from the MSCI

**Bit 4: RXINT0E (MSCI RXINT Enable)**

RXINT0E	Functions
0	Disables RXINT internal interrupts from the MSCI
1	Enables RXINT internal interrupts from the MSCI

**Bit 3: TXRDY0E (MSCI TXRDY Enable)**

TXRDY0E	Functions
0	Disables TXRDY internal interrupts from the MSCI
1	Enables TXRDY internal interrupts from the MSCI

**Bit 2: RXRDY0E (MSCI RXRDY Enable)**

RXRDY0E	Functions
0	Disables RXRDY internal interrupts from the MSCI
1	Enables RXRDY internal interrupts from the MSCI

**Bit 1: INT2E ( $\overline{\text{INT}}_2$  External Interrupt Enable)**

INT2E	Functions
0	Disables $\overline{\text{INT}}_2$ external interrupts
1	Enables $\overline{\text{INT}}_2$ external interrupts

**Bit 0: INT1E ( $\overline{\text{INT}}_1$  External Interrupt Enable)**

INT1E	Functions
0	Disables $\overline{\text{INT}}_1$ external interrupts
1	Enables $\overline{\text{INT}}_1$ external interrupts

**Interrupt Enable Register 1 (IER1):** Interrupt enable register 1 specifies whether to enable interrupt requests from interrupt status register 1. This register is cleared to 00H by a reset.

Bit Name	7	6	5	4	3	2	1	0
	T1IRQE	T0IRQE	DMIB1E	DMIA1E	DMIB0E	DMIA0E	TXINT1E	RXINT1E
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
	<u>Timer Channel 1 Interrupt Request Enable</u> 0: Disabled 1: Enabled	<u>Timer Channel 0 Interrupt Request Enable</u> 0: Disabled 1: Enabled	<u>DMA Interrupt B Channel 1 Enable</u> 0: Disabled 1: Enabled	<u>DMA Interrupt A Channel 1 Enable</u> 0: Disabled 1: Enabled	<u>DMA Interrupt B Channel 0 Enable</u> 0: Disabled 1: Enabled	<u>DMA Interrupt A Channel 0 Enable</u> 0: Disabled 1: Enabled	<u>TXINT1 Enable</u> 0: Disabled 1: Enabled	<u>RXINT Enable</u> 0: Disabled 1: Enabled

**Bit 7: T1IRQE (Timer Channel 1 Interrupt Request Enable)**

**T1IRQE      Functions**

0	Disables timer channel 1 internal interrupts (T1IRQ)
1	Enables timer channel 1 internal interrupts (T1IRQ)

**Bit 6: T0IRQE (Timer Channel 0 Interrupt Request Enable)**

**T0IRQE      Functions**

0	Disables timer channel 0 internal interrupts (T0IRQ)
1	Enables timer channel 0 internal interrupts (T0IRQ)

**Bit 5: DMIB1E (DMA Interrupt B Channel 1 Enable)**

**DMIB1E      Functions**

0	Disables DMAC channel 1 internal interrupts (DMIA)
1	Enables DMAC channel 1 internal interrupts (DMIA)



**Bit 4: DMIA1E (DMA Interrupt A Channel 1 Enable)**

DMIA1E	Functions
0	Disables DMAC channel 1 internal interrupts (DMIA)
1	Enables DMAC channel 1 internal interrupts (DMIA)

**Bit 3: DMIB0E (DMA Interrupt B Channel 0 Enable)**

DMIB0E	Functions
0	Disables DMAC channel 0 internal interrupts (DMIB)
1	Enables DMAC channel 0 internal interrupts (DMIB)

**Bit 2: DMIA0E (DMA Interrupt A Channel 0 Enable)**

DMIA0E	Functions
0	Disables DMAC channel 0 internal interrupts (DMIA)
1	Enables DMAC channel 0 internal interrupts (DMIA)

**Bit 1: TXINT1E (ASCI/CSIO TXINT Enable)**

TXINT1E	Functions
0	Disables TXINT internal interrupts from the ASCI/CSIO
1	Enables TXINT internal interrupts from the ASCI/CSIO

**Bit 0: RXINT1E (ASCI/CSIO RXINT Enable)**

RXINT1E	Functions
0	Disables RXINT internal interrupts from the ASCI/CSIO
1	Enables RXINT internal interrupts from the ASCI/CSIO

**Interrupt Enable Flags (IEF1 and IEF2):** Interrupt enable flag IEF1 specifies whether to enable maskable interrupts ( $\overline{INT0}$ ,  $\overline{INT1}$ ,  $\overline{INT2}$ , and internal interrupts except TRAP). IEF1 = 1 enables interrupts and IEF1 = 0 disables them.

IEF1	Functions
0	Disables maskable interrupts
1	Enables maskable interrupts

IEF1 directly controls interrupt enable/disable status. IEF2 is used to save the value of IEF1. For example, during execution of an  $\overline{\text{NMI}}$  interrupt processing routine when  $\overline{\text{NMI}}$  is requested to CPU, the IEF1 value is saved to IEF2 and IEF1 is reset so that interrupts other than an  $\overline{\text{NMI}}$  are disabled. When returning from an  $\overline{\text{NMI}}$  interrupt processing routine to the main program, the RETN (return from  $\overline{\text{NMI}}$ ) instruction restores the original IEF1 value (stored in IEF2) automatically restoring the CPU interrupt enable status.

Execution of an LD A, I or LD A, R instruction writes the IEF2 value to the P/V flag in the flag register. Thus, the IEF2 value can be read through this flag.

When a maskable interrupt is accepted, IEF1 and IEF2 are set to 0 to disable any subsequent maskable interrupts. When control is returned from the interrupt processing routine to the main routine by an RETI (return from interrupt) instruction, the IEF1 and IEF2 flags remain unchanged. Therefore, to enable interrupts after returning to the main routine, an EI (enable interrupt) instruction must be placed immediately before the last RETI instruction in the interrupt processing routine. This instruction sets IEF1 and IEF2 to 1, enabling maskable interrupts ( $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ , and internal interrupts except TRAP).

Table 3-8 lists the IEF1 and IEF2 values for CPU operations.

**Table 3-8. IEF1 and IEF2 Values**

CPU Operation	IEF1	IEF2	Description
Reset	0	0	Disables all interrupts except $\overline{\text{NMI}}$ and TRAP
After accepting $\overline{\text{NMI}}$	0	IEF1	Saves the IEF1 value to IEF2
RETN instruction execution	IEF2	Unchanged	Restores the status before the $\overline{\text{NMI}}$ was accepted
After accepting interrupts except $\overline{\text{NMI}}$ and TRAP	0	0	Disables all interrupts except $\overline{\text{NMI}}$ and TRAP
RETI instruction execution	Unchanged	Unchanged	–
After accepting TRAP	Unchanged	Unchanged	–
EI instruction execution	1	1	–
DI instruction execution	0	0	–

**Table 3-8. IEF1 and IEF2 Values (cont.)**

<b>CPU Operation</b>	<b>IEF1</b>	<b>IEF2</b>	<b>Description</b>
LD A, I instruction execution	Unchanged	Unchanged	Transfers the IEF2 value to the P/V flag
LD A, R instruction execution	Unchanged	Unchanged	Transfers the IEF2 value to the P/V flag

### 3.6.3 TRAP

TRAP is the highest priority interrupt. It occurs when an undefined opcode is fetched during an opcode fetch cycle. A TRAP interrupt is used to increase program reliability or to implement user-defined instructions.

When a TRAP interrupt occurs, the CPU performs the following:

- Sets the TRAP bit in the interrupt control register (ICR) to 1.
- Saves the program counter (PC) value of the undefined opcode in the stack, then restarts from logical address 0000H.

If logical address 0000H corresponds to physical address 00000H, a reset routine is executed. Since the TRAP bit is only initialized to 0 by an actual reset (not by a reset caused by a TRAP), testing the TRAP bit shows whether the restart has been caused by a reset or a TRAP.

A TRAP interrupt occurs regardless of the status of the IEF1 flag. Furthermore, it may occur when  $\overline{\text{INT0}}$  is used in mode 0 and an undefined opcode is fetched during an interrupt acknowledge cycle.

No undefined opcode can occur in the first byte. Thus, a TRAP may occur when the second opcode in a 2-byte opcode is undefined or when the second or third opcode in a 3-byte opcode is undefined. The PC value, which is saved in the stack when the second opcode is undefined, differs from the value saved in the stack when the third opcode is undefined. Therefore, the byte count of the undefined opcode must be known before a retry takes place. This is indicated by the UFO bit in the interrupt control register (ICR). This bit is set by the third opcode fetch and reset by the first opcode fetch. The UFO bit is not updated while the TRAP bit value is 1. Thus, the address of the first byte opcode can be calculated as follows:

1. When TRAP = 1 and UFO = 0: The TRAP occurred while the CPU was fetching the second opcode. The first opcode address is the PC value in the stack minus 1.

2. When TRAP = 1 and UFO = 1: The TRAP occurred while the CPU was fetching the third opcode. The first opcode address is the PC value in the stack minus 2.

Figure 3-24 shows the timing for TRAP.

In the figure the TTP state is used for TRAP processing. Bus release, refresh, DMA, or wait cycles must not be inserted immediately after this state.

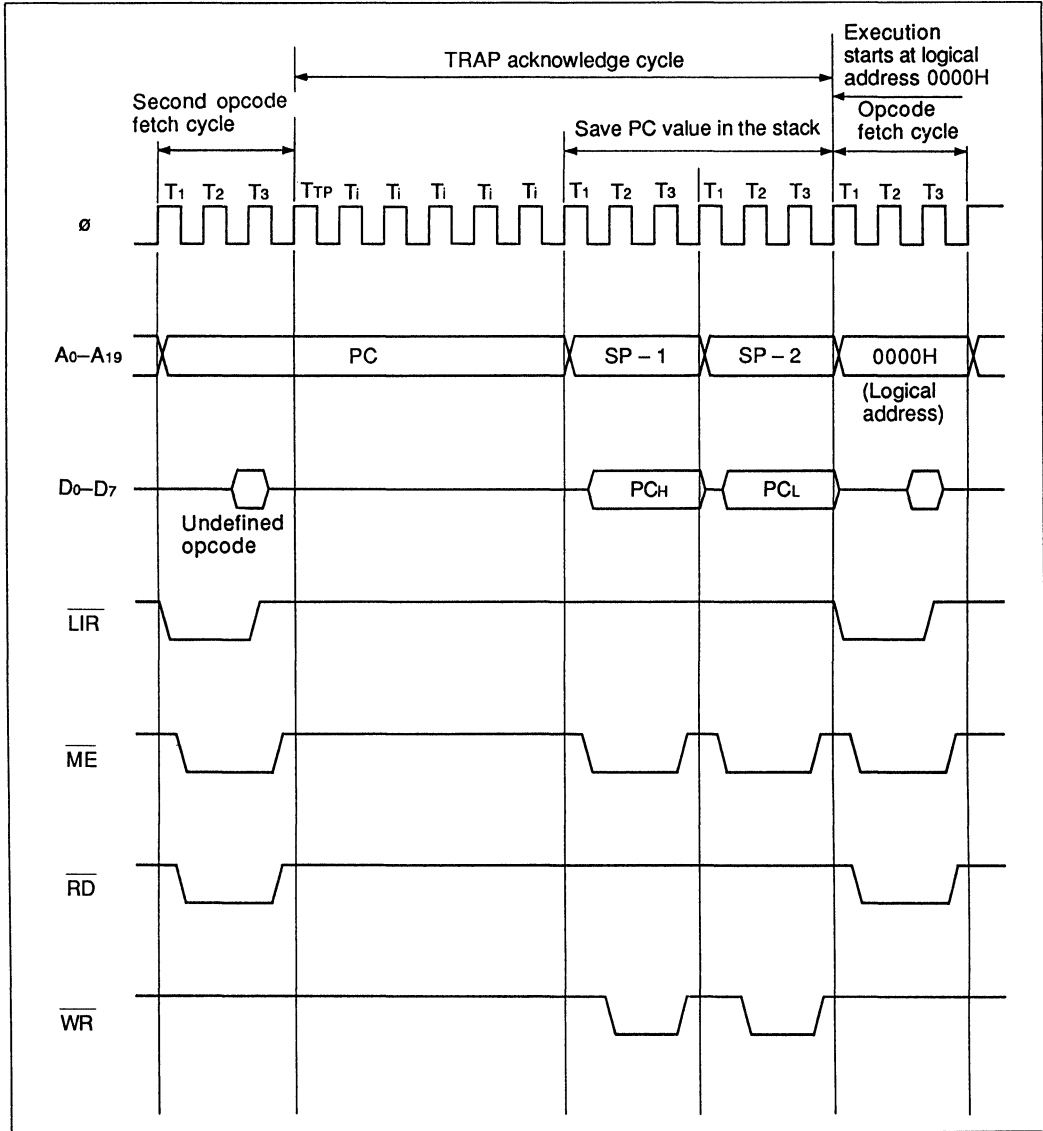


Figure 3-24. (a) TRAP Cycle (Second Opcode is Undefined)

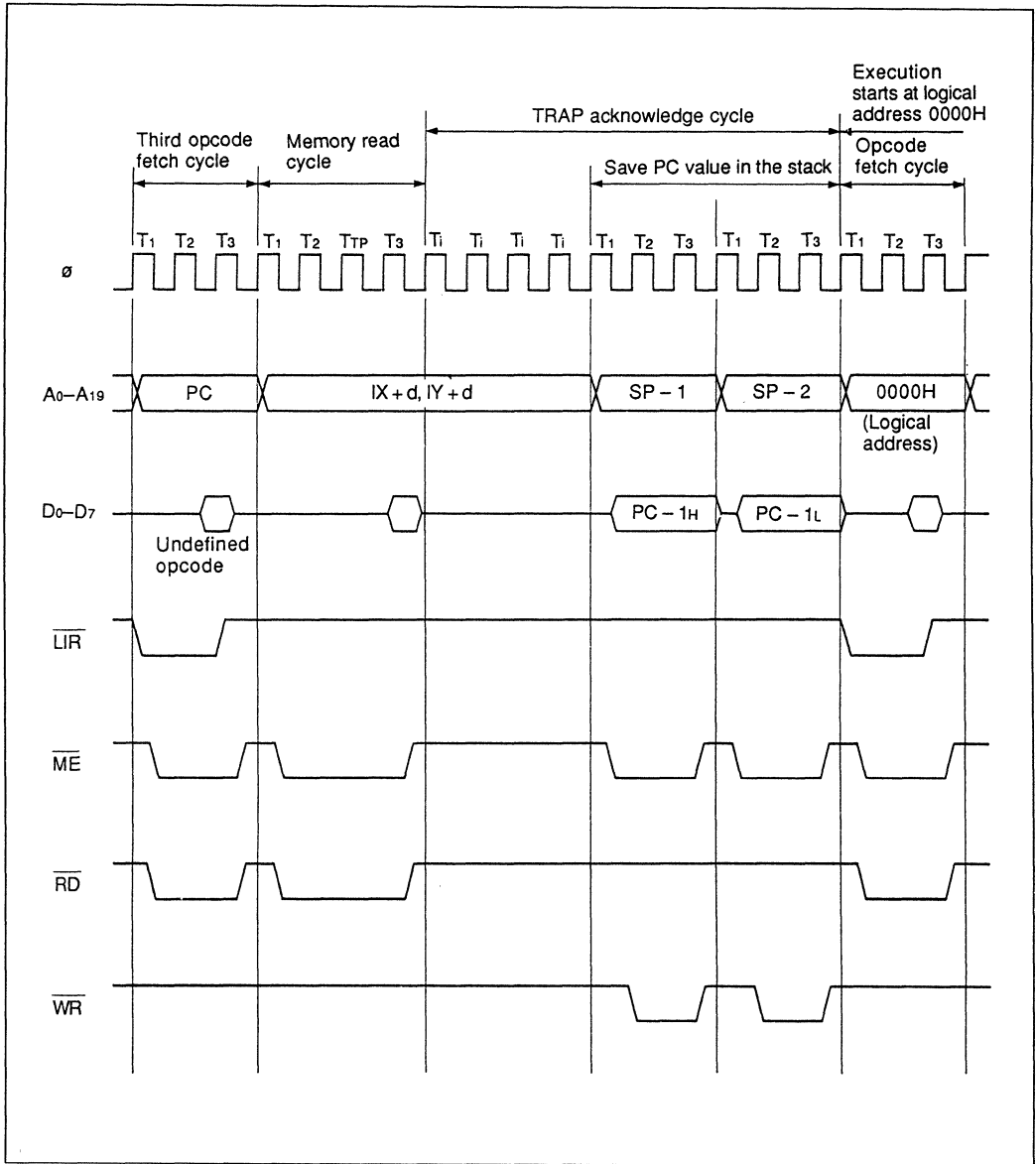


Figure 3-24. (b) TRAP Cycle (Third Opcode is Undefined)

### 3.6.4 Nonmaskable Interrupt ( $\overline{\text{NMI}}$ )

$\overline{\text{NMI}}$  is a nonmaskable interrupt. When an  $\overline{\text{NMI}}$  is detected, the current PC value is saved in the stack and the CPU is restarted at logical address 0066H. The value of IEF1 is saved to IEF2, then IEF1 is reset.

$\overline{\text{NMI}}$  is accepted even when the on-chip DMAC is in operation. It can be used to return bus control to the CPU operation, thus suspending the DMAC operation from an external source.

An RETN instruction is used to return from the  $\overline{\text{NMI}}$  interrupt processing routine to the main program. This instruction also moves the contents of IEF2 into IEF1, thus restoring the IEF1 status before the  $\overline{\text{NMI}}$  interrupt processing.

Figure 3-25 shows the  $\overline{\text{NMI}}$  processing flow.

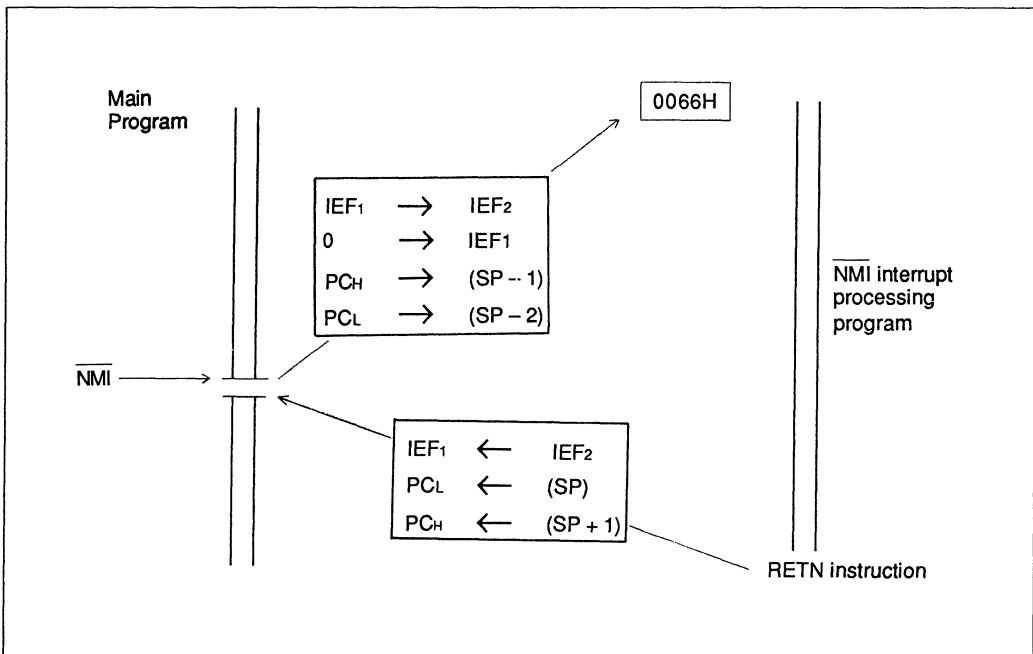


Figure 3-25.  $\overline{\text{NMI}}$  Processing Flowchart

Figure 3-26 shows the timing of  $\overline{\text{NMI}}$  processing.

Interrupt request lines are sampled 1.5 clock cycles before the end of each instruction execution cycle. Unlike other interrupt signals,  $\overline{\text{NMI}}$  is edge-sensitive.  $\overline{\text{NMI}}$  requests are not reset until  $\overline{\text{NMI}}$  is sampled again. If an  $\overline{\text{NMI}}$  is latched before the falling edge of the second state prior to the end of the last machine cycle of the current instruction, an  $\overline{\text{NMI}}$  acknowledge cycle is started immediately after this instruction execution cycle is completed.

**Note:** Other interrupt request signals ( $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ , and  $\overline{\text{INT2}}$ ) are level-sensitive. These signals are sampled 1.5 clocks prior to the end of each instruction execution cycle. In this case, the presence is determined at the sampling time.

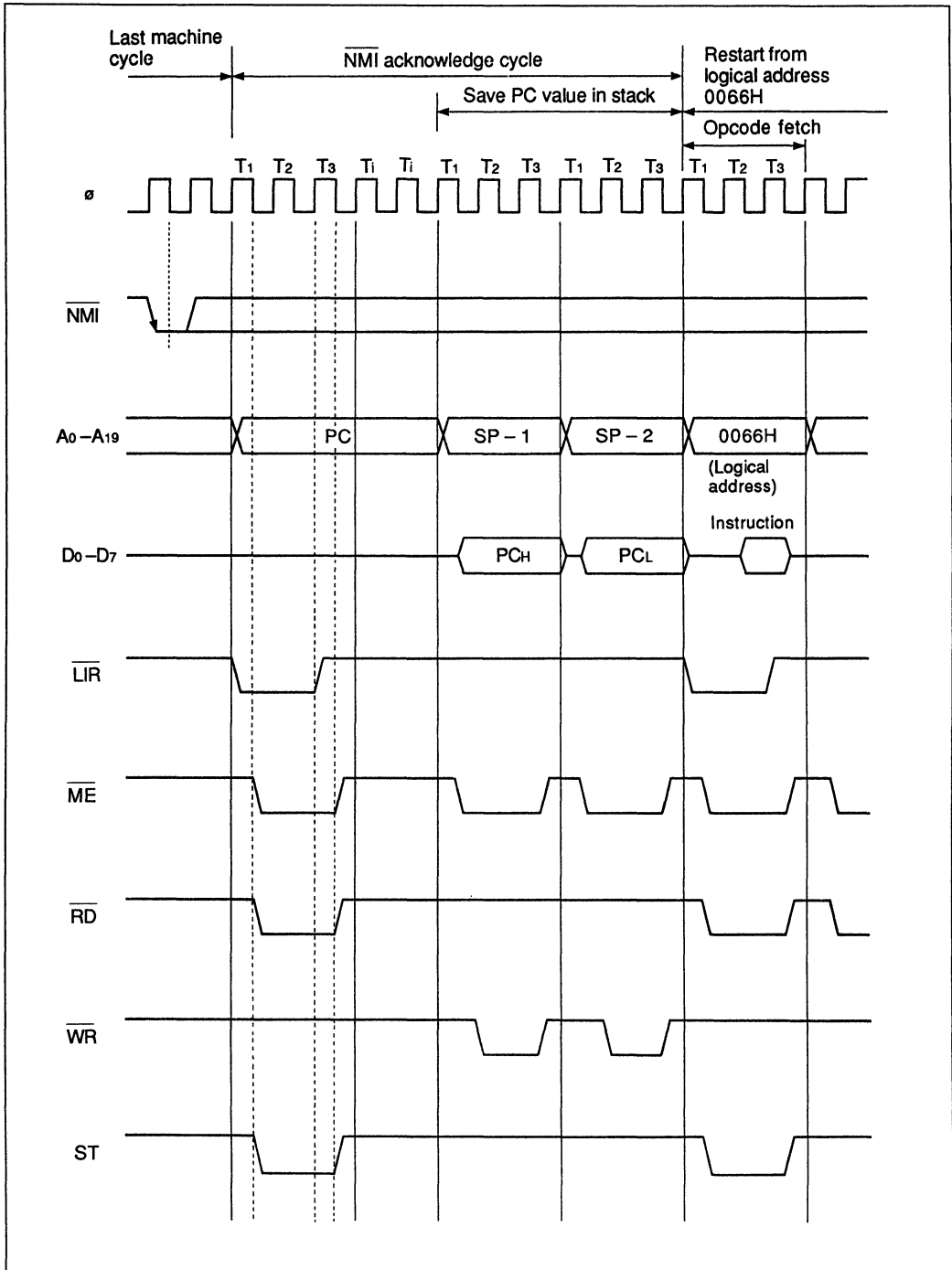


Figure 3-26.  $\overline{\text{NMI}}$  Processing Timing



### 3.6.5 $\overline{\text{INT0}}$

$\overline{\text{INT0}}$  is a level-sensitive maskable interrupt which has second-level priority after  $\overline{\text{NMI}}$ .  $\overline{\text{INT0}}$  can be enabled or disabled by setting or resetting interrupt enable flag IEF1 by an EI or DI instruction. It can also be enabled or disabled by setting or resetting the ITE0 bit (bit 0) of the interrupt control register.

A  $\overline{\text{RESET}}$  causes the following:

- IEF1 is reset, disabling  $\overline{\text{INT0}}$
- The ITE0 bit of the interrupt control register is set to 1, enabling  $\overline{\text{INT0}}$

$\overline{\text{INT0}}$  is disabled by a reset, but it is then enabled by an EI instruction execution. The  $\overline{\text{INT0}}$  line is sampled at the falling edge of the  $\phi$  clock in the second state prior to the end of the last machine cycle of each instruction. If  $\overline{\text{INT0}}$  is low when sampled, it is accepted.

When the interrupt is accepted, IEF1 and IEF2 are reset to disable other interrupts (except  $\overline{\text{NMI}}$  and TRAP). Therefore, to enable interrupts after returning from the interrupt processing routine, an EI and an RETI instruction must be included at the end of the interrupt processing routine. In this case, interrupts can be accepted immediately after the execution of RETI instruction following EI instruction.

The above procedure is also necessary when enabling interrupts after an  $\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ , or any internal interrupt except TRAP.

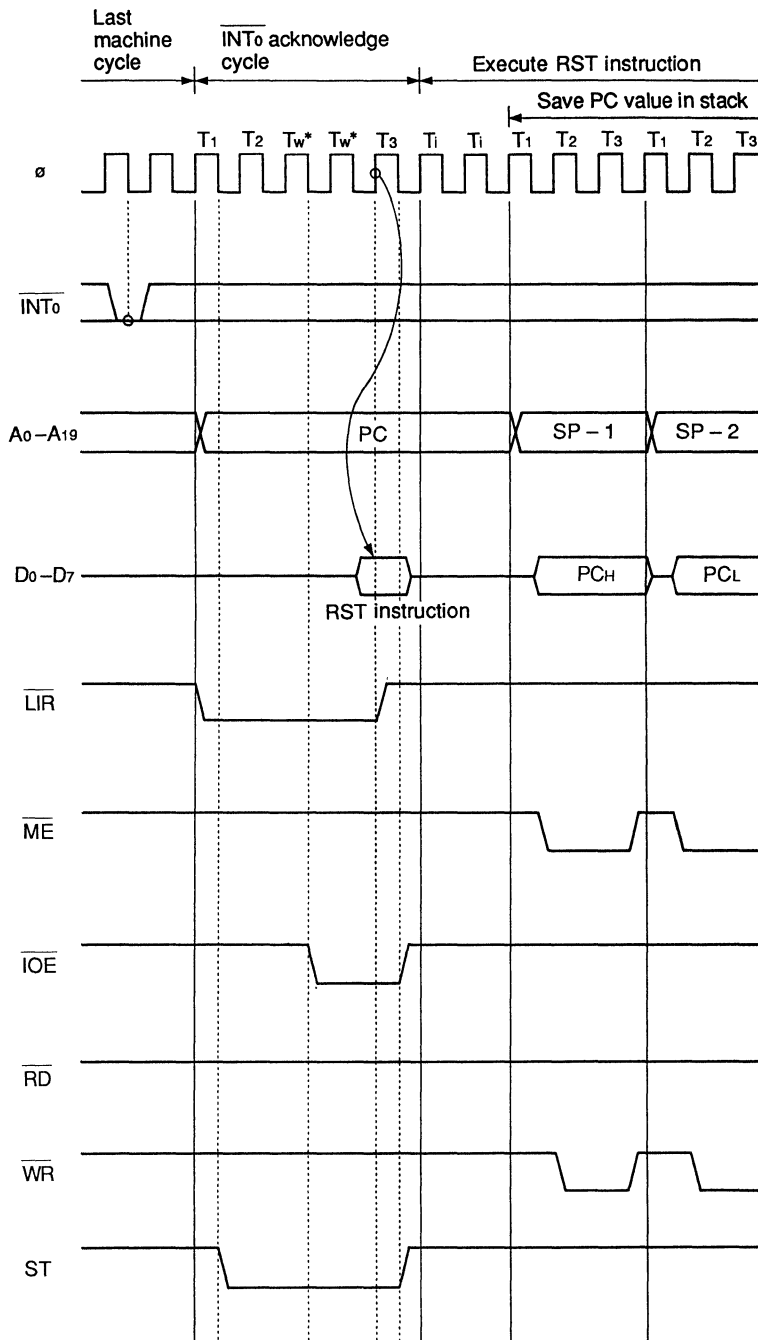
$\overline{\text{INT0}}$  has three operation modes: mode 0, 1, and 2, which are selected by using IM 0, IM 1, and IM 2 instructions, respectively.

After a  $\overline{\text{RESET}}$  signal,  $\overline{\text{INT0}}$  operates in mode 0. The three modes are explained below.

#### (1) $\overline{\text{INT0}}$ mode 0

After an interrupt in this mode, the CPU fetches an instruction placed on the data bus (D0 – D7) during an interrupt acknowledge cycle (at the rising edge of the  $\phi$  clock of the the T3 state) and executes it. Normally, the interrupting device issues a 1-byte RST instruction for restarting from one of eight fixed addresses. However, unlike other modes, the contents of the program counter (PC) are not automatically saved unless an RST instruction is issued.

Figure 3-27 shows the timing diagram for  $\overline{\text{INT0}}$  mode 0.



\*Two wait states are automatically inserted.

Figure 3-27. Timing of  $\overline{\text{INT}}_0$  Mode 0 Interrupt (with an RST instruction on the data bus)

(2)  $\overline{\text{INT0}}$  mode 1

In this mode, the PC value is saved in the stack and instruction execution is restarted at logical address 0038H.

Figure 3-28 shows the flow of interrupt processing in  $\overline{\text{INT0}}$  mode 1. Figure 3-29 shows the timing of an interrupt in this mode.

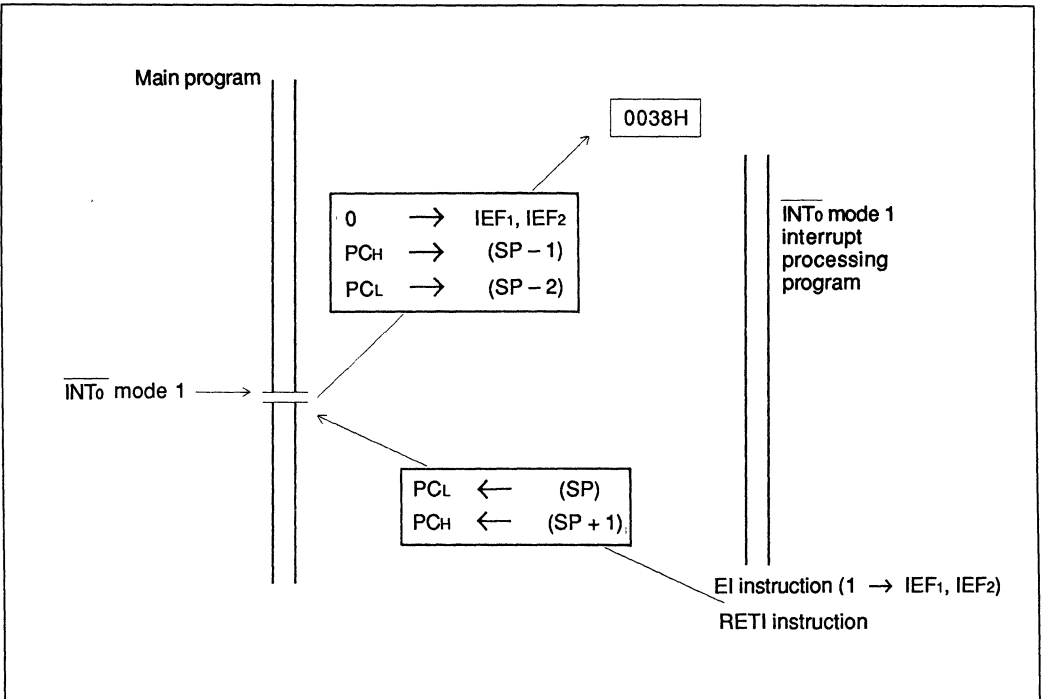
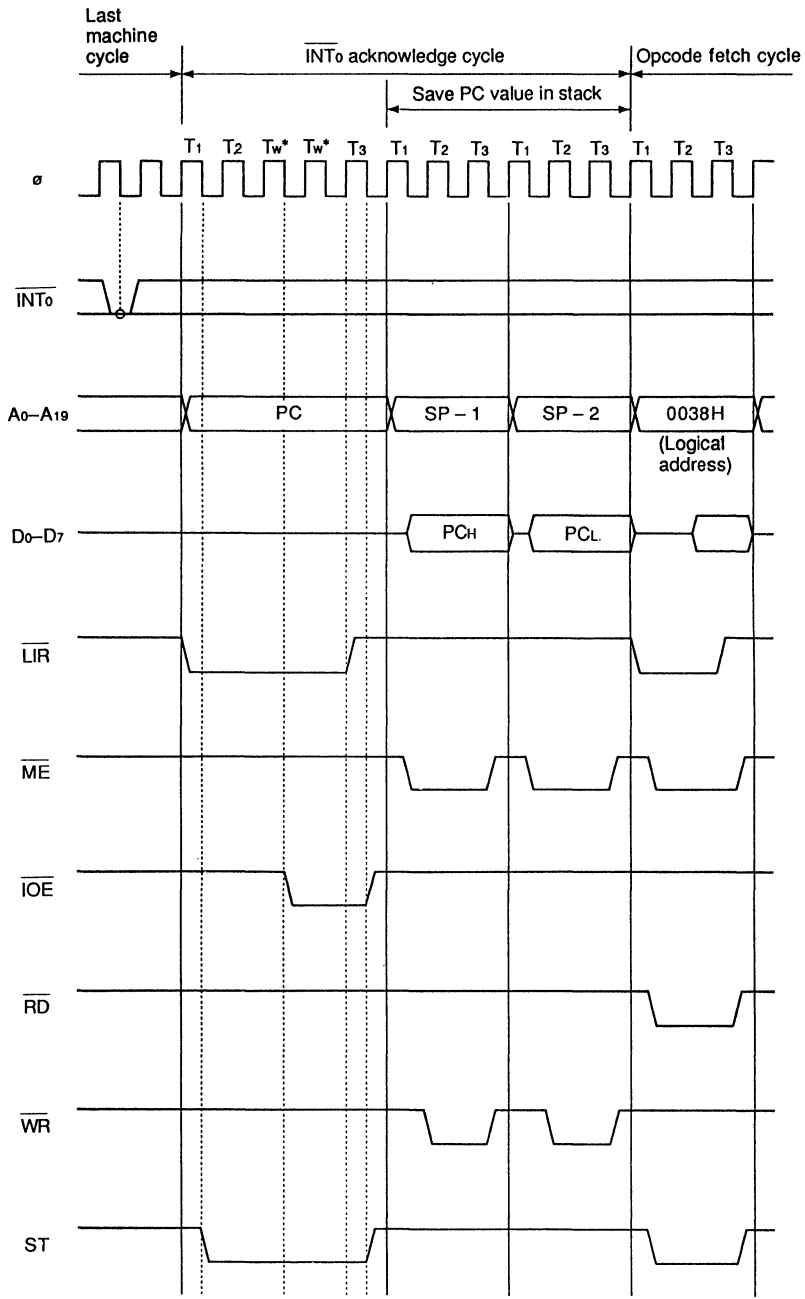


Figure 3-28. Flow of Interrupt Processing in  $\overline{\text{INT0}}$  Mode 1



\* Two wait states are automatically inserted.

Figure 3-29. Timing of  $\overline{\text{INT0}}$  Mode 1 Interrupt

### (3) $\overline{\text{INT0}}$ mode 2

Interrupt vectors are used in this mode. The high order byte of the interrupt vector is stored to the interrupt vector register (I). The low order byte is specified by the interrupting device.

The high-order byte of the interrupt vector must be loaded into the I register in advance. An "LD I, A" instruction can be used to load a new value into the I register. The I register is initialized to 00H by a reset.

During the interrupt acknowledge cycle, the device that generated the interrupt places the low order byte of the interrupt vector on the data bus. The CPU latches the byte at the rising edge of the T3 state ( $\phi$  clock) to generate the 16-bit interrupt vector. After saving the PC value to the stack, the CPU reads the start address (from the table in memory indicated by the interrupt vector), then begins execution from this address.

For more details about the I register, see "Interrupt Vector Register (I)" in section 3.6.2 "Interrupt Control Registers and Interrupt Enable Flags."

Figure 3-30 shows how an interrupt vector is used to generate the starting address for an interrupt processing routine in  $\overline{\text{INT0}}$  mode 2. Figure 3-31 shows the timing of an interrupt in this mode.

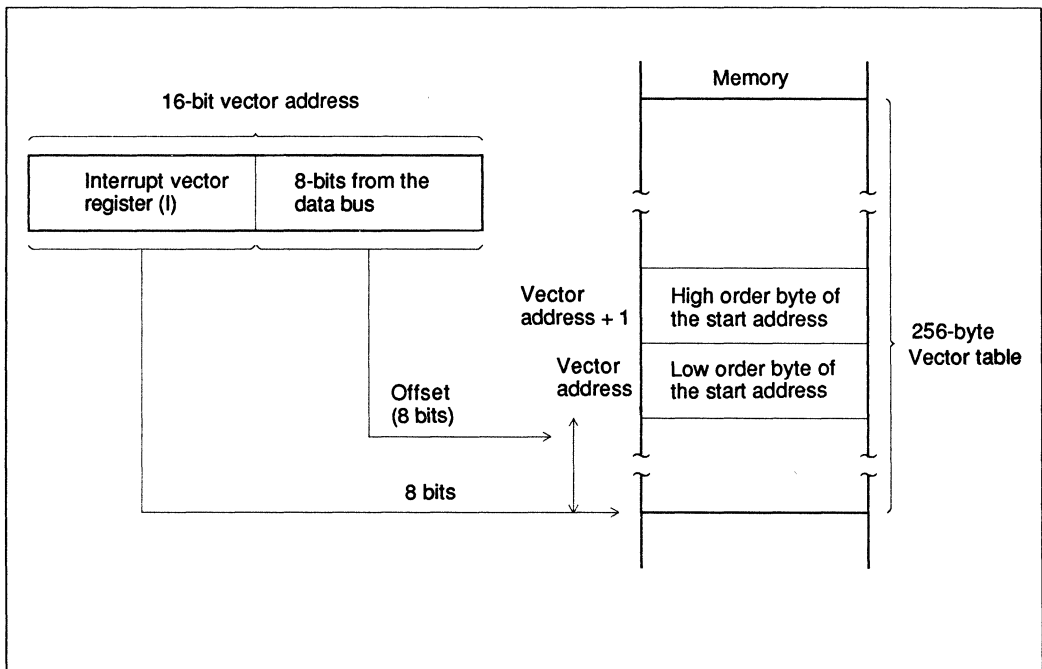


Figure 3-30. Start Address Generation in  $\overline{\text{INT0}}$  Mode 2

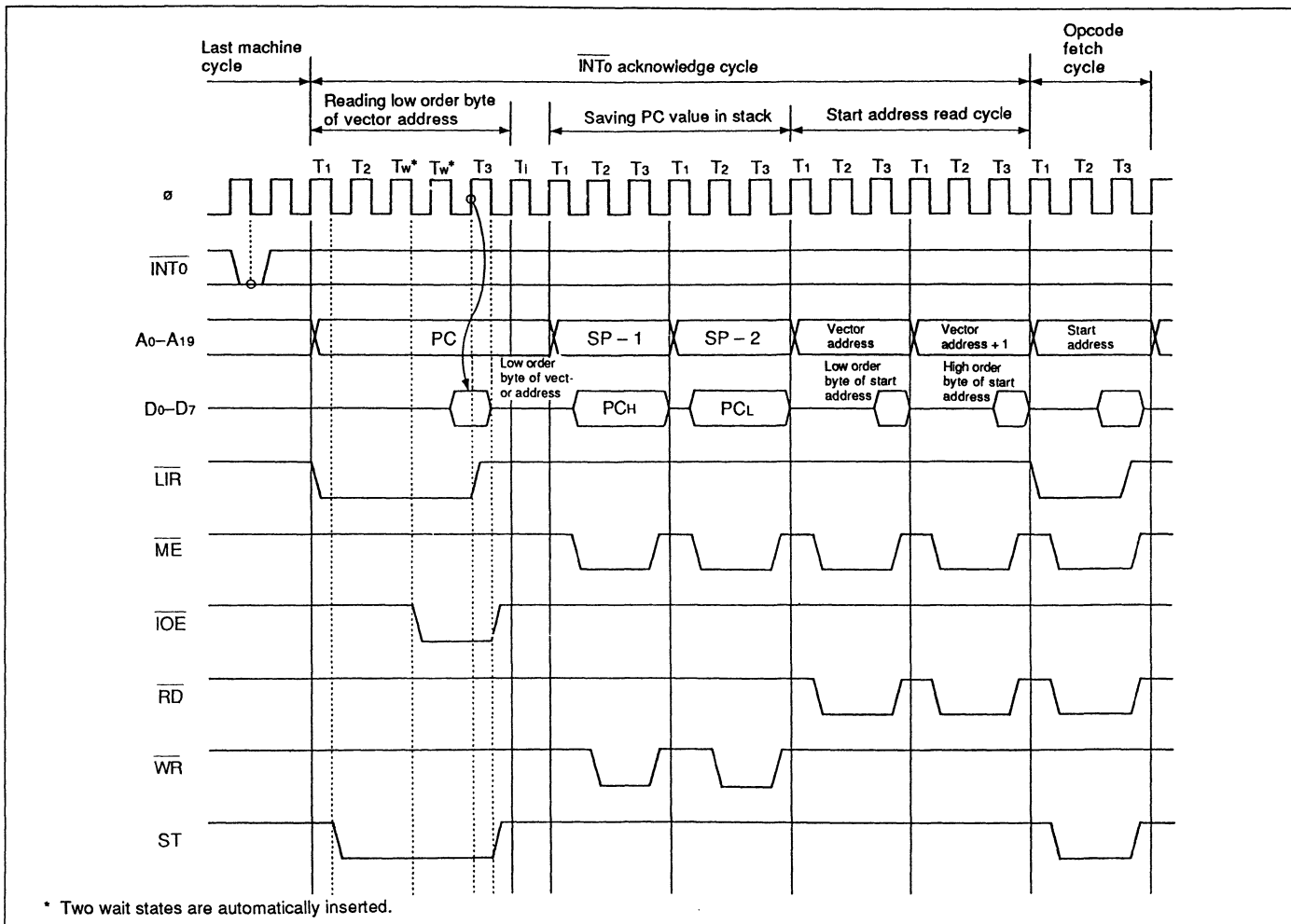


Figure 3-31. Timing of  $\overline{\text{INT0}}$  Mode 2 Interrupt

### 3.6.6 $\overline{\text{INT}}_1$ , $\overline{\text{INT}}_2$ , and the Internal Interrupts (Except TRAP)

$\overline{\text{INT}}_1$ ,  $\overline{\text{INT}}_2$ , and the internal interrupts (except TRAP) are level-sensitive, vector interrupts, similar to  $\overline{\text{INT}}_0$  mode 2. Each of these interrupts has its own enable flag, in addition to the master enable flag IEF1. The INT1E bit (bit 0) in interrupt enable register 0 serves to enable  $\overline{\text{INT}}_1$ , and the INT2E bit (bit 1) in the same register serves to enable  $\overline{\text{INT}}_2$ . Enable flags for the internal interrupts except TRAP are found in the corresponding control registers; in addition, enable bits are found in interrupt enable registers 0 and 1. See figure 3-32 for the circuit representations for  $\overline{\text{INT}}_1$ ,  $\overline{\text{INT}}_2$ , and the internal interrupts except TRAP.

The high order byte of a 16-bit interrupt vector is indicated by the interrupt vector register (I). The low order byte is indicated by the 8-bit interrupt vector low register (IL). The six low order bits (bits 5-0) of the IL register are set to a fixed code depending on the source of the interrupt (see table 3-9). The two high order bits (IL7-6) can be set by an instruction. Thus, the vector table containing the starting addresses of the interrupt processing routines can be located on any 64-byte boundary in the 64 kbyte logical address space by loading the appropriate value into the I register (using an "LD I, A" instruction) and the IL register (using an OUT instruction).

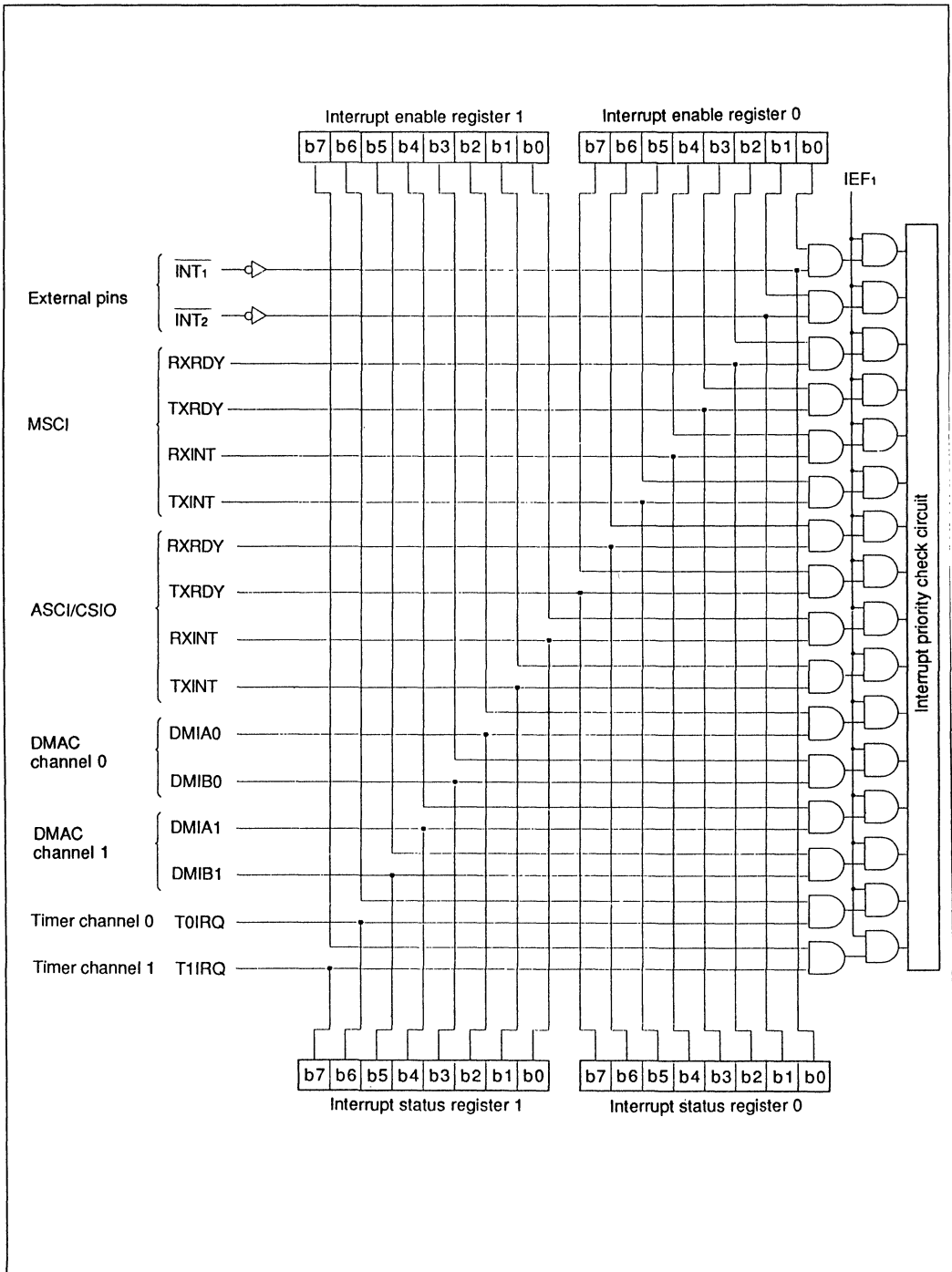


Figure 3-32. Circuit of  $\overline{INT1}$ ,  $\overline{INT2}$ , and Internal Interrupts Except TRAP



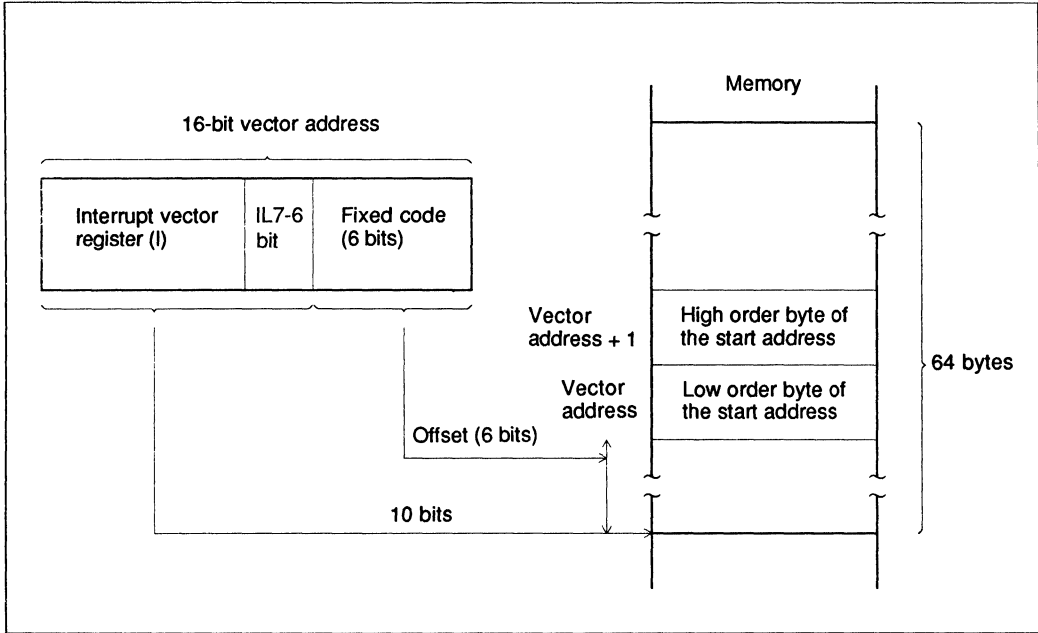
**Table 3-9. Interrupt Sources and Vectors**

Interrupt Source	Priority	IL		Fixed code*					
		b7	b6	b5	b4	b3	b2	b1	b0
INT1	Highest	*	*	0	0	0	0	0	0
INT2	↑	*	*	0	0	0	0	1	0
MSCI RXRDY		*	*	0	0	0	1	0	0
MSCI TXRDY		*	*	0	0	0	1	1	0
MSCI RXINT		*	*	0	0	1	0	0	0
MSCI TXINT		*	*	0	0	1	0	1	0
ASCI/CSIO RXRDY		*	*	0	0	1	1	0	0
ASCI/CSIO TXRDY		*	*	0	0	1	1	1	0
ASCI/CSIO RXINT		*	*	0	1	0	0	0	0
ASCI/CSIO TXINT		*	*	0	1	0	0	1	0
DMAC channel 0 (DMIA0)		*	*	0	1	0	1	0	0
DMAC channel 0 (DMIB0)		*	*	0	1	0	1	1	0
DMAC channel 1 (DMIA1)		*	*	0	1	1	0	0	0
DMAC channel 1 (DMIB1)		*	*	0	1	1	0	1	0
Timer channel 0 (TOIRQ)		*	*	0	1	1	1	0	0
Timer channel 1 (TIIRQ)		*	*	0	1	1	1	1	0
(Reserved)		*	*	1	0	0	0	0	0
		*	*	1	0	0	0	1	0
		*	*	1	0	0	1	0	0
		*	*	1	0	0	1	1	0
		*	*	1	0	1	0	0	0
		*	*	1	0	1	0	1	0
		*	*	1	0	1	1	0	0
		*	*	1	1	0	0	0	0
		*	*	1	1	0	0	1	0
		*	*	1	1	0	1	0	0
		*	*	1	1	0	1	1	0
		*	*	1	1	1	0	0	0
		*	*	1	1	1	0	1	0
		*	*	1	1	1	1	0	0
(Reserved)	Lowest	*	*	1	1	1	1	1	0

\*: Programmable

\* Bit 0 of fixed codes is always 0.

Figure 3-33 shows how an interrupt vector (from the I and IL registers) is used to generate the starting address of an interrupt processing routine.



**Figure 3-33. Start Address Generation for  $\overline{INT1}$ ,  $\overline{INT2}$ , and Internal Interrupts Except TRAP**

A reset initializes interrupt enable registers 0 and 1 to 0s, disables  $\overline{INT1}$ ,  $\overline{INT2}$ , and internal interrupts except TRAP, and resets bits 7 and 6 in the IL register to 0.

Figure 3-34 shows the timing for  $\overline{INT1}$ ,  $\overline{INT2}$ , and internal interrupts except TRAP.  $\overline{INT1}$  or  $\overline{INT2}$  is sampled at the falling edge of the second state prior to the end of the last machine cycle. The sampled  $\overline{INT1}$  or  $\overline{INT2}$  is accepted when it is low.

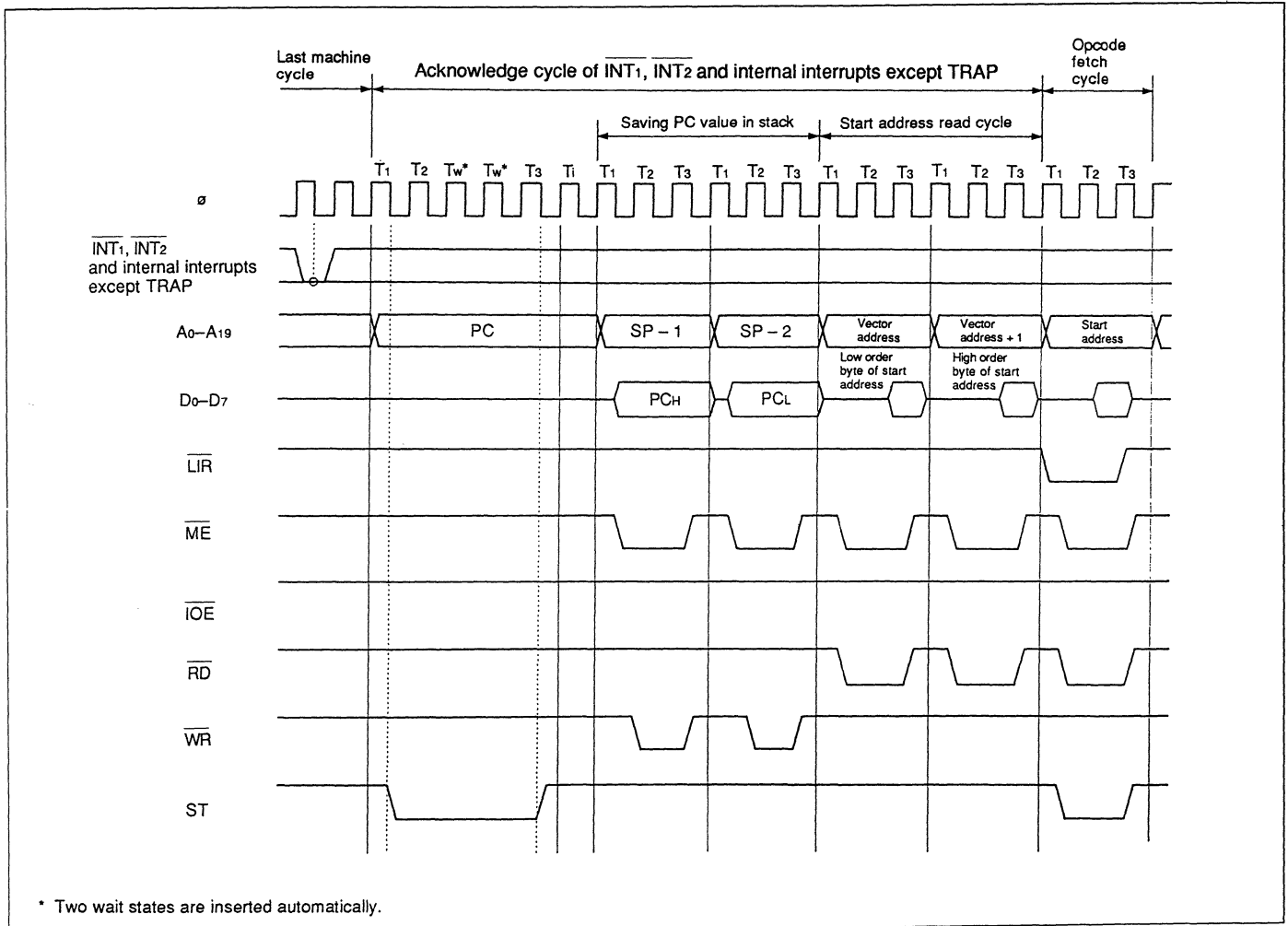


Figure 3-34. Timing for  $\overline{\text{INT}}_1$ ,  $\overline{\text{INT}}_2$ , and Internal Interrupts Except TRAP

### 3.6.7 Initial Values of Flags and Registers Associated with Interrupts

Table 3-10 lists the initial values of the interrupt-associated flags and registers after a reset.

**Table 3-10. Initial Values of Flags and Registers Associated with Interrupts**

Flag/Register	Initial Value	Description
Interrupt enable flags IEF1 and IEF2	0	Interrupts except $\overline{\text{NMI}}$ and TRAP are disabled. They can be enabled by setting IEF1 and IEF2 to 1 using an EI instruction.
Interrupt vector register (I)	0	A vector address table is generated from 0000H to 00FFH in the system memory, thus the vector table contends with reset start address 0000H, $\overline{\text{NMI}}$ restart address 0066H, and $\overline{\text{INT0}}$ mode 1 restart address 0038H. Appropriate values must be set at these addresses using an "LD I, A" instruction.
Interrupt vector low register (IL)	Bits 7 and 6 = 0	Appropriate values must be set in these bits before executing an EI instruction to set IEF1 and IEF2 (similar to the I register).
Interrupt control register (ICR)	All bits except Bit 0 are cleared to 0	Bit 0 = 1 enables $\overline{\text{INT0}}$ .
Interrupt status registers 0 and 1 (ISR0-1)	All bits except the INT2-1 bits are cleared to 0.	The internal interrupt request status is reset.
Interrupt enable registers 0 and 1 (IER0-1)	00H	$\overline{\text{INT1}}$ , $\overline{\text{INT2}}$ , and Internal Interrupts Except TRAP are disabled. If necessary, the interrupt enable bit for each interrupt source can be set to 1.

### 3.6.8 Control Signals for $\overline{\text{INT0}}$ , $\overline{\text{INT1}}$ , $\overline{\text{INT2}}$ , and Internal Interrupts Except TRAP

Control signals in the first machine cycle of an interrupt acknowledge cycle for  $\overline{\text{INT0}}$  differ from those for  $\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ , and other internal interrupts (except TRAP) as follows.

$\overline{\text{INT0}}$  interrupt:  $\overline{\text{LIR}} = 0$ ,  $\overline{\text{IOE}} = 0$ ,  $\text{ST} = 0$

$\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ , and internal interrupts except TRAP:  $\overline{\text{LIR}} = 1$ ,  $\overline{\text{IOE}} = 1$ ,  $\text{ST} = 0$

### 3.7 Memory Management Unit (MMU)

#### 3.7.1 Overview

The CPU contains a memory management unit (MMU). The MMU maps a 64 kbyte logical address space (16-bit addresses) into a 1 Mbyte physical address space (20-bit addresses).

The MMU functions only for CPU memory accesses and not for I/O accesses, DMA cycles, or refresh cycles.

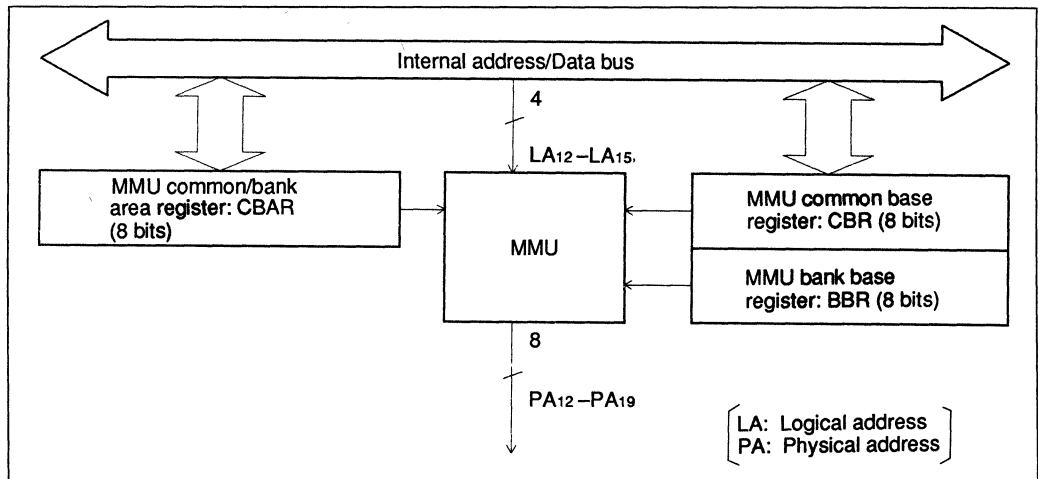
Table 3-11 lists the registers in the MMU.

**Table 3-11. MMU Registers**

Register Name	Symbol	I/O address	Initial Value*	
			MSB ↔ LSB	Read/Write
MMU common/bank area register	CBAR	0003H	11110000	R/W
MMU common base register	CBR	0001H	00000000	R/W
MMU bank base register	BBR	0002H	00000000	R/W

\* "Initial value" means a value after a hardware reset.

Figure 3-35 shows a block diagram of the MMU.



**Figure 3-35. MMU Block Diagram**

### 3.7.2 MMU Registers

The MMU has three registers. The common/bank area register (CBAR) is used to specify the lower address limits for common area 1 and the bank area. The common base register (CBR) is used to translate a logical address in common area 1 to a physical address. The bank base register (BBR) is used to translate a logical address in the bank area to a physical address.

**MMU Common/Bank Area Register (CBAR):** This register is used to specify the four high order bits of the lower address limit for common area 1 and for the bank area.

	7	6	5	4	3	2	1	0
Bit Name	CA3	CA2	CA1	CA0	BA3	BA2	BA1	BA0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	0	0	0	0

<div style="border-top: 1px solid black; width: 100%;"></div> <div style="border-left: 1px solid black; width: 100%; height: 100%;"></div> <div style="border-right: 1px solid black; width: 100%; height: 100%;"></div> <div style="border-bottom: 1px solid black; width: 100%;"></div>	<div style="border-top: 1px solid black; width: 100%;"></div> <div style="border-left: 1px solid black; width: 100%; height: 100%;"></div> <div style="border-right: 1px solid black; width: 100%; height: 100%;"></div> <div style="border-bottom: 1px solid black; width: 100%;"></div>
Four high order bits of the lower address limit for common area 1	Four high order bits of the lower address limit for the bank area

Only the four high order bits of the lower address limit can be specified in this register. The other 12 bits are fixed to 000H. As a result, these areas begin at 4 kbyte boundaries.

This register is set to 11110000 by a reset (the lower address limit for the bank area is 0000H and the lower address limit for common area 1 is F000H).

**Note:** When the lower address limits for common area 1 and the bank area are set, the upper address limits for the bank area and common area 0 are automatically determined (see figure 3-37).

**MMU Common Base Register (CBR):** This register is used to translate a logical address in common area 1 into a physical address. The register bit values are shifted 12 bits to the left and added to the logical address to generate a 20-bit physical address.

	7	6	5	4	3	2	1	0
Bit Name	CB7	CB6	CB5	CB4	CB3	CB2	CB1	CB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

This register is set to 00000000 by a reset.

**MMU Bank Base Register (BBR):** This register is used to translate a logical address in the bank area to a physical address. The register bit values are shifted 12 bits to the left and added to the logical address to generate a 20-bit physical address.

	7	6	5	4	3	2	1	0
Bit Name	BB7	BB6	BB5	BB4	BB3	BB2	BB1	BB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

This register is set to 00000000 by a reset.

### 3.7.3 MMU Operating Space

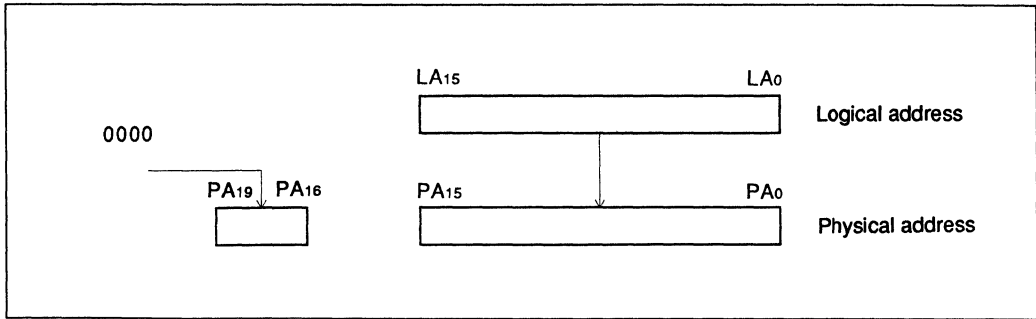
The MMU translates the logical addresses in programs into physical addresses. The role of the MMU in memory accesses, I/O accesses, in DMAC operations, and refresh cycles is explained below.

**Memory Access by CPU:** The MMU functions whenever the CPU accesses memory. It translates a 16-bit logical address into a 20-bit physical address in the following cases:

- Instruction fetch
- Memory read/write by instruction
- Interrupt vector address specification
- Interrupt restart address specification

For the relationship between logical and physical addresses during memory accesses, see section 3.7.4 "MMU Operation."

**I/O Access by CPU:** The MMU does not function when the CPU accesses the I/O space. Figure 3-36 shows the relationship between physical and logical addresses.



**Figure 3-36. Relationship Between Physical and Logical Addresses for I/O Accesses**

As shown in the figure above, bits 15-0 of the 20-bit physical address correspond to the logical address, and bits 19-16 are 0000.

**DMAC Operation or Refresh Cycle:** The MMU is not used during DMAC operations or refresh cycles. The address value generated by the DMAC or refresh controller is output directly as the physical address.

### 3.7.4 MMU Operation

The logical address space consists of common areas 0 and 1, and a bank area (see figure 3-37). The contents of the common/bank area register (CBAR) specify the boundaries between these areas. These boundaries can be set at any 4 kbyte boundary in the logical address space.

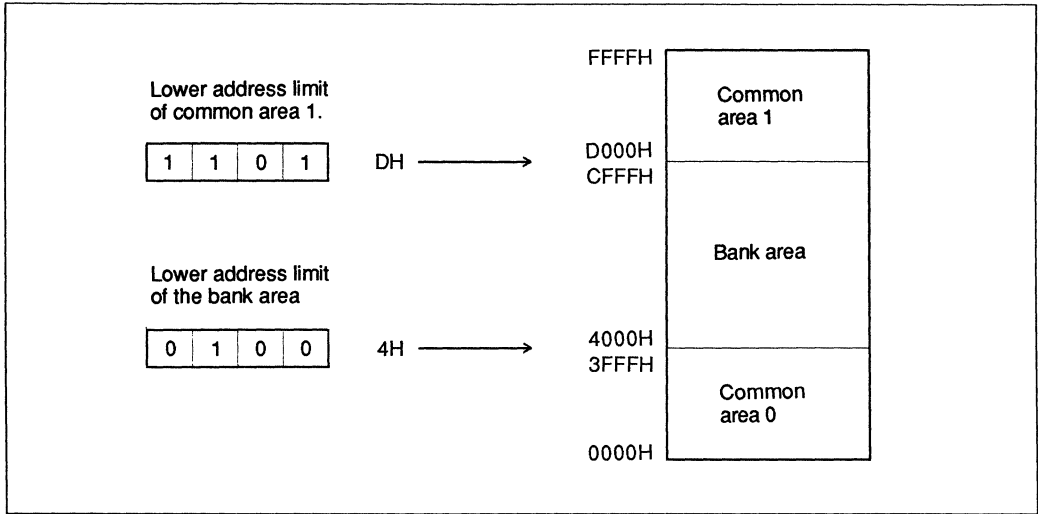
The four high order bits (CA3-0) of the CBAR specify the four high order bits (A15 – A12) of the lower address limit of common area 1. The four low order bits (BA3-0) of the CBAR specify the four high order bits of the lower address limit of the bank area. For example, when the CA3-0 bits of the CBAR are set to 1101 (DH) and the BA3-0 bits are set to 0100 (4H), the lower address limit of common area 1 is D000H and the lower address limit of the bank area is 4000H. As a result, common area 1, the bank area, and common area 0 are defined as follows:

	Upper Address Limit	Lower Address Limit
Common area 1	FFFFH	D000H
Bank area	CFFFH *	4000H
Common area 0	3FFFH *	0000H

\* The upper address limits of the bank area and common area 0 are automatically determined by setting the lower address limits of common area 1 and the bank area.



The lower address limit of common area 1 must be greater than or equal to the lower address limit of the bank area. If this is violated, normal operation is not guaranteed.



**Figure 3-37. Example of Logical Address Space Division**

If the lower address limit of the bank area is set to 0H (reset condition), common area 0 has no size.

The following paragraphs explain how the logical address space is mapped into the physical address space. Figure 3-38 shows how physical addresses are generated. Figure 3-39 shows the relationship between the logical and physical spaces.

The MMU base registers are used to translate a logical address into a physical address. The four high order bits (LA12 – LA15) of the logical address are added to the 8-bit base register value to generate the eight high order bits (PA12 – PA19) of the physical address. The LA0 – LA11 values are used directly as the 12 low order physical address bits (PA0 – PA11).

The base register for common area 1 is the common base register (CBR). The base register for the bank area is the bank base register (BBR). For a common area 0, the base register is assumed to be fixed at 00H (i.e., the logical address equals the physical address).

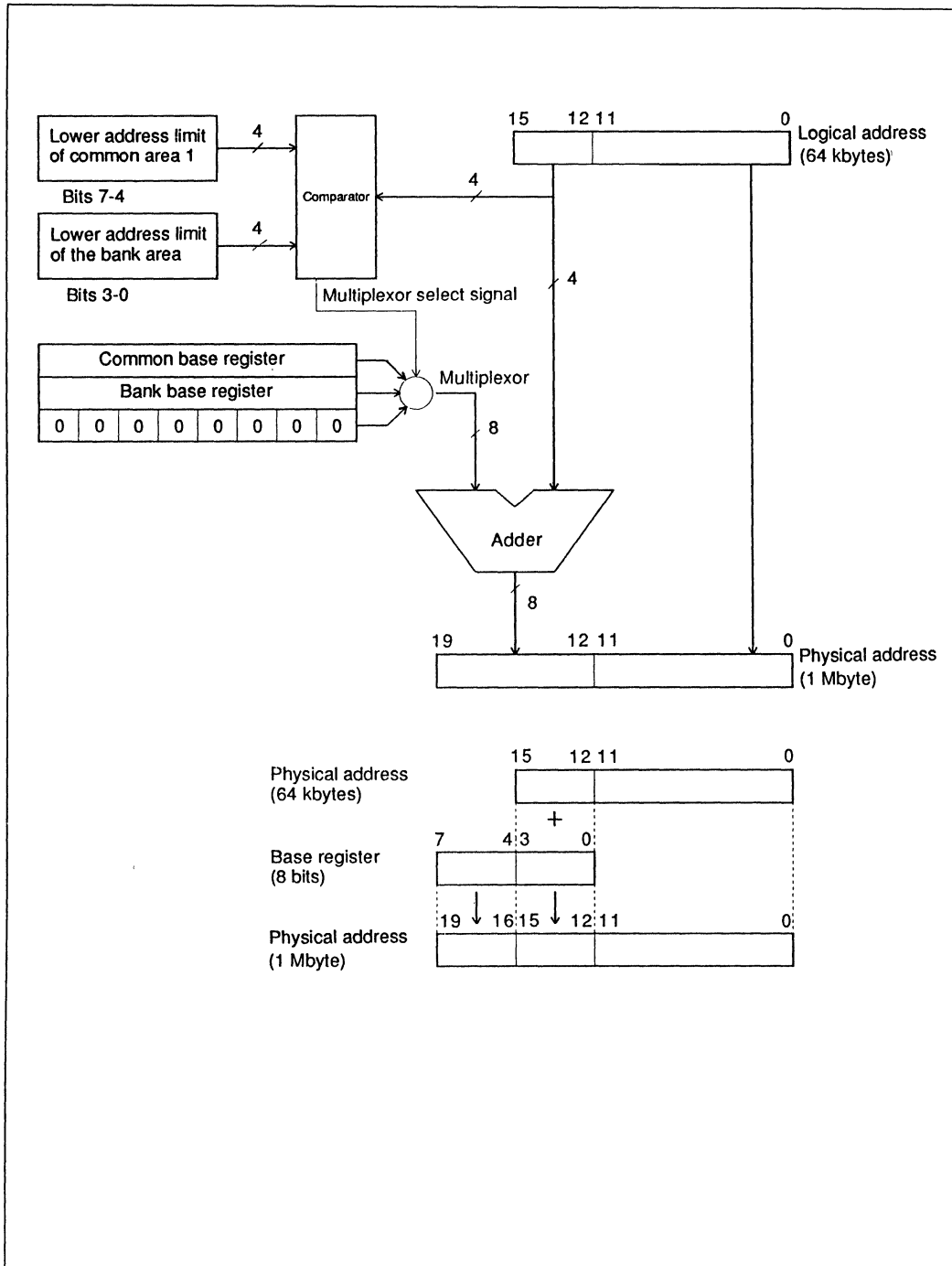
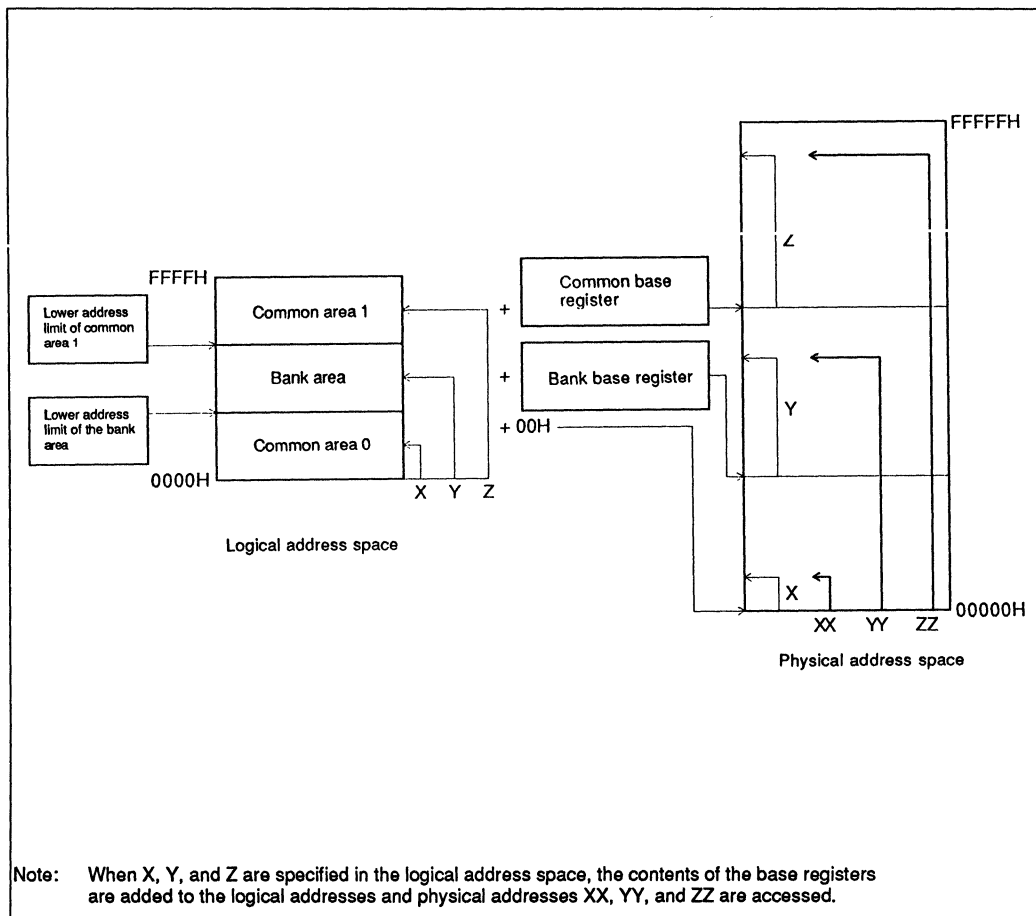


Figure 3-38. Physical Address Generation



**Figure 3-39. Relationship Between Logical and Physical Spaces**

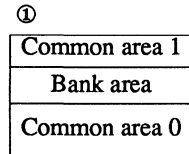
### 3.7.5 MMU and Reset

A reset sets the CBAR to 11110000 and the other registers (CBR and BBR) to 00000000. In this case, logical addresses equal physical addresses and act as if there were no MMU.

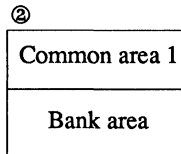
After a reset, instruction execution is always restarted at physical address 00000H (logical address 0000H).

### 3.7.6 MMU Operating Precautions

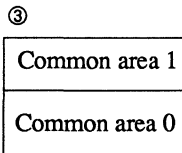
**Setting Lower Address Limits:** The three areas of the logical address space can be divided in various ways by setting their lower address limits:



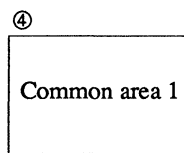
Lower address limit of  
common area 1  
> Lower address limit of  
the bank area  
> 0000H



Lower address limit of  
common area 1  
> Lower address limit of  
the bank area  
= 0000H



Lower address limit of  
common area 1  
= Lower address limit of  
the bank area  
> 0000H



Lower address limit of  
common area 1  
= Lower address limit of the  
bank area  
= 0000H

Note: The areas are initialized by reset to the condition shown in ② above.

Lower address limits must meet the following condition:

Lower address limit of common area 1  $\geq$  lower address limit of the bank area.

If this is not met, normal operation is not guaranteed.

**Setting MMU Registers:** When data is loaded into an MMU register (CBAR, CBR, or BBR), it does not become valid until the first machine cycle after the end of the I/O write cycle.

If the area in which the program is stored is changed during program execution, the first instruction after the MMU register write will be fetched from the new physical address.

## Section 4. Multiprotocol Serial Communications Interface (MSCI)

### 4.1 Overview

The multiprotocol serial communications interface (MSCI) supports three different operating modes: asynchronous, byte synchronous, and bit synchronous.

#### 4.1.1 Functions

The MSCI includes the following functions:

- Program-selectable operating modes: asynchronous, byte synchronous, and bit synchronous
- Transmission codes NRZ, NRZI, Manchester, FM0 and FM1 are supported. (Only NRZ code is supported in the asynchronous mode.)
- Full duplex communications, auto echo, and local loop back functions are available.
- Separate transmit and receive buffers are provided for each three stages.
- Modem control signals  $\overline{\text{RTSM}}$ ,  $\overline{\text{CTSM}}$  and  $\overline{\text{DCDM}}$  can be automatically controlled using the auto-enable function.

$\overline{\text{RTSM}}$  (Request To Send): General-purpose output/transmission request

$\overline{\text{CTSM}}$  (Clear To Send): General-purpose input/transmit enable/transition-triggered interrupt

$\overline{\text{DCDM}}$  (Data Carrier Detect): General-purpose input/receive carrier detection/transition-triggered interrupt

- Programmable on-chip baud rate generator for transmission and reception
- Clock is program-selectable from three sources: external clock input, on-chip baud rate generator output and internal ADPLL (Advanced Digital PLL) output.
- Noise suppression function for receive clock and receive data
- Data transmission rate of 7.1 Mbps for a 10 MHz system clock
- Four internal interrupt signals: RXRDY, TXRDY, RXINT, and TXINT

Functions of the MSCI in the synchronous, byte synchronous, and bit synchronous operation modes can be summarized as follows:

#### (1) Asynchronous mode

- Full duplex mode supported

- Programmable character length (5-8 bits/character) is specified for transmission and reception
- Programmable parity (odd, even, or no parity)
- Programmable stop bit length (1, 1.5, or 2 bits)
- Programmable clock rate for transmission and reception (input clock frequency  $\times$  1/1, 1/16, 1/32, or 1/64)
- Detection of parity, overrun, and framing errors
- Transmission and reception breaks
- Multiprocessor (MP) bit transmission and reception

## (2) Byte synchronous mode

- 8-bit character length
- Mono-sync, bi-sync, and external synchronous modes supported
- CRC code generation and check. Initial value (all 0s or 1s) is selectable for either CRC-16 or CRC-CCITT generator polynomials
- Automatic SYN character transmission, detection, and deletion
- CRC code transmission/no-transmission is program-selectable for transmission buffer underruns
- Transmission of SYN character or mark is program-selectable for the idle state
- Detection of CRC, overrun, and underrun errors

## (3) Bit synchronous mode

- 8-bit character length
- HDLC and loop modes supported
- Information (I) field configured in bytes
- Automatic zero insertion in transmit data and deletion from receive data
- Flag or mark transmission is program-selectable in idle state
- 8- or 16-bit address (A) field is selectable. Four address field check modes are program-selectable
- End-of-frame detection
- CRC code generation and detection

### 4.1.2 Configuration and Operation

Figure 4-1 shows a block diagram of the MSCI.

The MSCI has 21 internal registers that can be accessed by the user. These registers are used for specifying the operating mode and controlling transmission and reception operations.

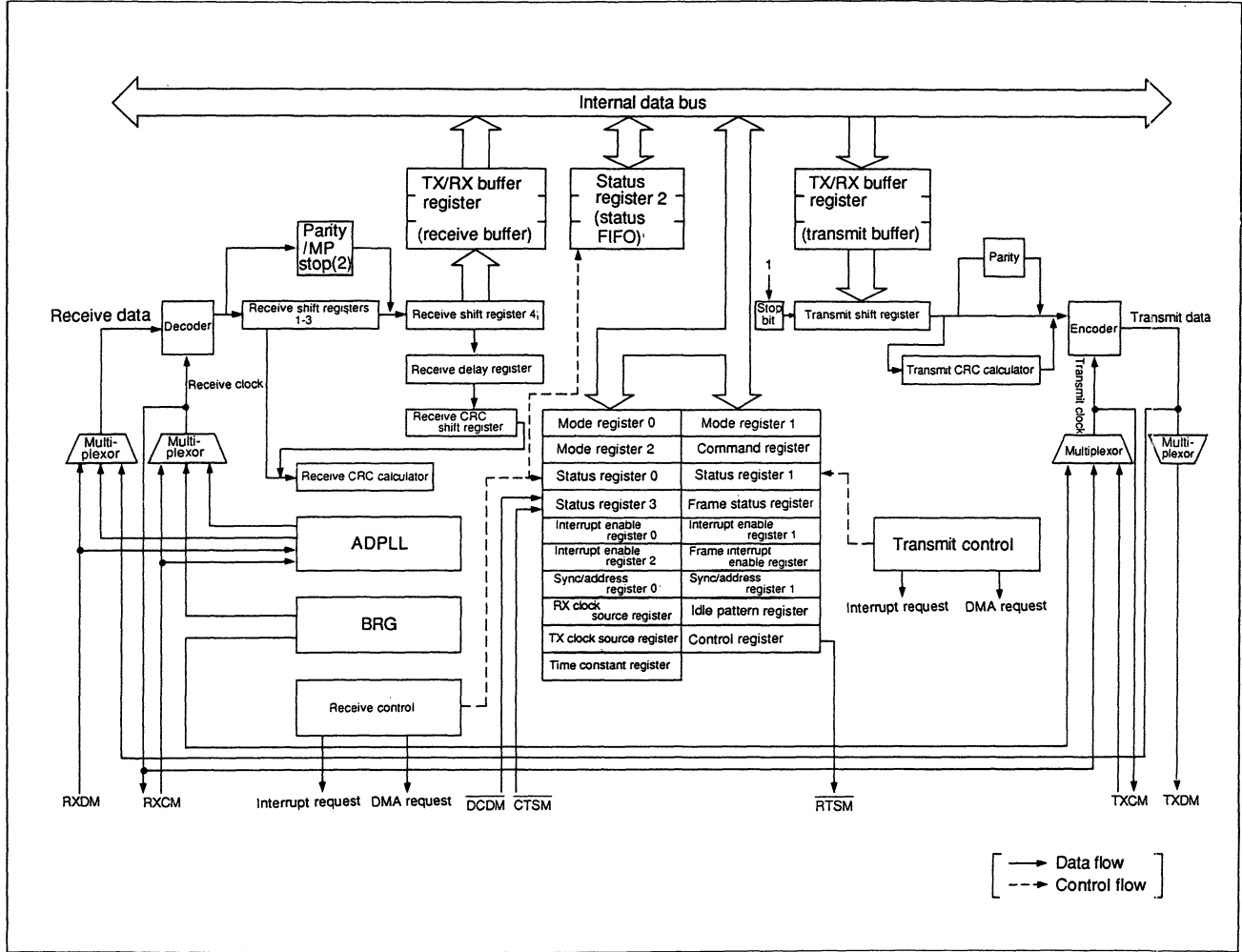


Figure 4-1. MSCI Block Diagram

**Receiver:** Figure 4-2 shows a block diagram of the MSCI receiver.

The MSCI receiver has one 3-stage FIFO buffer, five 8-bit shift registers, and one delay register.

The receiver also has a 6-bit status buffer (FIFO)\*1. This buffer retains status information, such as parity or framing errors, related to the received data.

Input data is received via the RXDM line and enters the MSCI internal circuitry after passing through a decoder. The data path inside the MSCI differs according to the operating mode (asynchronous, byte synchronous, or bit synchronous).

In the asynchronous mode, input data is checked for the parity/MP bit and for framing errors before being passed to receive shift register 4. The data is then sent to the receive buffer as each character is received\*2. The CPU or DMAC can read the receive buffer data via the internal data bus.

In the byte synchronous mode, input data enters receive shift register 1 before branching toward both receive shift register 2 and receive shift register 4.

The data received by receive shift register 2 is used to detect SYN character(s). The data received by receive shift register 4 is transmitted to the receive buffer. And the received data is transmitted to the receive CRC calculator via the receive delay register and the receive CRC shift register.

Output from the CRC calculator is sent to the MSCI status register 2 (MST2). The CPU or DMAC can read the received data and status via the internal data bus.

In the bit synchronous mode, the input data enters receive shift register 1, which deletes 0s, and detects flags, abort status, and idle status. The data then branches toward receive shift register 2 and toward the receive CRC calculator. Output from the CRC calculator is passed to MST2, as in the case of the byte synchronous mode. Its contents are also sent to the MSCI frame status register (MFST) at the completion of frame reception. Therefore, the MFST always holds the status of the most recently received frame.

\*1 MSCI status register 2 (MST2) is located at the top of the status buffer (FIFO) and interfaces with the internal data bus. For details, see section 4.2.11 "MSCI Status Register 2."

\*2 The MSCI TX/RX buffer register (MTRB) is located at the top of the receive buffer and interfaces with the internal data bus. For details, see section 4.2.21 "MSCI TX/RX Buffer Register."



As for the data sent to receive shift register 2, the secondary station address is detected. The data is then sent via receive shift register 3 to receive shift register 4 and the receive buffer. The CPU or DMAC can read the received data and status via the internal data bus. If CRC calculation is disabled (the CRCCC bit of the MSCI mode register 0 is 0), the received data is sent directly from receive shift register 1 to receive shift register 4. The secondary station address is detected in the same way.

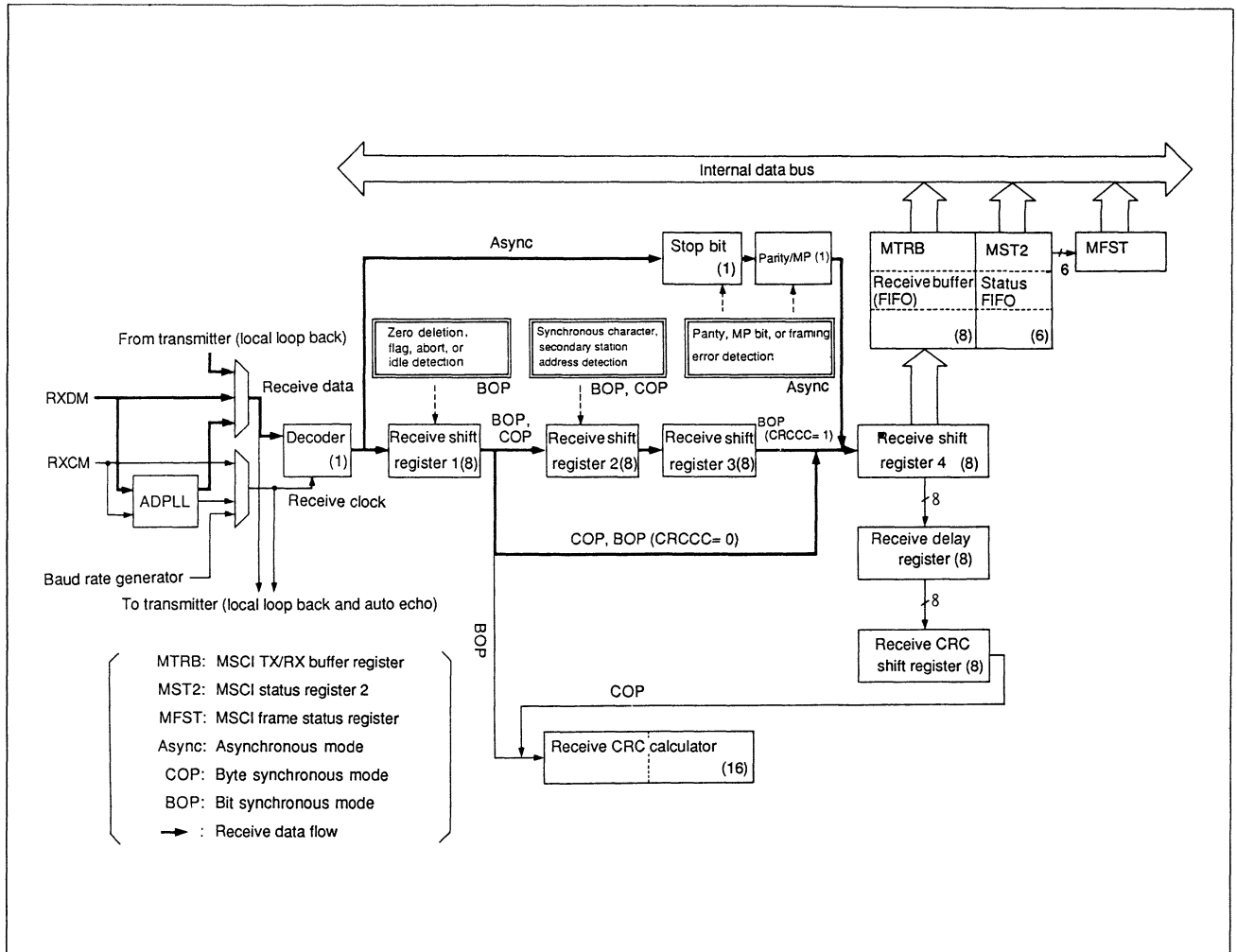


Figure 4-2. Block Diagram of the MSCI Receiver

**Transmitter:** Figure 4-3 shows a block diagram of the MSCI transmitter.

The MSCI transmitter has a 3-stage FIFO buffer, a transmit shift register, and a TX pattern register. It also has a CRC calculator similar to that of the receiver.

Output data is written via the internal data bus to the transmit buffer by the CPU or DMAC. Information necessary to assemble frames in the associated communications mode is appended to the output data in the transmit shift register. The data is then output to the TXDM line after passing through the encoder.

See sections 4.2.1 "MSCI Mode Register 0," 4.2.2 "MSCI Mode Register 1," 4.2.4 "MSCI Control Register," 4.2.18 "MSCI Synchronous/Address Register 0" and 4.2.19 "MSCI Synchronous/Address Register 1" for details about specifying parity, stop bit length, and break transmission in the asynchronous mode. These sections also contain information about specifying SYN characters, aborts, flags, and details about CRC calculation in the byte and bit synchronous modes.

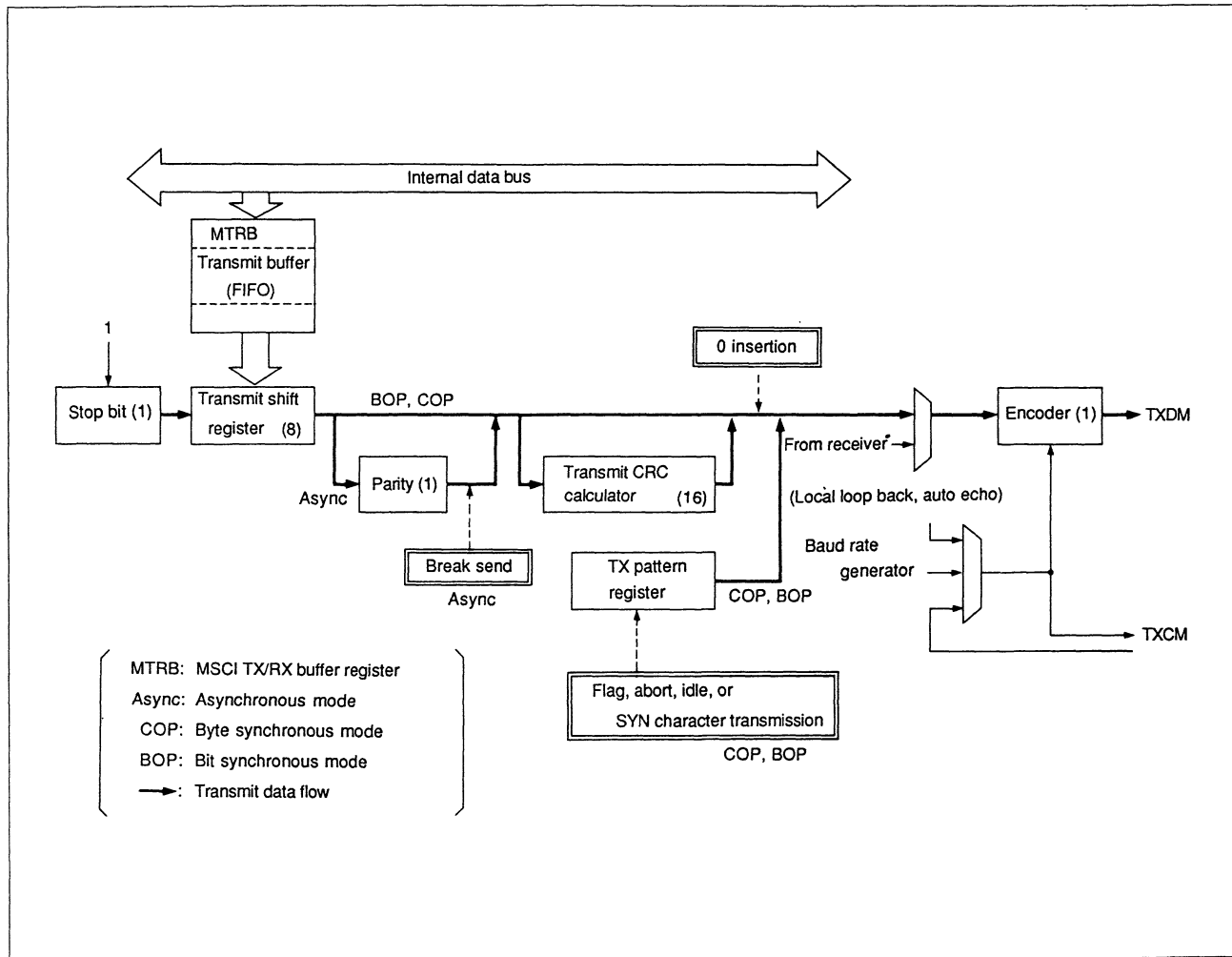


Figure 4-3. Block Diagram of the MSCI Transmitter

### 4.1.3 Registers

Table 4-1 lists the MSCI registers.

**Table 4-1. MSCI Registers**

Register Name	Symbol	I/O Address	Initial Value*1	
			MSB↔LSB	Read/Write*2
MSCI mode register 0	MMD0	002BH	00000000	R/W
MSCI mode register 1	MMD1	002CH	00000000	R/W
MSCI mode register 2	MMD2	002DH	00000000	R/W
MSCI control register	MCTL	002EH	00000001	R/W
MSCI RX clock source register	MRXS	0033H	00000000	R/W
MSCI TX clock source register	MTXS	0034H	00000000	R/W
MSCI time constant register	MTMC	0032H	00000001	R/W
MSCI command register	MCMD	002AH	—	W
MSCI status register 0	MST0	0021H	00000000	R
MSCI status register 1	MST1	0022H	00000000	R/W
MSCI status register 2	MST2	0023H	00000000	R/W
MSCI status register 3	MST3	0024H	0000XX*300	R
MSCI frame status register	MFST	0025H	00000000	R/W
MSCI interrupt enable register 0	MIE0	0026H	00000000	R/W
MSCI interrupt enable register 1	MIE1	0027H	00000000	R/W
MSCI interrupt enable register 2	MIE2	0028H	00000000	R/W
MSCI frame interrupt enable register	MFIE	0029H	00000000	R/W
MSCI synchronous/address register 0	MSA0	002FH	11111111	R/W
MSCI synchronous/address register 1	MSA1	0030H	11111111	R/W
MSCI idle pattern register	MIDL	0031H	11111111	R/W
MSCI TX/RX buffer register	MTRB	0020H	XXXXXXXX	R/W*4

X : Undefined

\*1 Value after a hardware reset or a reset command

\*2 Writing in the same bit might mean other function respectively according to the operating mode (asynchronous, byte synchronous, or bit synchronous). For details, see the explanation of registers from section 4.2.1.

\*3 Bits 3 and 2 in the MSCI status register 3 read the  $\overline{\text{CTS}}\text{M}$  and  $\overline{\text{DCD}}\text{M}$  line levels.

\*4 The MSCI TX/RX buffer register functions as a character receive buffer during read operations and as a character transmit buffer during write operations.

## 4.2 Registers

The MSCI has 21 registers which are used to select the operating mode (asynchronous, byte synchronous, or bit synchronous), control the transmitter and receiver, and control the ADPLL and baud rate generator. CPU I/O instructions are used for accessing these registers.

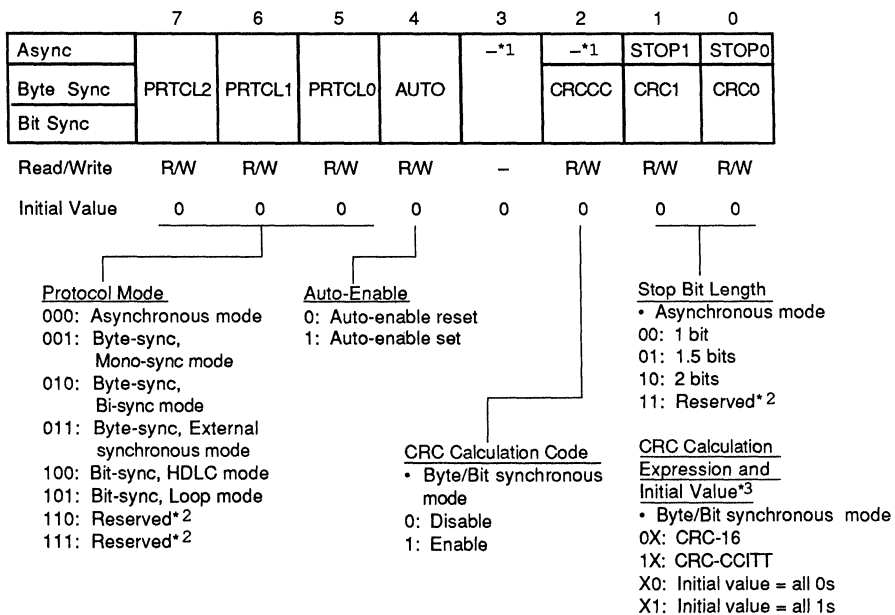
When changing the operating mode, these registers must be initialized by a channel reset command.

### 4.2.1 MSCI Mode Register 0 (MMD0)

This register is used to specify the operating mode (asynchronous, byte synchronous, or bit synchronous), set the auto-enable function, specify the expression for calculating the CRC, and specify the stop bit length for the asynchronous mode.

MSCI mode register 0 is reset under either of the following conditions:

- Hardware reset, or channel reset command



\*1 Reserved. These bits always read 0 and should be set to 0.

\*2 Reserved. If these settings are selected, correct operation is not guaranteed.

\*3 X indicates any value (0 or 1).

### Bits 7-5: PRTCL2-0 (protocol mode)

These bits specify the transmission protocol (transmission control procedure). Before changing the bit settings, these bits must be initialized by a channel reset command. If these bits are changed during operation, normal operation is not guaranteed.

PRTCL2	PRTCL1	PRTCL0	Function
0	0	0	Specifies asynchronous mode
0	0	1	Specifies byte synchronous (mono-sync) mode
0	1	0	Specifies byte synchronous (bi-sync) mode
0	1	1	Specifies byte synchronous (external synchronous) mode
1	0	0	Specifies bit synchronous HDLC mode
1	0	1	Specifies bit synchronous loop mode
1	1	0	Reserved
1	1	1	

#### Bit 4: AUTO (auto-enable)

This bit controls the modem control signals ( $\overline{\text{CTS}}\text{M}$ ,  $\overline{\text{DCD}}\text{M}$  and  $\overline{\text{RTS}}\text{M}$ ).

- Asynchronous/Byte/Bit synchronous mode

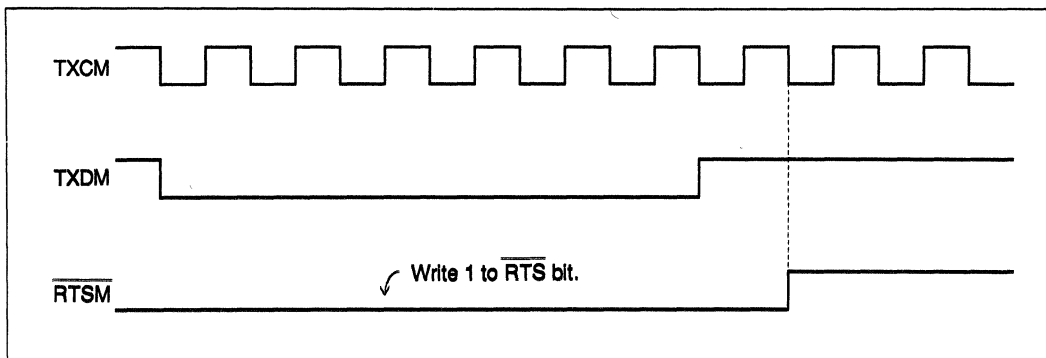
AUTO	Function
0	$\overline{\text{CTS}}\text{M}$ and $\overline{\text{DCD}}\text{M}$ are used as general-purpose inputs, and $\overline{\text{RTS}}\text{M}$ is used as a general-purpose output. $\overline{\text{CTS}}\text{M}$ , $\overline{\text{DCD}}\text{M}$ , and $\overline{\text{RTS}}\text{M}$ have no effect on MSCI transmission or reception.
1	Sets the auto-enable function. The $\overline{\text{CTS}}\text{M}$ , $\overline{\text{DCD}}\text{M}$ , and $\overline{\text{RTS}}\text{M}$ lines can be used as modem control signals for such as an RS-232C interface. For example, the $\overline{\text{CTS}}\text{M}$ input can be used to control transmission operations. When the $\overline{\text{CTS}}\text{M}$ input goes high, the transmitter sends the data in the transmit shift register*1 in the asynchronous mode, and then enters the idle state (maintains the TXDM line at high level). After this, no data is transferred from the transmit buffer to the transmit shift register. The $\overline{\text{DCD}}\text{M}$ input can be used to control reception operations. When $\overline{\text{DCD}}\text{M}$ is high, reception is prevented. If $\overline{\text{DCD}}\text{M}$ goes high during character assembly*2, the data being assembled is lost. However, the data in the receive buffer remains intact. The $\overline{\text{RTS}}\text{M}$ output is held at low level during transmission in the asynchronous mode. When not transmitting (TX disabled or in the idle state), the $\overline{\text{RTS}}\text{M}$ line outputs the value of the $\overline{\text{RTS}}$ bit in the MCTL.

\*1 The transmitter transmits one frame in the byte or bit synchronous mode.

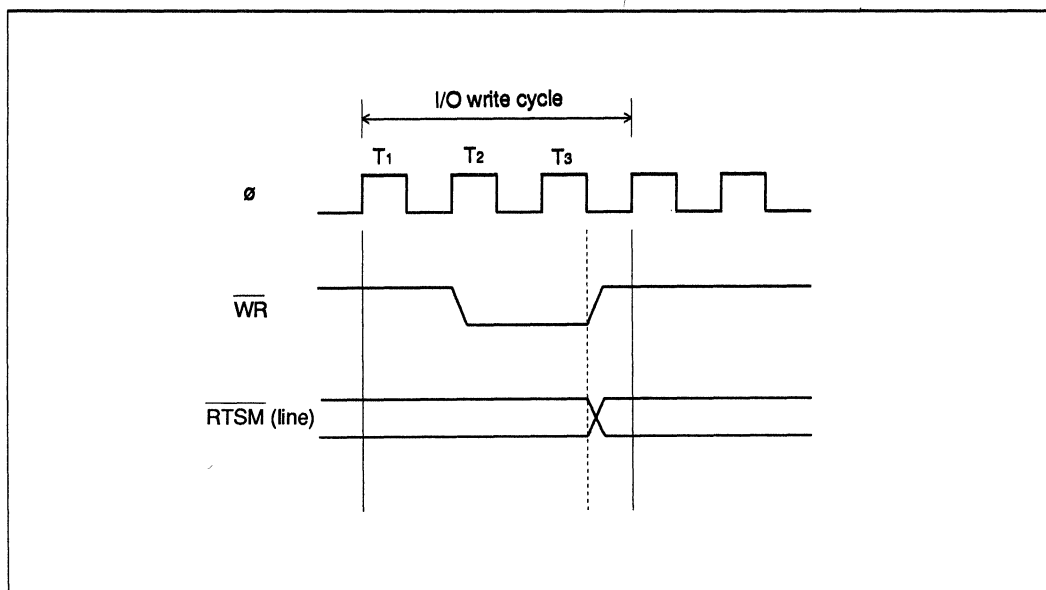
\*2 Character assembly implies sampling of received data and assembly of a character in the receive shift register.

Figures 4-4 (a) and (b) show the timing for modem control signal  $\overline{\text{RTS}}\text{M}$ . The  $\overline{\text{RTS}}\text{M}$  output during data write to the transmit buffer (MTRB) is provided on the falling edge of the T3 state. The  $\overline{\text{RTS}}\text{M}$  output is set to high level one clock cycle after the TXDM line has been set to mark after data transmission.





**Figure 4-4. (a) Modem Control Signal Timing**  
 (auto-enable, 5 bits/character, no parity and 1/1 clock mode)



**Figure 4-4. (b) Modem Control Signal Timing**

### Bit 3: Reserved

This bit always reads 0 and should be set to 0.

### Bit 2: CRCCC (CRC calculation code)

This bit specifies CRC code generation/detection in the byte synchronous or bit synchronous mode.

- Asynchronous mode

Reserved. This bit always reads 0 and should be set to 0.

- Byte synchronous/Bit synchronous mode

CRCCC	Function
0	CRC code generation/detection is not performed.
1	In the byte or bit synchronous mode, CRC calculation for transmission and reception is performed. Results of the CRC calculation for transmission are output as CRC code whereas results of the CRC calculation for reception are indicated by the CRCE bit in MST2. In the bit synchronous mode, FCS (CRC) are deleted without being transferred to the receive buffer.

### Bits 1-0: STOP1-0/CRC1-0 (Stop bit length/CRC calculation expression and initial value)

These bits specify the stop bit length in the asynchronous mode and the CRC calculation expression in the bit and byte synchronous mode.

- Asynchronous mode

STOP1	STOP0	Function
0	0	Stop bit length is 1
0	1	Stop bit length is 1.5
1	0	Stop bit length is 2
1	1	Reserved

When settings of these bits are changed, the newly specified stop bit length takes effect from the character currently transmitted.

- Byte synchronous mode, bit synchronous mode

<b>CRC1</b>	<b>Function</b>
0	CRC-16 ( $X^{16} + X^{15} + X^2 + 1$ ) is used for CRC calculations in the transmitter and receiver.
1	CRC-CCITT ( $X^{16} + X^{12} + X^5 + 1$ ) is used for CRC calculations in the transmitter and receiver.

<b>CRC0</b>	<b>Function</b>
0	Sets the CRC calculator reset value to all 0s.
1	Sets the CRC calculator reset value to all 1s.

#### 4.2.2 MSCI Mode Register 1 (MMD1)

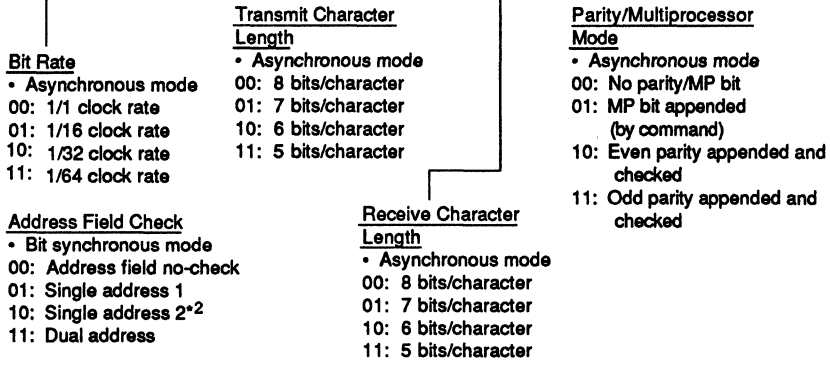
This register is used to specify the relationship between the transmit/receive data and transmit/receive clock, the transmit/receive character length, the parity/MP bit in the asynchronous mode, and the method for checking the address field in the bit synchronous mode.

The MSCI mode register 1 is reset under either of the following conditions:

- Hardware reset, or channel reset command

	7	6	5	4	3	2	1	0
Async	BRATE1	BRATE0	TXCHR1	TXCHR0	RXCHR1	RXCHR0	PMPM1	PMPM0
Byte Sync	-*1	-*1	-*1	-*1	-*1	-*1	-*1	-*1
Bit Sync	ADDRS1	ADDRS0						

Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0



\*1 Reserved. These bits always read 0 and should be set to 0.  
 \*2 This is a 4-bit address in bit synchronous loop mode. For details, see "Address Field Check" in section 4.3.3. "Bit Synchronous Mode" and 4.3.4 "Bit Synchronous Loop Mode."

**Bits 7-6: BRATE1-0/ADDRS1-0 (Bit rate/Address field check)**

These bits specify the relationships between the bit rate and the transmit/receive clock in the asynchronous mode and the method for checking the address field in bit synchronous mode. These bits are used for both transmission and reception.

- Asynchronous mode

For details, see section 4.3.1 "Asynchronous Mode."

BRATE1	BRATE0	Bit Rate
0	0	1/1 clock rate
0	1	1/16 clock rate
1	0	1/32 clock rate
1	1	1/64 clock rate

- Byte synchronous mode

Reserved. These bits always read 0 and should be set to 0.

- Bit synchronous mode

For details, see "Address Field Check" in section 4.3.3 "Bit Synchronous Mode."

ADDRS1	ADDRS0	Function
0	0	Skips the address field check
0	1	Sets single address 1
1	0	Sets single address 2 *1
1	1	Sets dual address

\*1 This specifies the 4-bit address mode in the bit synchronous loop mode. For details, see "Address Field Check" in section 4.3.4 "Bit Synchronous Loop Mode."

#### Bits 5-4: TXCHR1-0 (Transmit character length)

These bits specify the character length of the transmit data in the asynchronous mode. When these bits are changed during operation, the new character length applies to the next transmit character.

- Asynchronous mode

TXCHR1	TXCHR0	Transmit Character Length
0	0	8 bits/character
0	1	7 bits/character
1	0	6 bits/character
1	1	5 bits/character

- Byte synchronous/Bit synchronous mode

Reserved. These bits always read 0 and should be set to 0.

### Bits 3-2: RXCHR1-0 (Receive character length)

These bits specify the character length of receive data in the asynchronous mode. When these bits are changed during operation, the new character length applies to the next receive character.

- Asynchronous mode

RXCHR1	RXCHR0	Receive Character Length
0	0	8 bits/character
0	1	7 bits/character
1	0	6 bits/character
1	1	5 bits/character

- Byte synchronous/Bit synchronous mode

Reserved. These bits always read 0 and should be set to 0.

### Bits 1-0: PMPM1-0 (Parity/Multiprocessor mode)

These bits specify whether or not to use the parity check and multiprocessor (MP) mode in the asynchronous mode. When these bits are changed during operation, the new settings apply to the next transmit/receive character.

- Asynchronous mode

For details, see "Parity/MP Bit" in section 4.3.1 "Asynchronous Mode."

PMPM1	PMPM0	Function
0	0	Parity/MP bit not appended; parity check not performed
0	1	Specifies appending an MP bit (MP bit value is specified by command)*1
1	0	Specifies even parity and parity check performed
1	1	Specifies odd parity and parity check performed

\*1 See section 4.2.8 "MSCI Command Register."

- Byte synchronous/Bit synchronous mode

Reserved. These bits always read 0 and should be set to 0.

### 4.2.3 MSCI Mode Register 2 (MMD2)

This register is used to specify the transmission code type, the ratio of the advanced digital phase locked loop (ADPLL) operating clock to the bit rate, and the connection path between the transmit/receive data and the TXDM/RXDM lines.

MSCI mode register 2 is reset under the following conditions:

- Hardware or channel reset command

	7	6	5	4	3	2	1	0
Async	-*1	-*1	-*1	-*1	-*1	-*1		
Byte Sync	NRZFM	CODE1	CODE0	DRATE1	DRATE0	-*1	CNCT1	CNCT0
Bit Sync								
Read/Write	R/W	R/W	R/W	R/W	R/W	-	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

<p><u>NRZ or FM Select</u></p> <ul style="list-style-type: none"> <li>• Byte/Bit synchronous mode</li> <li>0: NRZ</li> <li>1: FM</li> </ul>	<p><u>Transmission Code Type</u></p> <ul style="list-style-type: none"> <li>• Byte/Bit synchronous mode</li> <li>• NRZ</li> <li>00: NRZ</li> <li>01: NRZI</li> <li>10: Reserved*2</li> <li>11: Reserved*2</li> <li>• FM</li> <li>00: Manchester</li> <li>01: FM1</li> <li>10: FM0</li> <li>11: Reserved*2</li> </ul>	<p><u>ADPLL Operating Clock/Bit Rate</u></p> <ul style="list-style-type: none"> <li>• Byte/Bit synchronous mode</li> <li>00: x8</li> <li>01: x16</li> <li>10: x32</li> <li>11: Reserved*2</li> </ul>	<p><u>Channel Connection</u></p> <ul style="list-style-type: none"> <li>00: Full duplex communications</li> <li>01: Auto echo</li> <li>10: Reserved*2</li> <li>11: Local loop back</li> </ul>
---	--	--	---

\*1 Reserved. These bits always read 0 and should be set to 0.

\*2 If these settings are selected, normal operation is not guaranteed.

### Bit 7: NRZFM (NRZ or FM Select)

This bit specifies the transmission code type (NRZ or FM) and is used in conjunction with CODE1-0 bits (see below). It specifies decode and encode types of MSCI. In the asynchronous mode only the NRZ type is available.

- Asynchronous mode

Reserved. This bit always reads 0 and should be set to 0.

- Byte synchronous/Bit synchronous mode

NRZFM	Function
0	Specifies NRZ transmission code type
1	Specifies FM transmission code type

### Bits 6-5: CODE1-0 (Transmission code type)

These bits are used in conjunction with the NRZFM bit (above) to specify the signal decoding and encoding type. In the asynchronous mode only the NRZ type is available.

- Asynchronous mode

Reserved. These bits always read 0 and should be set to 0.

- Byte synchronous/Bit synchronous mode

NRZFM		CODE1	CODE0	Function
0	NRZ	0	0	Specifies NRZ transmission code type
		0	1	Specifies NRZI transmission code type
		1	0	Reserved
		1	1	Reserved
1	FM	0	0	Specifies Manchester transmission code type
		0	1	Specifies FM1 transmission code type
		1	0	Specifies FM0 transmission code type
		1	1	Reserved



### Bits 4-3: DRATE1-0 (ADPLL operating clock/bit rate)

These bits specify the ratio of the ADPLL operating clock frequency to the bit rate in the byte or bit synchronous mode.

- Asynchronous mode

Reserved. These bits always read 0 and should be set to 0.

- Byte synchronous/Bit synchronous mode

DRATE1	DRATE0	Function
0	0	ADPLL operating clock frequency = bit rate $\times$ 8
0	1	ADPLL operating clock frequency = bit rate $\times$ 16
1	0	ADPLL operating clock frequency = bit rate $\times$ 32
1	1	Reserved

### Bits 1-0: CNCT1-0 (Channel connection)

CNCT1	CNCT0	Function
0	0	Specifies full duplex communications (normal operation).
0	1	Specifies auto-echo. In this mode, input data, via the RXDM line, is directly output to the TXDM line. This mode allows data reception, but not data transmission. The TXCM line echoes the RXCM line input.
1	0	Reserved
1	1	Specifies local loop-back mode. In this mode, the transmit shift register output is internally connected to the receive shift register input to directly receive the transmit data. Independent of the above operation, the TXDM line echoes the RXDM line input and the TXCM line echoes the RXCM line input.

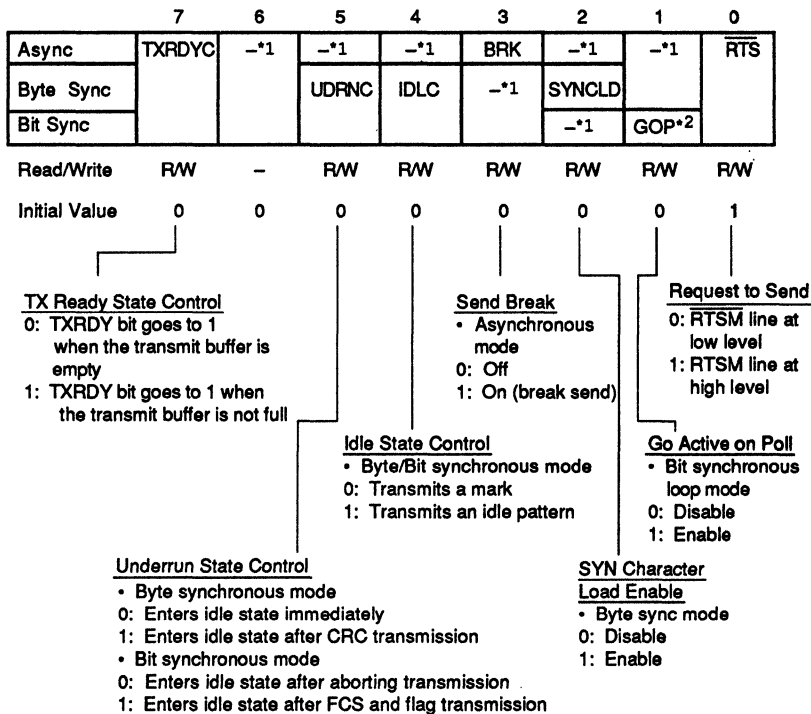
## 4.2.4 MSCI Control Register (MCTL)

This register is used to specify the relationship between the TXRDY bit and the transmit buffer status, the transmit operation upon underrun, an output pattern for the idle state in byte and bit synchronous modes, a break send for the asynchronous mode, a SYN character transfer from the data field to the receive buffer, and the RTS line output level.

The MSCI control register is reset under the following conditions:

- Hardware reset, or channel reset command

The BRK bit (bit 3) is also cleared by a TX reset command.



\*1 Reserved. These bits always read 0 and should be set to 0.

\*2 This bit is valid only in bit synchronous loop mode.

**Bit 7: TXRDYC (TX ready state control)**

This bit specifies the transmit buffer state which will set the TXRDY bit of MSCI status register 0 to 1.

- Asynchronous/Byte synchronous/Bit synchronous mode

<b>TXRDYC</b>	<b>Function</b>
0	The TXRDY bit goes to 1 when the transmit buffer is empty.
1	The TXRDY bit goes to 1 when the transmit buffer is not full. Use this setting for DMA transfer.

**Bit 6:** Reserved. This bit always reads 0 and should be set to 0.

**Bit 5: UDRNC (Underrun state control)**

This bit specifies the transmit operation to be executed when the underrun state is entered in the byte or bit synchronous mode.

- Asynchronous mode

Reserved. This bit always reads 0 and should be set to 0.

- Byte synchronous mode

<b>UDRNC</b>	<b>Function</b>
0	Enters the idle state immediately in the underrun state.
1	Enters the idle state after CRC code transmission in the underrun state.

- Bit synchronous mode

<b>UDRNC</b>	<b>Function</b>
0	Enters the idle state after aborting transmission in the underrun state.
1	Enters the idle state after FCS (CRC code) and flag transmission in the underrun state.

#### Bit 4: IDLC (Idle state control)

This bit specifies the TXDM line output when the idle state is entered in the byte or bit synchronous mode.

- Asynchronous mode

Reserved. This bit always reads 0 and should be set to 0.

- Byte synchronous mode and bit synchronous mode

<b>IDLC</b>	<b>Function</b>
0	Sets the TXDM line to the high level (mark) in the idle state.
1	Repeatedly transmits the 8-bit idle pattern in the MSCI idle pattern register (MIDL) in the idle state.

#### Bit 3: BRK (Send break)

This bit specifies whether or not to transmit a break in the asynchronous mode.

- Asynchronous mode

<b>BRK</b>	<b>Function</b>
0	Transmits no break (normal operation)
1	Transmits a break When this bit is set to 1, the TXDM line goes to low level (space) beginning from the next transmit clock falling edge. To transmit a break, this state must continue for two or more character cycles.

The BRK bit is cleared to 0 by a TX reset command.

For details on transmitting breaks, see "Break Transmission and Detection" in section 4.3.1 "Asynchronous Mode."

- Byte/Bit synchronous mode

Reserved. This bit always reads 0 and should be set to 0.

## Bit 2: SYNCLD (SYN character load enable)

This bit specifies whether or not to transfer the SYN character (in the data field) to the receive buffer in the byte synchronous mode. See section 4.3.2 "Byte Synchronous Mode."

In this case, the SYN character specified by the MSCI sync/address register 0 is valid.

- Asynchronous mode

Reserved. This bit always reads 0 and should be set to 0.

- Byte synchronous mode

SYNCLD	Function
0	Does not transfer the SYN character in the data field to the receive buffer, but deletes it.
1	Transfers the SYN character in the data field to the receive buffer.

- Bit synchronous mode

Reserved. This bit always reads 0 and should be set to 0.

## Bit 1: GOP (Go active on poll)

This bit specifies whether or not to send transmit buffer data when a GA pattern is received in bit synchronous loop mode. For information about the GA pattern, see section 4.3.4 "Bit Synchronous Loop Mode."

- Asynchronous/Byte synchronous mode

Reserved. This bit always reads 0 and should be set to 0.

- Bit synchronous loop mode

GOP	Function
0	Does not send transmit buffer data when a GA pattern is received.
1	Sends transmit buffer data when a GA pattern is received.

### Bit 0: $\overline{\text{RTS}}$ (Request to send)

This bit specifies the  $\overline{\text{RTSM}}$  line output level.

- Asynchronous/Byte synchronous/Bit synchronous mode

$\overline{\text{RTS}}$	Function
0	The $\overline{\text{RTSM}}$ line level goes low.
1	The $\overline{\text{RTSM}}$ line level goes high.

When auto-enable has been selected (AUTO bit of the MMD0 register is 1) in the asynchronous mode, the  $\overline{\text{RTSM}}$  line goes low in the transmit operation, regardless of the  $\overline{\text{RTS}}$  bit setting.

### 4.2.5 MSCI RX Clock Source Register (MRXS)

This register is used to specify the receive clock and the baud rate of the baud rate generator (BRG) in the receiver.

The MSCI RX clock source register is reset under the following conditions:

- Hardware reset, or channel reset command

	7	6	5	4	3	2	1	0
Async	—*1	RXCS2	RXCS1	RXCS0	RXBR3	RXBR2	RXBR1	RXBR0
Byte Sync								
Bit Sync								
Read/Write	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

<p><u>Receive Clock Source</u></p> <p>000: RXCM line input</p> <p>010: RXCM line input (noise suppression)</p> <p>100: Internal baud rate generator (BRG) output</p> <p>110: ADPLL output (BRG output for ADPLL operating clock)</p> <p>111: ADPLL output (RXCM line input for ADPLL operating clock)</p> <p>Others: Reserved*2</p>	<p><u>Receive Baud Rate</u></p> <p>• Clock division ratio</p> <p>0000: 1/1</p> <p>0001: 1/2</p> <p>0010: 1/4</p> <p>0011: 1/8</p> <p>0100: 1/16</p> <p>0101: 1/32</p> <p>0110: 1/64</p> <p>0111: 1/128</p> <p>1000: 1/256</p> <p>1001: 1/512</p> <p>Others: Reserved*2</p>
---	--

\*1 Reserved. This bit always read 0 and should be set to 0.

\*2 Reserved. When these settings are selected, normal operation is not guaranteed.

**Bit 7:** Reserved. This bit always reads 0 and should be set to 0.

**Bits 6-4: RXCS2-0 (Receive clock source)**

These bits specify the receive clock source.

- Asynchronous/Byte synchronous/Bit synchronous mode

<b>RXCS2</b>	<b>RXCS1</b>	<b>RXCS0</b>	<b>Function</b>
0	0	0	Specifies the RXCM input as the receive clock. The noise suppressor does not function for the receive clock and receive data.
0	1	0	Specifies the RXCM input as the receive clock. The noise suppressor of the ADPLL functions for both the receive clock and receive data.
1	0	0	Specifies the internal BRG as the receive clock. The receive clock generated by the BRG is output from the RXCM line.
1	1	0	Specifies the clock extracted by the ADPLL as the receive clock. The BRG output is used as the ADPLL operating clock. At this time, the receive data noise is suppressed. The receive clock extracted by the ADPLL is output from the RXCM line.
1	1	1	Specifies the clock extracted by the ADPLL as the receive clock. The RXCM input line is used as the ADPLL operating clock. Receive data noise is suppressed.
<b>Others</b>			<b>Reserved</b>

### Bits 3-0: RXBR (Receiver baud rate)

These bits, used in conjunction with the MSCI time constant register (MTMC) setting, specify the baud rate (when baud rate generation is used in the receiver). For details, see section 4.6 "Baud Rate Generator."

- Asynchronous/Byte synchronous/Bit synchronous mode

RXBR3	RXBR2	RXBR1	RXBR0	Division ratio
0	0	0	0	1/1
0	0	0	1	1/2
0	0	1	0	1/4
0	0	1	1	1/8
0	1	0	0	1/16
0	1	0	1	1/32
0	1	1	0	1/64
0	1	1	1	1/128
1	0	0	0	1/256
1	0	0	1	1/512
1	0	1	0	
	:			Reserved
	:			
1	1	1	1	



## 4.2.6 MSCI TX Clock Source Register (MTXS)

This register is used to specify the transmit clock source and the baud rate in the transmitter BRG.

The MSCI TX clock source register is reset under the following conditions:

- Hardware reset, or channel reset command

	7	6	5	4	3	2	1	0
Async	-*1	TXCS2	TXCS1	TXCS0	TXBR3	TXBR2	TXBR1	TXBR0
Byte Sync								
Bit Sync								
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

<p><u>Transmit Clock Source</u></p> <p>000: TXCM line input          100: Internal baud rate generator (BRG) output          110: Receiver clock          Others: Reserved*2</p>	<p><u>Transmit Baud Rate</u></p> <p>• Clock division ratio</p> <p>0000: 1/1          0001: 1/2          0010: 1/4          0011: 1/8          0100: 1/16          0101: 1/32          0110: 1/64          0111: 1/128          1000: 1/256          1001: 1/512          Others: Reserved*2</p>
--	---

\*1 Reserved. This bit always reads 0 and should be set to 0.

\*2 Reserved. When these settings are selected, normal operation is not guaranteed.

**Bit 7:** Reserved. This bit always reads 0 and should be set to 0.

**Bits 6-4: TXCS2-0 (Transmit clock source)**

These bits are used to specify the transmit clock source.

- Asynchronous/Byte synchronous/Bit synchronous mode

TXCS2	TXCS1	TXCS0	Function
0	0	0	Specifies the TXCM input as the transmit clock.
1	0	0	Specifies the internal BRG output as the transmit clock. The transmit clock generated by the BRG is output from the TXCM line.
1	1	0	Specifies the receive clock as the transmit clock. Use this specification in the following cases: <ul style="list-style-type: none"> <li>• When using the clock extracted by the ADPLL as the transmit clock</li> <li>• When using the receive clock as the transmit clock in bit synchronous loop mode</li> </ul>
Others			Reserved

### Bits 3-0: TXBR (Transmit baud rate)

These bits are used in conjunction with the time constant register (MTMC) to specify the baud rate (when the baud rate generator is used in the transmitter). For details, see section 4.6 "Baud Rate Generator."

- Asynchronous /Byte synchronous/Bit synchronous mode

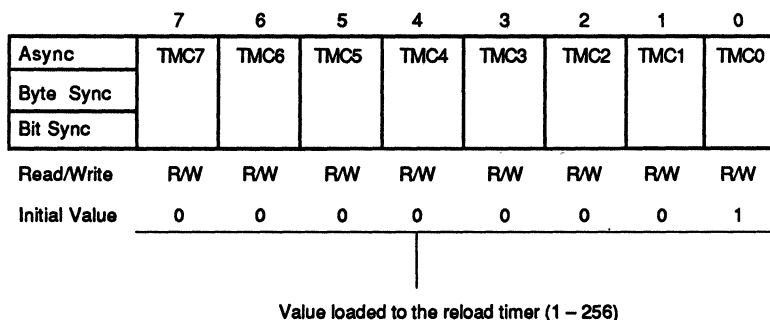
TXBR3	TXBR2	TXBR1	TXBR0	Division Ratio
0	0	0	0	1/1
0	0	0	1	1/2
0	0	1	0	1/4
0	0	1	1	1/8
0	1	0	0	1/16
0	1	0	1	1/32
0	1	1	0	1/64
0	1	1	1	1/128
1	0	0	0	1/256
1	0	0	1	1/512
1	0	1	0	
		⋮		Reserved
1	1	1	1	

#### 4.2.7 MSCI Time Constant Register (MTMC)

This register is used to specify a value (1 – 256) to be loaded to the reload timer in the internal baud rate generator (BRG). For details, see section 4.6 "MSCI Baud Rate Generator."

The MSCI time constant register is reset under the following conditions:

- Hardware reset, or channel reset command



#### Bits 7-0: TMC 7-0 (time constant)

- Asynchronous/Byte synchronous/Bit synchronous mode

These bits specify the value (1 – 256) to be loaded to the reload timer of the internal baud rate generator. If zero is specified, the value is assumed to be 256. These bits are used in combination with the TXBR3-0 bits of the MSCI TX clock source register (MTXS) and the RXBR3-0 bits of the RX clock source register (MRXS) to determine the BRG output frequency for transmission and reception.

The generated clock frequency can be calculated by the following expression:

$$f_{BRG} = \frac{f_{\phi}}{TMC} + 2BR$$

- |                    |  |
|--------------------|--|
| f <sub>BRG</sub> : | Transmit (receive) BRG output frequency  |
| f <sub>φ</sub> :   | CPU clock frequency  |
| TMC:               | Time constant register set value (1 – 256)   |
| BR:                | Value (0 – 9) set in TXBR3-0 bits of TX clock source register or RXBR3-0 bits of RX clock source register. |

## 4.2.8 MSCI Command Register (MCMD)

This register is used to specify the command for MSCI transmission/reception control. The register is a write-only register and always reads 00H.

	7	6	5	4	3	2	1	0
Async								
Byte Sync	–*1	–*1	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
Bit Sync								
Read/Write	–	–	W	W	W	W	W	W
Initial Value	–	–	–	–	–	–	–	–

Command

- |  |  |  |
|--|--|--|
| <ul style="list-style-type: none"> <li>• Transmit Commands</li> <li>000001: TX reset</li> <li>000010: TX enable</li> <li>000011: TX disable</li> <li>000100: TX CRC initialization</li> <li>000101: Exclusion from TX<br/>CRC calculation</li> <li>000110: End of message</li> <li>000111: Abort transmission</li> <li>001000: MP bit on</li> <li>001001: TX buffer clear</li> <li>[Others: Reserved*2]</li> </ul> | <ul style="list-style-type: none"> <li>• Receive Commands</li> <li>010001: RX reset</li> <li>010010: RX enable</li> <li>010011: RX disable</li> <li>010100: RX CRC initialization</li> <li>010101: Message reject</li> <li>010110: Search MP bit</li> <li>010111: Exclusion from RX<br/>CRC calculation</li> <li>011000: Forcing RX CRC<br/>calculation</li> </ul> | <ul style="list-style-type: none"> <li>• Other Commands</li> <li>100001: Channel reset</li> <li>110001: Enter search mode</li> <li>000000: No operation</li> </ul> |
|--|--|--|

\*1 Reserved. These bits always read 0 and should be set to 0.

\*2 When these reserved values are specified, normal operation is not guaranteed.

**Bits 7-6:** Reserved. These bits always read 0 and should be set to 0.

### Bits 5-0: CMD 5-0 (Command)

- Asynchronous/Byte synchronous/Bit synchronous mode

Transmit, receive, and other commands are specified in bits 5-0. Tables 4-2 to 4-4 summarize these command settings and functions.

(1) Transmit commands

**Table 4-2. Transmit Commands**

<b>Command Name (set value)</b>	<b>Function</b>
TX reset (01H)	Immediately places the transmitter in the TX disable state (the transmit line goes to mark). The transmit buffer is cleared, the transmit status in the MSCI status register 3-0 (MST 3-0) is reset, and the BRK bit of the MCTL is reset. Other registers are not affected.
TX enable (02H)	Places the transmitter in the idle state when the transmitter is in the TX disable state. For auto-enable operation, see the explanation regarding the AUTO bit in section 4.2.1 "MSCI Mode Register 0 (MMD0)."
TX disable (03H)	The transmitter enters the TX disable state after sending the transmit buffer data in the asynchronous mode or after transmitting one frame in byte or bit synchronous mode (the TXRDY bit of MSCI status register 0 immediately goes to 0.)
TX CRC initialization (04H)	Sets the transmitter CRC calculator to the initial value specified by the CRC0 bit of MMD0. After the command is issued, the CRC calculator is initialized when the first transmit character is transferred to the transmit shift register. Use this command in byte or bit synchronous mode.
TX CRC calculation exclusion (05H)	Excludes a specific character from the transmit CRC calculation. This command is valid only for the first character transferred to the transmit shift register after the command is issued. If the first character needs to be excluded, the command must be issued during transmission of the SYN character preceding the character to be excluded.*1 Command operation is not guaranteed for modes other than byte synchronous.
End-of-message (06H)	Specifies the first transmit character transferred to the transmit buffer after the command is issued as the last character of the frame. The transmitter sends the character that specifies the end of a message. It then transmits the CRC code in byte synchronous mode or sequentially transmits the FCS (CRC code) and flag in bit synchronous mode.

\*1 If SYN character transmission timing is not explicit, write a SYN character to the MSCI TX/RX buffer register before the first character, and then issue the command during the SYN character transmission.

**Table 4-2. Transmit Commands (cont.)**

<b>Command Name (set value)</b>	<b>Function</b>
Abort transmission (07H)	Immediately transmits an 8-bit abort pattern 11111111 and clears the transmit buffer. Use this command in the bit synchronous mode.
MP bit on (08H)	Sets the transmit data MP bit to 1, and then transmits a character. This command is valid only for the first character transferred to the transmit buffer after the command is issued. Operation of this command is not guaranteed in modes except asynchronous mode.
TX buffer clear (09H)	Clears the transmit buffer. The buffer contents are lost. Other registers are not affected.

**(2) Receive commands****Table 4-3. Receive Commands**

<b>Command Name (set value)</b>	<b>Function</b>
RX reset (11H)	Halts the receive shift register and places the receiver in the RX disable state. The receive buffer is cleared and the receive status values stored in the MSCI status registers 3-0 (MST 3-0) are reset. Other registers are not affected.
RX enable (12H)	Places the receiver into the start bit search state in the asynchronous mode, the SYN1 wait state in byte synchronous mode, and the flag wait state in bit synchronous mode. If the receiver is in the enable state, this command is invalid. For operation in the auto-enable, see the explanation regarding the AUTO bit in section 4.2.1 "MSCI Mode Register 0 (MMD0)."
RX disable (13H)	Halts the receive shift register and places the receiver in the RX disable state. The receive shift register contents are lost, but the receive buffers are not affected.

**Table 4-3. Receive Commands (cont.)**

<b>Command Name (set value)</b>	<b>Function</b>
RX CRC initialization (14H)	<p>Sets the receiver CRC calculator to the initial value specified by the CRC0 bit of MMD0. After issuing this command the CRC calculator is initialized when the first receive character is transferred to the receive shift register.</p> <p>Use this command in byte and bit synchronous modes.</p>
Message reject (15H)	<p>In byte synchronous mode, the receiver re-establishes character synchronization immediately after this command is issued.</p> <p>In bit synchronous mode, this command prevents the current data frame from being transferred to the receive buffer. Data transfer to the receive buffer resumes beginning with the next frame.</p>
Search MP bit (16H)	<p>Prevents the receive character with MP bit = 0 from being loaded into the receive buffer. This command remains valid until a character with MP bit = 1 is received.</p> <p>If necessary, re-issue this command after receiving a character with MP bit = 1.</p> <p>This command is valid only in asynchronous mode.</p>
RX CRC calculation exclusion (17H)	<p>Excludes a specific character from the receiver CRC calculation.</p> <p>This command must be issued within 8 bit cycles after the character to be excluded from the CRC calculation enters the receive buffer.</p> <p>Command operation is not guaranteed in modes other than byte synchronous.</p>
Forcing RX CRC calculation (18H)	<p>Forcibly starts CRC calculation using the 8-bit data in the receive delay register.</p> <p>In byte synchronous mode, issue this command after the second byte of the CRC code has entered the receive buffer. This allows CRC calculation to be completed even when the receive clock is halted after CRC code reception.</p> <p>CRC error status is valid 15 system clock cycles after issuing this command and it remains valid until the next data enters the receive buffer.</p>

### (3) Other commands

**Table 4-4. Other Commands**

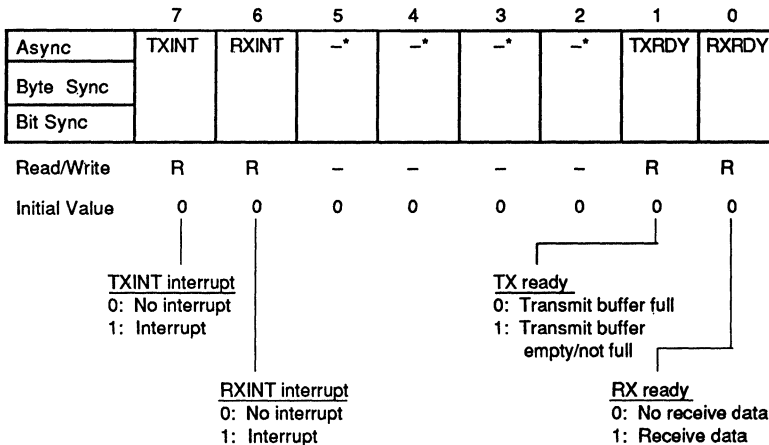
Command Name (set value)	Function
Channel reset (21H)	Resets all registers to their initial values and places the receiver in the disable state (the receive buffer is cleared).
Enter search mode (31H)	Places the ADPLL in the search mode. For FM code transmission, synchronization between the extracted receive clock and receive data can be established in a single operation (in this case, the SRCH bit of MSCI status register 1 goes to 1). For details, see section 4.5 "ADPLL"
No Operation (00H)	The transmitter and receiver continue the current operation.

#### 4.2.9 MSCI Status Register 0 (MST0)

This register is used to indicate the status of interrupts (TXINT and RXINT) and transmit/receive buffer. When a bit (except bits 5-2) goes to 1, an interrupt is generated (if enabled).

MSCI status register 0 (MST0) is reset under the following conditions:

- Hardware reset, or channel reset command
- System stop mode



\* Reserved. These bits always read 0.



### Bit 7: TXINT (TXINT interrupt)

This bit indicates whether or not the TXINT internal interrupt has been set. The TXINT interrupt is enabled when this bit and the TXINTE bit of MIE0 are both 1.

- Asynchronous/Byte synchronous/Bit synchronous mode

TXINT	Function
0	Indicates that a TXINT interrupt has not occurred.
1	Indicates that a TXINT interrupt has occurred. That is, the TXINT bit is set under the following conditions: (1) The UDRNE bit is set to 1 and an underrun error has occurred, or (2) The IDLE bit is set to 1 and the idle state is entered, or (3) The CCTSE bit is set to 1 and the $\overline{\text{CTSM}}$ line level is changed. This bit is set to 1 under the following conditions: $\text{TXINT} = \text{UDRN} \cdot \text{UDRNE} + \text{IDL} \cdot \text{IDLE} + \text{CCTS} \cdot \text{CCTSE}$ (UDRN, IDL, CCTS: Bits 7, 6, and 3 of MSCI status register 1 UDRNE, IDLE, CCTSE: Bits 7, 6, and 3 of MSCI interrupt enable register 1)

### Bit 6: RXINT (RXINT interrupt)

This bit indicates whether or not the internal RXINT interrupt has been set. Internal RXINT interrupt is enabled when this bit and the RXINTE bit of MIE0 are both equal to 1.

- Asynchronous/Byte synchronous/Bit synchronous mode

RXINT	Function
0	Indicates that an RXINT interrupt has not occurred.
1	Indicates that an RXINT interrupt has occurred. The RXINT bit is set under the following conditions: (1) The SYNCDE/FLGDE bit is set and a SYN character or flag has been detected, or (2) The CDCDE bit is set and the $\overline{\text{DCDM}}$ line level is changed, or (3) The BRKDE/ABTDE/GAPDE bit is set and a break start/abort or GA pattern is detected, or (4) The BRKEE/IDLDE bit is set and a break end or idle start is detected, or (5) The EOME bit is set and the receive frame ends, or

RXINT	Function
	(6) The PMPE/SHRTE bit is set, and parity/MP bit is set or a short frame is detected, or
	(7) The PEE/ABTE bit is set and a parity error or abort frame is detected, or
	(8) The FRMEE/RBITE bit is set and a framing error or residue bit frame is detected, or
	(9) The OVRNE bit is set and an overrun error is detected, or
	(10) The CRCEE bit is set and a CRC error is detected, or
	(11) The EOMFE bit is set, the receive frame has ended, and the last character has been read from the receive buffer.
	 This bit is set to 1 under the following conditions:
	 $\text{RXINT} = (\text{SYNCD}/\text{FLGD}) \cdot (\text{SYNCDE}/\text{FLGDE}) + \text{CDCD} \cdot \text{CDCDE} + (\text{BRKD}/\text{ABTD}/\text{GAPD}) \cdot (\text{BRKDE}/\text{ABTDE}/\text{GAPDE}) + (\text{BRKE}/\text{IDLD}) \cdot (\text{BRKEE}/\text{IDLDE}) + \text{EOM} \cdot \text{EOME} + (\text{PMP}/\text{SHRT}) \cdot (\text{PMPE}/\text{SHRTE}) + (\text{PE}/\text{ABT}) \cdot (\text{PEE}/\text{ABTE}) + (\text{FRME}/\text{RBIT}) \cdot (\text{FRMEE}/\text{RBITE}) + \text{OVRN} \cdot \text{OVRNE} + \text{CRCE} \cdot \text{CRCEE} + \text{EOMF} \cdot \text{EOMFE}$
	SYNCD/FLGD, CDCD, BRKD/ABTD/GAPD, BRKE/IDLD: Bits 4, 2, 1 and 0 of MSCI status register 1
	EOM, PMP/SHRT, PE/ABT, FRME/RBIT, OVRN, CRCE: Bits 7-2 of MSCI status register 2
	EOMF: Bit 7 of MSCI frame status register
	SYNCDE/FLGDE, CDCDE, BRKDE/ABTDE/GAPDE, BRKEE/IDLDE: Bits 4, 2, 1 and 0 of MSCI interrupt enable register 1
	EOME, PMPE/SHRTE, PEE/ABTE, FRMEE/RBITE, OVRNE, CRCEE: Bits 7-2 of MSCI interrupt enable register 2
	EOMFE: Bit 7 of MSCI frame interrupt enable register

**Bits 5-2:** Reserved. These bits always read 0.

**Bit 1: TXRDY (TX ready)**

This bit indicates the transmit buffer status.

When the transmitter is enabled and the transmit buffer is empty (TXRDYC bit = 0)/non-full (TXRDYC = 1), the TXRDY bit is set. Otherwise, the TXRDY bit is cleared. This means that the transmit buffer can be written only while the TXRDY bit is 1.

When the TXRDY bit and the TXRDYE bit of the MSCI interrupt enable register 0 are both set to 1, a TXRDY interrupt request is issued to the CPU. Also, when the TXRDY bit is set, a DMA request is issued to the on-chip DMAC. For details, see section 4.8.1 "Serial Data Transfer by the CPU and Internal DMAC."

- Asynchronous/Byte synchronous/Bit synchronous mode

<b>TXRDY</b>	<b>Function</b>
0	This bit is cleared to 0 when the transmit buffer is full in the TX enable state. It is also cleared when the TX disable state is entered or when an underrun error is generated.
1	This bit is set when the transmit buffer is empty or not full in TX enable state. The set condition is specified by the TXRDYC bit of MSCI control register.

#### **Bit 0: RXRDY (RX ready)**

This bit indicates the receive buffer status.

This bit is set to 1 when receive data remains in the receive buffer, regardless of the RX enable or RX disable status.

A RXRDY interrupt request is issued to the CPU when the RXRDY bit and RXRDYE bit of the MSCI interrupt enable register 0 are both set to 1. A DMA request is issued to the on-chip DMAC when the RXRDY bit is set to 1. For details, see section 4.8.1 "Serial Data Transfer by the CPU and DMAC."

- Asynchronous/Byte synchronous/Bit synchronous mode

<b>RXRDY</b>	<b>Function</b>
0	Indicates that all receive data has been read from the receive buffer.
1	Indicates that receive data remains in the receive buffer.

## 4.2.10 MSCI Status Register 1 (MST1)

This register is used to specify information such as the start/stop break sequence in the asynchronous mode, underrun errors and the SYN pattern in the byte synchronous mode, underrun errors, flags, aborts, the start of GA patterns and idle sequences in the bit synchronous mode, transmitter idle status, and CTSM and DCDM input level changes.

The bits of MST1 are reset under the following conditions:

- Bits 7, 4, 3, 2, 1 and 0 are reset by writing 1 to them.
- A TX reset command resets bits 7, 6, and 3.
- An RX reset command resets bits 4, 2, 1, and 0.
- A channel reset command or the system stop mode resets all bits.

The IDL bit is cleared by writing data to the transmit buffer.

When any bit is set in this register, a CPU interrupt is generated (if enabled).

	7	6	5	4	3	2	1	0
Async	—*1	IDL	—*1	—*1	CCTS	CDCD	BRKD	BRKE
Byte Sync	UDFN			SYNCD			—*1	—*1
HDLG				FLGD			ABTD/ GAPD*2	IDL
Loop								

Read/Write    R/W    R    —    R/W    R/W    R/W    R/W    R/W

Initial Value    0    0    0    0    0    0    0    0

### Underrun Error

- Byte/Bit synchronous mode
- 0: No underrun detected
- 1: Underrun detected

### SYN Pattern Detection

- Byte synchronous mode
- 0: No pattern detected
- 1: Pattern detected

### Flag Detection

- Bit synchronous mode
- 0: No flag detected
- 1: Flag detected

### Transmitter Idle Status

- 0: Not idle
- 1: Idle

### DCDM Line Level Change

- 0: Not changed
- 1: Changed

### CTSM Line Level Change

- 0: Not changed
- 1: Changed

### Break End

- Asynchronous mode
- 0: Break sequence end not detected
- 1: Break sequence end detected

### Idle Start Detection

- Bit synchronous mode
- 0: Idle sequence start not detected
- 1: Idle sequence start detected

### Break Detection

- Asynchronous mode
- 0: Break sequence starts not detected
- 1: Break sequence starts detected

### Abort Detection/GA Pattern Detection

- Bit synchronous mode
- 0: Abort sequence start/GA pattern start not detected
- 1: Abort sequence start/GA pattern start detected

\*1 Reserved. These bits always read 0. These bits may be set to 0 or 1.

\*2 This bit can be accessed only in bit synchronous loop mode.

### Bit 7: UDRN (Underrun error)

This bit indicates an underrun error in byte and bit synchronous modes. (In the asynchronous mode, underrun errors do not occur.) This bit is cleared by writing 1 to this bit position.

- Asynchronous mode

Reserved. This bit always reads 0 and can be set to 0 or 1.

- Byte/Bit synchronous mode

UDRN	Function
0	Indicates that an underrun error has not occurred.
1	Indicates that an underrun error has occurred.

### Bit 6: IDL (Transmitter idle status)

This bit indicates whether or not the MSC1 transmitter is in the idle state. This bit is cleared by writing the transmit data to the transmit buffer.

- Asynchronous/Byte synchronous/Bit synchronous mode

IDL	Function
0	Indicates that the transmitter is not in the idle state.
1	Indicates that the transmitter is in the idle state.

Bit 5: Reserved. This bit always reads 0 and can be set to 0 or 1.

### Bit 4: SYNCDFLGD (SYN pattern detection/flag detection)

This bit indicates whether or not synchronization has been established in byte or bit synchronous mode. This bit is cleared by writing 1 to this bit position.

- Asynchronous mode

Reserved. This bit always reads 1 and can be set to 0 or 1.

- Byte synchronous mode

<b>SYNCD</b>	<b>Function</b>
0	Indicates that synchronization has not been established.
1	Indicates that synchronization has been established (SYN pattern detection in mono-sync or bi-sync mode; or by the SYNC line in the external synchronous mode).

- Bit synchronous mode

<b>FLGD</b>	<b>Function</b>
0	Indicates that synchronization has not been established.
1	Indicates that synchronization has been established (flag pattern detection).

### Bit 3: CCTS ( $\overline{\text{CTSM}}$ line level change)

This bit indicates whether or not the  $\overline{\text{CTSM}}$  line level has changed. This bit is cleared by writing 1 to this bit position.

- Asynchronous/Byte synchronous/Bit synchronous mode

<b>CCTS</b>	<b>Function</b>
0	Indicates that the $\overline{\text{CTSM}}$ line level has not changed.
1	Indicates that the $\overline{\text{CTSM}}$ line level has changed.

### Bit 2: CDCD ( $\overline{\text{DCDM}}$ line level change)

This bit indicates whether or not the  $\overline{\text{DCDM}}$  line level has changed. This bit is cleared by writing 1 to this bit position.

- Asynchronous/Byte synchronous/Bit synchronous mode

<b>CDCD</b>	<b>Function</b>
0	Indicates that the $\overline{\text{DCDM}}$ line input level has not changed.
1	Indicates that the $\overline{\text{DCDM}}$ line input level has changed.

**Bit 1: BRKD/ABTD/GAPD****(Break detection/abort detection/GA pattern detection)**

This bit signals the start of a break sequence (a space state) in the asynchronous mode, the detection of an abort in the bit synchronous HDLC mode, and the detection of a GA pattern in the bit synchronous loop mode. This bit is cleared by writing 1 to this bit position.

- Asynchronous mode

<b>BRKD</b>	<b>Function</b>
0	Indicates that the start of a break has not been detected.
1	Indicates that the start of a break has been detected.

- Byte synchronous mode

Reserved. This bit always reads 0 and can be set to 0 or 1.

- Bit synchronous mode

**ABTD/**

<b>GAPD</b>	<b>Function</b>
0	Indicates that neither an abort nor a GA pattern have been detected.
1	Indicates that an abort has been detected in the bit synchronous HDLC mode or that a GA pattern has been detected in the bit synchronous loop mode.

**Bit 0: BRKE/IDLD (Break end/idle start detection)**

This bit signals the end of a break in the asynchronous mode and the start of an idle state in the bit synchronous mode. The bit is cleared by writing 1 to this bit position.

- Asynchronous mode

<b>BRKE</b>	<b>Function</b>
0	Indicates that the end of a break has not been detected.
1	Indicates that the end of a break has been detected.

- Byte synchronous mode

Reserved. This bit always reads 0 and can be set to 0 or 1.

- Bit synchronous mode

<b>IDLD</b>	<b>Function</b>
0	Indicates that the start of an idle state has not been detected.
1	Indicates that the start of an idle state has been detected.

#### 4.2.11 MSCI Status Register 2 (MST2)

This register is used to indicate status information such as parity/MP bit value, parity error detection and framing error detection in the asynchronous mode, CRC error detection in the byte synchronous mode, the detection of a receive frame end, short frame, abort stop frame, residue bit frame, and CRC error in the bit synchronous mode, as well as overrun errors.

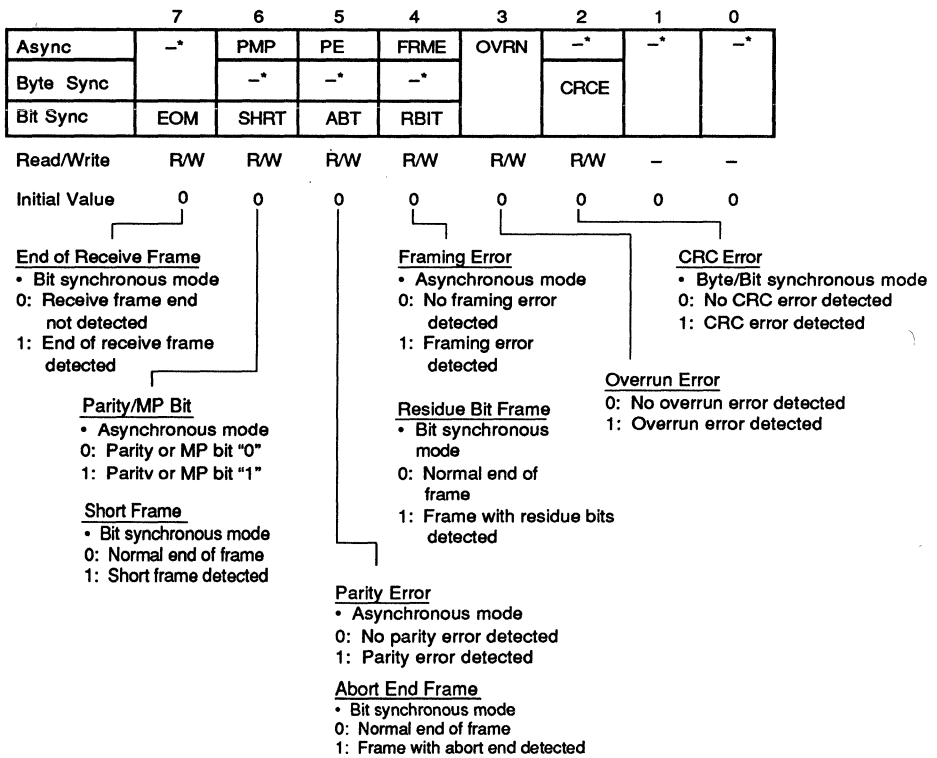
This register is at the top of the 3-stage status FIFO that corresponds to the receive buffer (see figure 4-2). Once a bit is set, it will not be reset by a status FIFO change. For the CRCE bit clear conditions, see Bit 2: CRCE in this section. The PMP bit is updated when the next receive character is ready to be read.

The bits in MST2 are reset under the following conditions:

- When 1 is written to a bit position, the bit is reset
- An RX or channel reset command resets all bits
- All bits are reset in the system stop mode
- All bits in this register are reset when data is transferred to the MSCI frame status register (MFST) (See section 4.2.13 "MSCI Frame Status Register.")

When any bit in this register is set, an interrupt is generated to the CPU (if enabled).





\* Reserved. These bits always read 0 and can be set to 0 or 1.

### Bit 7: EOM (End of receive frame)

This bit signals the end of a receive frame in the bit synchronous mode. The bit is cleared by writing 1 to this bit position.

- Asynchronous/Byte synchronous mode

Reserved. This bit always reads 0 and can be set to 0 or 1.

- Bit synchronous mode

This bit signals the end of the receive frame. When the CRCCC bit in the MSC1 mode register 0 (MMD0) is 1, the EOM bit is set to 1 by the last character in the I field of the receive frame. When the CRCCC bit in MMD0 is 0, the EOM bit is set to 1 by the last character of FCS. Also, when the receive frame end status indicates either a short frame, residue bit frame or abort, the EOM bit is set.

<b>EOM</b>	<b>Function</b>
0	Indicates that the receive frame has not ended.
1	Indicates that the receive frame has ended.

#### **Bit 6: PMP/SHRT (Parity/MP bit/short frame)**

This bit is used to select parity/MP bit value in asynchronous mode, and short frame detection in bit synchronous mode. This bit is cleared by writing 1 to this bit position.

- Asynchronous mode

This bit indicates the status of the parity bit, MP bit or receive character MSB under the following conditions:

Parity bit status: When MMD0 register PMPM1-0 = 10 or 11

MP bit status: When PMPM1-0 = 01

Receive character MSB status: When PMPM1-0 = 00

The PMP bit is updated when the next receive character is available to be read.

<b>PMP</b>	<b>Function</b>
0	The parity bit, MP bit or receive character MSB is 0.
1	The parity bit, MP bit or receive character MSB is 1.

- Byte synchronous mode

Reserved. This bit always reads 0 and can be set to 0 or 1.

- Bit synchronous mode

The SHRT bit indicates a short frame detection under the following conditions:

While the CRCCC bit of the MMD0 register is 1, this bit is set to 1 corresponding to the last character in the I field when the receive frame is short and a part of the data is sent to the receive buffer. While the CRCCC bit is 0, this bit is set to 1 by the last character of FCS.

However, even if a short frame is detected, this bit is not set until data is sent to the receive buffer.

When the SHRT bit is set, the EOM bit is also set.

<b>SHRT</b>	<b>Function</b>
0	Indicates that a short frame has not been detected.
1	Indicates that a short frame has been detected and that data has been sent to the receive buffer.

For details, see "Short frame detection" in section 4.3.3 "Bit Synchronous Mode."

#### **Bit 5: PE/ABT (Parity error/abort end frame)**

This bit indicates detection of a parity error in the asynchronous mode or an abort end frame in the bit synchronous mode.

This bit is cleared by writing 1 to this bit position.

- Asynchronous mode

<b>PE</b>	<b>Function</b>
0	Indicates no parity error
1	Indicates a parity error

Once this bit is set, it can be cleared to 0 by resetting the receiver or by writing 1 to this bit position.

- Byte synchronous mode

Reserved. This bit always reads 0 and can be set to 0 or 1.

- Bit synchronous mode

The ABT bit indicates the abort end frame detection.

This bit is set by the character preceding the abort sequence when the receive frame ends with an abort. When this bit is set, the EOM bit is also set. See "Supplementary Explanation" below.

<b>ABT</b>	<b>Function</b>
0	Indicates that an abort end frame has not been detected.
1	Indicates that the receive frame has ended with an abort sequence.

#### Bit 4: FRME/RBIT (Framing error/residue bit frame)

This bit indicates the framing error status in the asynchronous mode, and the presence or absence of residue bits in the bit synchronous mode. This bit is cleared by writing 1 to this bit position.

- Asynchronous mode

FRME	Function
0	Indicates no framing error.
1	Indicates a framing error.

Once this bit is set, it can be cleared to 0 by resetting the receiver or by writing 1 to this bit position.

- Byte synchronous mode

Reserved. This bit always reads 0 and can be set to 0 or 1.

- Bit synchronous mode

The RBIT bit indicates residue bit frame detection.

When the CRCCC bit of the MMD0 register is set to 1, this bit is set to 1 by the residue bit of the last character in the receive frame I field. When the CRCCC bit of the MMD0 is 0, the RBIT bit is set to 1 by the residue bit of the last character of FCS.

When this bit is set, the EOM bit is also set. See "Supplementary Explanation" below.

RBIT	Function
0	Indicates a residue bit frame has not been detected.
1	Indicates a residue bit frame has been detected.

#### Bit 3: OVRN (Overrun error)

This bit indicates the overrun generated. This bit is cleared by writing 1 to this bit position. In the asynchronous and byte synchronous modes this bit is not cleared until 1 is written to the bit position or the receiver is reset. In the bit synchronous mode, all of the bits of this register are reset when the status data is loaded to the MSCI frame status register (MFST).

- Asynchronous/Byte/Bit synchronous modes

<b>OVRN</b>	<b>Function</b>
0	Indicates that no overrun has occurred.
1	Indicates that an overrun has occurred.

**Bit 2: CRCE (CRC error)**

This bit indicates a CRC error in the byte/bit synchronous mode.

- Asynchronous mode

Reserved. This bit always reads 0 and can be set to 0 or 1.

- Byte/Bit synchronous mode

This bit indicates a CRC error. When the CRCCC bit of MMD0 is 1, this bit is set when a CRC error occurs.

When the CRCCC bit is 0, this bit is not set.

This bit is cleared to 0 when 1 is written to this bit position or when the CRC calculation result is normal. The CRCE bit is the only bit in MST2 which changes as a result of changes in the status FIFO. For the timing of this bit, see "Error Checking CRC errors" in sections 4.3.2 "Byte Synchronous Mode" and 4.3.3 "Bit Synchronous Mode."

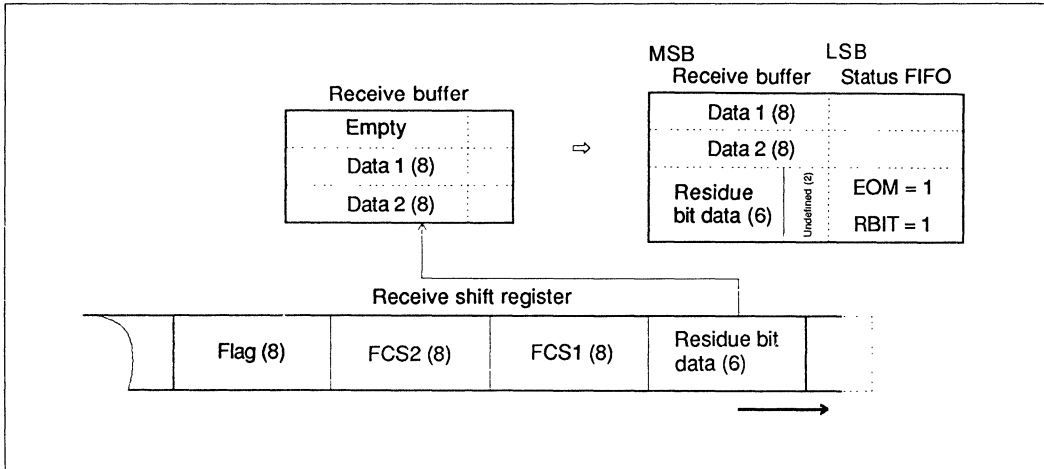
<b>CRCE</b>	<b>Function</b>
0	Indicates that a CRC error has not occurred.
1	Indicates that a CRC error has occurred.

**Bits 1-0:** Reserved. These bits always read 0 and can be set to 0 or 1.

## Supplementary explanation

- Operation when receiving a residue bit frame

Figure 4-5 shows how a residue bit frame is received. The residue bit frame data is transferred from the receive shift register to the receive buffer, and the residue bit frame status is set in the status FIFO.

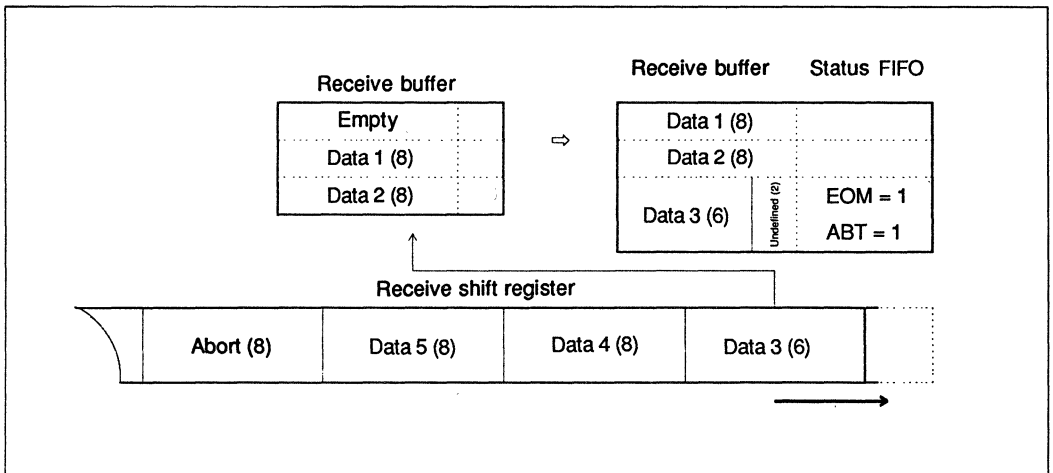


**Figure 4-5. Residue Bit Frame Reception Operation**

- ① Residue bit data is transferred from the receive shift register to the receive buffer. At this time, all other data is undefined.
- ② The EOM and RBIT bits of the status FIFO are set to 1.
- ③ If enabled, an interrupt is generated when the residue bit data becomes available to be read. However, because receive status is normally read from the MSCI frame status register (MFST) in the bit synchronous mode, the residue bit interrupt should be disabled.

- Abort end frame reception operation

Figure 4-6 shows the operation when receiving an abort end frame. Abort end frame data is transferred from the receive shift register to the receive buffer and the abort end frame status is set in the status FIFO.



**Figure 4-6. Abort End Frame Reception Operation**

- ① Part of the aborted data (data 3 in figure 4-6) is transferred from the receive shift register to the receive buffer. The bits other than this data are undefined.
- ② The EOM and ABT bits of the status FIFO are set to 1.
- ③ An interrupt is generated (if enabled) when the last data in the frame becomes available to be read. However, because receive status is normally read from the MSCFI frame status register (MFST) in the bit synchronous mode, the abort end frame interrupt should be disabled.

- Checking receive error

The receive error status is set in MST2. The occurrence of receive errors can be determined by reading this register as follows:

IN0 A, (23H) AND MSK	}	If the Z flag is 1 after executing the AND instruction, no error has occurred. If the Z flag is 0, it indicates an error.
-------------------------	---	---

Where 23H is the low-order byte of the MST2 I/O address and the mask pattern (MSK) is 11111100.

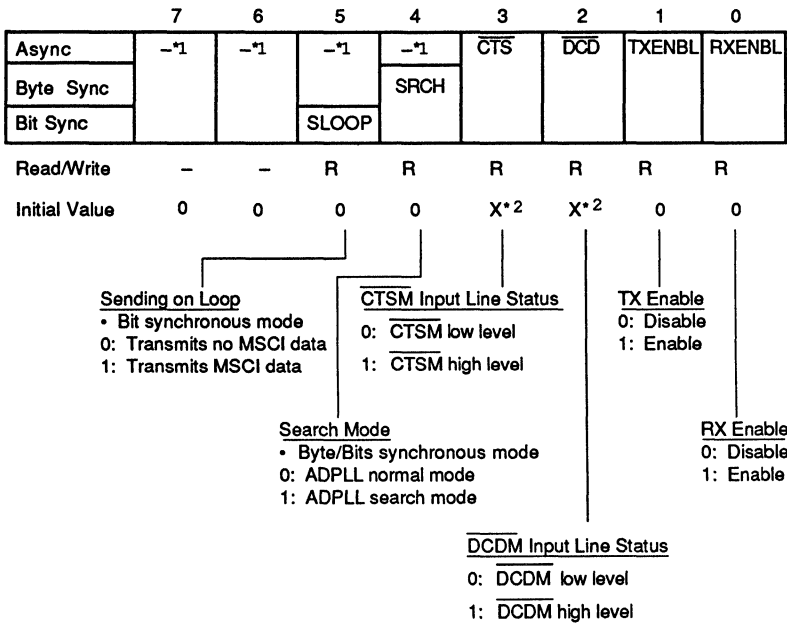
#### 4.2.12 MSCI Status Register 3 (MST3)

This register indicates the MSCI data transmit status in the bit synchronous mode, whether or not the ADPLL is in the search mode in the byte and bit synchronous modes, the  $\overline{\text{CTS}}$  and  $\overline{\text{DCDM}}$  line levels, and MSCI transmitter/receiver status (enable or disable). The register is a read-only register.

The bits of the MST3 are reset under the following conditions:

- A TX reset command resets bits 5, 3, and 1
- An RX reset command resets bits 2 and 0
- Hardware or a channel reset command resets all bits of the register
- All bits of the register are reset in system stop mode

The bits in this register do not generate interrupts.



\*1 Reserved. These bits always read 0.

\*2 X: Undefined



**Bits 7-6:** Reserved. These bits always read 0.

**Bit 5: SLOOP (Sending on loop)**

This bit indicates MSCI data transmission in the bit synchronous mode. It is cleared to 0 when the MSCI is not transmitting data. The bit is read-only, and writing to it has no effect.

- Asynchronous/Byte synchronous mode

Reserved. This bit always reads 0.

- Bit synchronous mode

SLOOP	Function
0	The MSCI is not transmitting data.
1	The MSCI is transmitting data.

**Bit 4: SRCH (Search mode)**

This bit indicates whether or not the ADPLL is in the search mode. This bit is valid for transmission using FM coding in the byte or bit synchronous mode. This is a read-only bit and writing to it has no effect. It is set by an enter search mode command and cleared to 0 when the ADPLL detects a level change in the receive data.

- Asynchronous mode

Reserved. This bit always reads 0.

- Byte/Bit synchronous mode

SRCH	Function
0	The ADPLL is not in the search mode.
1	The ADPLL is in the search mode.

**Bit 3:  $\overline{CTS}$  ( $\overline{CTS}$  input line status)**

This bit indicates the  $\overline{CTS}$  line level. This is a read-only bit, and writing to it has no effect.

- Asynchronous/Byte synchronous/Bit synchronous mode

$\overline{\text{CTS}}$	Function
0	The $\overline{\text{CTS}}$ input line is low.
1	The $\overline{\text{CTS}}$ input line is high.

#### Bit 2: $\overline{\text{DCD}}$ ( $\overline{\text{DCDM}}$ input line status)

This bit indicates the  $\overline{\text{DCDM}}$  line level. This bit is read only, and writing to it has no effect.

- Asynchronous/Byte synchronous/Bit synchronous mode

$\overline{\text{DCD}}$	Function
0	The $\overline{\text{DCDM}}$ input line is low.
1	The $\overline{\text{DCDM}}$ input line is high.

#### Bit 1: TXENBL (TX enable)

This bit indicates whether or not the transmitter is in the enable or disable state. Transmit enable/disable selection is performed by command. This bit is read only, and writing to it has no effect.

- Asynchronous/Byte synchronous/Bit synchronous mode

TXENBL	Function
0	The transmitter is disabled.
1	The transmitter is enabled.

#### Bit 0: RXENBL (RX enable)

This bit indicates whether or not the MSCI receiver is in the enable or disable state. Receive enable/disable selection is performed by command. This bit is read only, and writing to it has no effect.

RXENBL	Function
0	The receiver is disabled.
1	The receiver is enabled.

### 4.2.13 MSCI Frame Status Register (MFST)


This register stores the status of the last frame received in the bit synchronous mode.

The bits in the MSCI frame status register are reset under the following conditions:

- When a 1 is written to a particular bit position, that bit is reset.
- An RX or channel reset command resets all bits of the register.
- All bits of the register are reset when the system stop mode is entered.

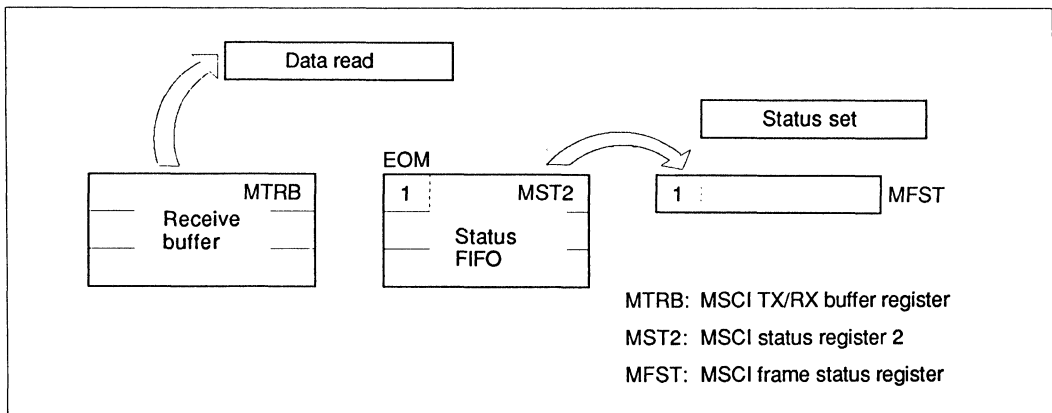
When the EOMF bit is set, a CPU interrupt is generated (if enabled). The other bits do not generate interrupts.

	7	6	5	4	3	2	1	0
Async	-*	-*	-*	-*	-*	-*	-*	-*
Byte Sync								
Bit Sync	EOMF	SHRTF	ABTF	RBITF	OVRNF	CRCEF		
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	-	-
Initial Value	0	0	0	0	0	0	0	0


  
 Frame status at receive completion

\* Reserved. These bits always read 0 and can be set to 0 or 1.

When data with EOM bit = 1 (last character of the frame) is read from the receive buffer, bits 7-2 of the MST2 associated with the character are set. Then, all of the bits in MST2 are cleared. See figure 4-7.



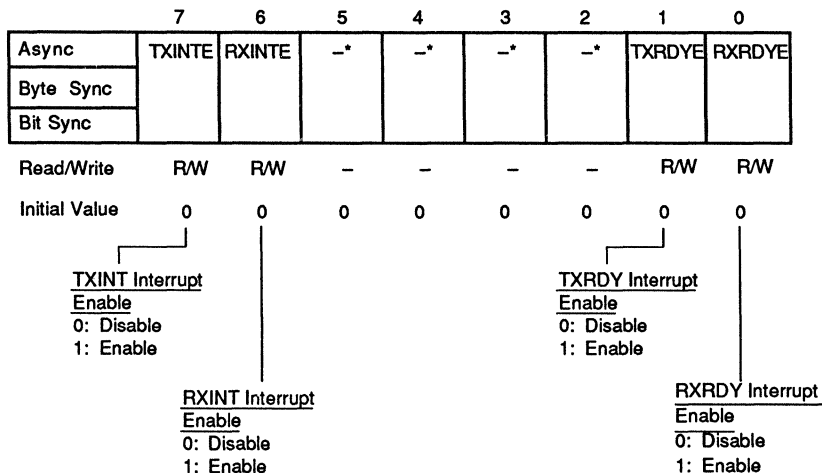
**Figure 4-7. MSCI Frame Status Register**

A frame end interrupt will occur when status data is read into the MFST. After this interrupt has occurred, the status of the received frame can be read from the MFST register.

This method is used for CPU data transfer. In this case, residue bit frame interrupts, abort end frame interrupts, and CRC error interrupts must be disabled.

#### 4.2.14 MSCI Interrupt Enable Register 0 (MIE0)

This register enables or disables interrupts TXINT, RXINT, TXRDY and RXRDY. Interrupt requests are issued to the CPU when both the MST0 register bits and the corresponding bits in this register are set. For details on interrupts, see section 4.7 "Internal Interrupts."



\* Reserved. These bits always read 0 and should be set to 0.

##### Bit 7: TXINTE (TXINT interrupt enable)

- Asynchronous/Byte synchronous/Bit synchronous mode

TXINTE	Function
0	Disables interrupts set by the TXINT bit in MST0.
1	Enables interrupts set by the TXINT bit in MST0; a TXINT interrupt request is issued to the CPU when the TXINT bit in MST0 is set to 1.

##### Bit 6: RXINTE (RXINT interrupt enable)

- Asynchronous/Byte synchronous/Bit synchronous mode

RXINTE	Function
0	Disables interrupts set by the RXINT bit in MST0.
1	Enables interrupts set by the RXINT bit in MST0; an RXINT interrupt request is issued to the CPU when the RXINT bit in MST0 is set to 1.

**Bits 5-2:** Reserved. These bits always read 0 and should be set to 0.

**Bit 1: TXRDYE (TXRDY interrupt enable)**

- Asynchronous/Byte synchronous/Bit synchronous mode

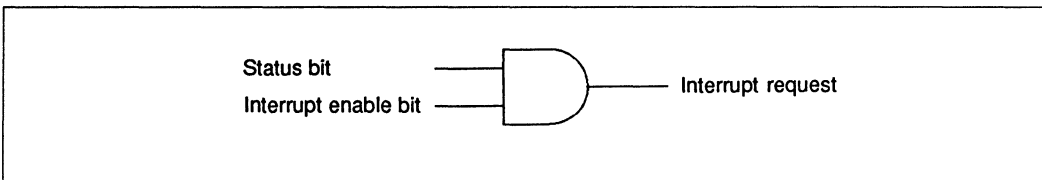
TXRDYE	Function
0	Disables interrupts set by the TXRDY bit in MST0.
1	Enables interrupts set by the TXRDY bit in MST0; a TXRDY interrupt request is issued to the CPU when the TXRDY bit in MST0 is set to 1.

**Bit 0: RXRDYE (RXRDY interrupt enable)**

- Asynchronous/Byte synchronous/Bit synchronous mode

RXRDYE	Function
0	Disables interrupts set by the RXRDY bit in MST0.
1	Enables interrupts set by the RXRDY bit in MST0; an RXRDY interrupt request is issued to CPU when the RXRDY bit in MST0 is set to 1.

Figure 4-8 shows the relationship between the interrupt enable bit and status bit.



**Figure 4-8. Interrupt Conditions**

An interrupt request is issued only when both the status bit and the interrupt enable bit are 1. This same rule is applicable to MSCI interrupt enable registers 0-2 (MIE0-2), the MSCI frame interrupt enable register (MFIE), MSCI status registers 0-2 (MST0-2), and the MSCI frame status register (MFST).

#### 4.2.15 MSCI Interrupt Enable Register 1 (MIE1)

This register specifies enable/disable of interrupts when the status bits in MSCI status register 1 (MST1) are set. For details on interrupts, see section 4.7 "Internal Interrupts."

	7	6	5	4	3	2	1	0
Async	-*	IDLE	-*	-*	CCTSE	CDCDE	BRKDE	BRKEE
Byte Sync	UDRNE			SYNCDE			-*	-*
HDLC				FLGDE			ABTDE/ GAPDE	IDLDE
Loop								
Read/Write	R/W	R/W	-	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

UDRN Interrupt Enable

- Byte/Bit synchronous mode

0: Disable  
1: Enable

IDL Interrupt Enable

0: Disable  
1: Enable

SYNCD Interrupt Enable

- Byte synchronous mode

0: Disable  
1: Enable

CCTS Interrupt Enable

0: Disable  
1: Enable

FLGD Interrupt Enable

- Bit synchronous mode

0: Disable  
1: Enable

CDCD Interrupt Enable

0: Disable  
1: Enable

BRKD Interrupt Enable

- Asynchronous mode

0: Disable  
1: Enable

ABTD/GAPD Interrupt Enable

- Bit synchronous mode

0: Disable  
1: Enable

BRKE Interrupt Enable

- Asynchronous mode

0: Disable  
1: Enable

IDLDE Interrupt Enable

- Bit synchronous mode

0: Disable  
1: Enable

\* Reserved. These bits always read 0 and should be set to 0.

#### Bit 7: UDRNE (UDRN interrupt enable)

- Asynchronous mode

Reserved. This bit always reads 0 and should be set to 0.

- Byte/Bit synchronous mode

<b>UDRNE</b>	<b>Function</b>
0	Disables interrupts set by the UDRN bit in MST1.
1	Enables interrupts set by the UDRN bit in MST1. (The TXINT bit in MST0 is set when the UDRN and UDRNE bits are both 1.)

**Bit 6: IDLE (IDL interrupt enable)**

- Asynchronous/Byte synchronous/Bit synchronous mode

<b>IDLE</b>	<b>Function</b>
0	Disables interrupts set by the IDL bit in MST1.
1	Enables interrupts set by the IDL bit in MST1. (The TXINT bit in MST0 is set when the IDL and IDLE bits are both 1.)

**Bit 5:** Reserved. This bit always reads 0 and should be set to 0.

**Bit 4: SYNCDE/FLGDE (SYNCD/FLGD interrupt enable)**

- Asynchronous mode

Reserved. This bit always reads 0 and should be set to 0.

- Byte/Bit synchronous mode

<b>SYNCDE /FLGDE</b>	<b>Function</b>
0	Disables interrupts set by the SYNCD/FLGD bit in MST1.
1	Enables interrupts set by the SYNCD/FLGD bit in MST1. (The RXINT bit in MST0 is set when the SYNCD/FLGD and SYNCDE/FLGDE bits are both 1.)



**Bit 3: CCTSE (CCTS interrupt enable)**

- Asynchronous/Byte synchronous/Bit synchronous mode

<b>CCTSE</b>	<b>Function</b>
0	Disables interrupts set by the CCTS bit in MST1.
1	Enables interrupts set by the CCTS bit in MST1. (The TXINT bit in MST0 is set when the CCTS and CCTSE bits are both 1.)

**Bit 2: CDCDE (CDCD interrupt enable)**

- Asynchronous/Byte synchronous/Bit synchronous mode

<b>CDCDE</b>	<b>Function</b>
0	Disables interrupts set by the CDCD bit in MST1.
1	Enables interrupts set by the CDCD bit in MST1. (The RXINT bit in MST0 is set when the CDCD and CDCDE bits are both 1.)

**Bit 1: BRKDE/ABTDE/GAPDE (BRKD/ABTD/GAPD interrupt enable)**

- Asynchronous/Bit synchronous mode

<b>BRKDE/ ABTDE/ GAPDE</b>	<b>Function</b>
0	Disables interrupts set by the BRKD/ABTD/GAPD bit in MST1.
1	Enables interrupts set by the BRKD/ABTD/GAPD bit in MST1. (The RXINT bit in MST0 is set when the BRKD/ABTD/GAPD and BRKDE/ABTDE/GAPDE bits are both 1.)

- Byte synchronous mode

Reserved. This bit always reads 0 and should be set to 0.

**Bit 0: BRKEE/IDLDE (BRKE/IDLD interrupt enable)**

- Asynchronous/Bit synchronous mode

**BRKEE  
/IDLDE**

	<b>Function</b>
0	Disables interrupts set by the BRKE/IDLD bit in MST1.
1	Enables interrupts set by the BRKE/IDLD bit in MST1. (The RXINT bit in MST0 is set when the BRKE/IDLD and BRKEE/IDLDE bits are both 1.)

- Byte synchronous mode

Reserved. This bit always reads 0 and should be set to 0.

## 4.2.16 MSCI Interrupt Enable Register 2 (MIE2)

This register specifies enable/disable of interrupts when the status bits in MSCI status register 2 (MST2) are set. For details on interrupts, see section 4.7 "Internal Interrupts."

	7	6	5	4	3	2	1	0
Async	-*	PMPE	PEE	FRMEE	OVRNE	-*		
Byte Sync	-*	-*	-*	-*		CRCEE	-*	-*
Bit Sync	EOME	SHRTE	ABTE	RBITE				
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	-	-
Initial Value	0	0	0	0	0	0	0	0

**EOM Interrupt Enable**

- Bit synchronous mode
- 0: Disable
- 1: Enable

**PMP Interrupt Enable**

- Asynchronous mode
- 0: Disable
- 1: Enable

**SHRT Interrupt Enable**

- Bit synchronous mode
- 0: Disable
- 1: Enable

**PE Interrupt Enable**

- Asynchronous mode
- 0: Disable
- 1: Enable

**ABT Interrupt Enable**

- Bit synchronous mode
- 0: Disable
- 1: Enable

**CRCE Interrupt Enable**

- Byte/bit synchronous mode
- 0: Disable
- 1: Enable

**OVRN Interrupt Enable**

- 0: Disable
- 1: Enable

**FRME Interrupt Enable**

- Asynchronous mode
- 0: Disable
- 1: Enable

**RBIT Interrupt Enable**

- Bit synchronous mode
- 0: Disable
- 1: Enable

\* Reserved. These bits always read 0 and should be set to 0.

### Bit 7: EOME (EOM interrupt enable)

- Asynchronous/Byte synchronous mode

Reserved. This bit always reads 0 and should be set to 0.

- Bit synchronous mode

<b>EOME</b>	<b>Function</b>
0	Disables interrupts set by the EOM bit in MST2.
1	Enables interrupts set by the EOM bit in MST2. (The RXINT bit in MST0 is set when the EOM and EOME bits are both 1.)

**Bit 6: PMPE/SHRTE (PMP/SHRT interrupt enable)**

- Asynchronous/Bit synchronous mode

<b>PMPE/ SHRTE</b>	<b>Function</b>
0	Disables interrupts set by the PMP/SHRT bit in MST2.
1	Enables interrupts set by the PMP/SHRT bit in MST2. (The RXINT bit in MST0 is set when the PMP/SHRT and PMPE/SHRTE bits are both 1.)

- Byte synchronous mode

Reserved. This bit always reads 0 and should be set to 0.

**Bit 5: PEE/ABTE (PE/ABT interrupt enable)**

- Asynchronous/Bit synchronous mode

<b>PEE/ ABTE</b>	<b>Function</b>
0	Disables interrupts set by the PE/ABT bit in MST2.
1	Enables interrupts set by the PE/ABT bit in MST2. (The RXINT bit in MST0 is set when the PE/ABT and PEE/ABTE bits are both 1.)

- Byte synchronous mode

Reserved. This bit always reads 0 and should be set to 0.

#### Bit 4: FRMEE/RBITE (FRME/RBIT interrupt enable)

- Asynchronous/Bit synchronous mode

FRMEE/ RBITE	Function
0	Disables interrupts set by the FRME/RBIT bit in MST2.
1	Enables interrupts set by the FRME/RBIT bit in MST2. (The RXINT bit in MST0 is set when FRME/RBIT and FRMEE/RBITE bits are both 1.)

- Byte synchronous mode

Reserved. This bit always reads 0 and should be set to 0.

#### Bit 3: OVRNE (OVRN interrupt enable)

- Asynchronous/Byte synchronous/Bit synchronous mode

OVRNE	Function
0	Disables interrupts set by the OVRN bit in MST2.
1	Enables interrupts set by the OVRN bit in MST2. (The RXINT bit in MST0 is set when the OVRN and OVRNE bits are both 1.)

#### Bit 2: CRCEE (CRCE interrupt enable)

- Asynchronous mode

Reserved. This bit always reads 0 and should be set to 0.

- Byte/Bit synchronous mode

CRCEE	Function
0	Disables interrupts set by the CRCE bit in MST2.
1	Enables interrupts set by the CRCE bit in MST2. (The RXINT bit in MST0 is set when the CRCE and CRCEE bits are both 1.)

Bits 1 and 0: Reserved. These bits always read 0 and should be set to 0.

#### 4.2.17 MSCI Frame Interrupt Enable Register (MFIE)

This register specifies whether to enable or disable an interrupt when the EOMF bit in the MSCI frame status register (MFST) is set.

	7	6	5	4	3	2	1	0
Async	-*							
Byte Sync	-*	-*	-*	-*	-*	-*	-*	-*
Bit Sync	EOMFE							
Read/Write	RW	-	-	-	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0

**EOMF Interrupt Enable**

- Bit synchronous mode
- 0: Disable
- 1: Enable

\* Reserved. These bits always read 0 and should be set to 0.

#### Bit 7: EOMFE (EOMF interrupt enable)

- Asynchronous/Byte synchronous mode

Reserved. This bit always reads 0 and should be set to 0.

- Bit synchronous mode

EOMFE	Function
0	Disables interrupts set by the EOMF bit in MFST.
1	Enables interrupts set by the EOMF bit in MFST. (The RXINT bit in MST0 is set when the EOMF and EOMFE bits are both 1.)

Bits 6-0: Reserved. These bits always read 0 and should be set to 0.

#### 4.2.18 MSCI Synchronous/Address Register 0 (MSA0)

This register is used to specify the SYN character pattern for reception in the byte synchronous/mono-sync mode, the low order byte of the SYN character pattern for transmission and reception in the byte synchronous/bi-sync mode, and a secondary station address in bit synchronous mode. This register is not used in the asynchronous or external byte synchronous mode.

	7	6	5	4	3	2	1	0
Async *	-	-	-	-	-	-	-	-
Byte Sync	SA07	SA06	SA05	SA04	SA03	SA02	SA01	SA00
Bit Sync								
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	1

|  
SYN Pattern for Reception/Address Field Check

- Byte synchronous mode

Mono-sync	SYN pattern for reception
Bi-sync	SYN pattern for transmission and reception (bits 7-0)
External-sync	Unused

- Bit synchronous mode

HDLC mode	Address field not checked	Unused
	Single address 1	Bits 7-0 of the secondary station address
	Single address 2	Unused
	Dual address	Bits 7-0 of the secondary station address
Loop mode	Address field not checked	Unused
	Single address 1	Bits 7-0 of the secondary station address
	4-bit address	Bits 7-4 of the secondary station address
	Dual address	Bits 7-0 of the secondary station address

\* This register is not used in the asynchronous mode.

## Bits 7-0: SA07-0 (Synchronous/Address)

### Asynchronous mode

This register is not used in asynchronous mode.

- Byte synchronous (mono- or bi-sync) mode

Specifies bits 7-0 of the SYN character pattern for reception in the mono-sync mode, and the eight lower bits (bits 7-0) of the SYN character pattern for transmission and reception in the bi-sync mode.

- Bit synchronous mode

Sets the following values according to the address field check selection in the HDLC and loop modes.

Mode	Address Field Check	Bits 7-0 of MSA0
HDLC mode	Address field not checked	Unused
	Single address 1	Bits 7-0 of the secondary station address
	Single address 2	Unused
	Dual address	Bits 7-0 of the secondary station address
Loop mode	Address field not checked	Unused
	Single address 1	Bits 7-0 of the secondary station address
	4-bit address	Bits 7-4 of the secondary station address *
	Dual address	Bits 7-0 of the secondary station address

\* Bits 3-0 of MSCI synchronous/address register 0 are unused in the 4-bit address mode of the bit synchronous loop mode.



#### 4.2.19 MSCI Synchronous/Address Register 1 (MSA1)

This register is used to specify a SYN character pattern for transmission in the byte synchronous (mono-sync or external-sync) mode, a SYN character pattern for transmission and reception in the byte synchronous bi-sync mode, and a secondary station address in the bit synchronous mode. This register is not used in the asynchronous mode.

	7	6	5	4	3	2	1	0
Async *	-	-	-	-	-	-	-	-
Byte Sync	SA17	SA16	SA15	SA14	SA13	SA12	SA11	SA10
Bit Sync								
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	1	1	1	1	1	1	1	1

|  
SYN Pattern for Transmission/Address Field Check

- Byte synchronous mode

Mono-sync	SYN pattern for transmission
Bi-sync	SYN pattern for transmission and reception (bits 15–8)
External-sync	SYN pattern for transmission

- Bit synchronous mode

HDLC mode	Address field not checked	Unused
	Single address 1	Unused
	Single address 2	Bits 15–8 of the secondary station address
	Dual address	Bits 15–8 of the secondary station address
Loop mode	Address field not checked	Unused
	Single address 1	Unused
	4-bit address	Unused
	Dual address	Bits 15–8 of the secondary station address

\* This register is not used in the asynchronous mode.

#### Bits 7-0: SA17-0 (Synchronous/Address)

- Asynchronous mode

This register is not used in the asynchronous mode.

- Byte synchronous mode

Specifies bits 7-0 of the SYN character pattern for transmission in the byte synchronous (mono-sync or external-sync) mode, and the eight upper bits (bits 15-8) of the SYN character pattern in the byte synchronous (bi-sync) mode.

- Bit synchronous mode

Sets the following values according to the address field check selection in the HDLC and loop modes.

Mode	Address Field Check	Bits 7-0 of MSA1
HDLC mode	Address field not checked	Unused
	Single address 1	Unused
	Single address 2	Bits 15-8 of the secondary station address
	Dual address	Bits 15-8 of the secondary station address
Loop mode	Address field not checked	Unused
	Single address 1	Unused
	4-bit address	Unused
	Dual address	Bits 15-8 of the secondary station address

#### 4.2.20 MSCI Idle Pattern Register (MIDL)

This register is used to specify the idle pattern output by the transmitter when it is in the idle state.

	7	6	5	4	3	2	1	0
Async *	–	–	–	–	–	–	–	–
Byte Sync	IDL7	IDL6	IDL5	IDL4	IDL3	IDL2	IDL1	IDL0
Bit Sync								
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	1

\* This register is not used in the asynchronous mode.

Idle Pattern

#### Bits 7-0: IDL 7-0 (Idle pattern)

- Asynchronous mode

This register is not used in the asynchronous mode.

- Byte/Bit synchronous mode

If the IDLC bit of MCTL is 1, the idle pattern set in this register is output from the TXDM line during the idle state. When the IDLC bit is 0, the TXDM line is fixed at high level.

#### 4.2.21 MSCI TX/RX Buffer Register (MTRB)

This register is located at the top of the 3-stage transmit/receive buffer and is connected directly to the internal data bus. Although the transmit and receive MTRBs are physically different, this register can be accessed both to read receive data and to write transmit data.

	7	6	5	4	3	2	1	0
Async	TRB7	TRB6	TRB5	TRB4	TRB3	TRB2	TRB1	TRB0
Byte Sync								
Bit Sync								
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	X	X	X	X	X	X	X	X

Value written to, or read from, the transmit/receive buffer

X: Undefined

#### Bits 7-0: TRB7-0 (TX/RX buffer)

- Asynchronous/Byte synchronous/Bit synchronous mode

A receive character in the receive buffer can be read by reading bits 7-0 of this register. These bit values are undefined when the RXRDY bit in MST0 is cleared to 0.

A transmit character can be written to the transmit buffer by writing bits 7-0 of this register. If data is written to the TRB 7-0 bits while the TXRDY bit of the MST0 register is cleared to 0, the write data and/or the data in the transmit buffer may be lost.

## 4.3 Operation

### 4.3.1 Asynchronous Mode

In the asynchronous mode, each character is synchronized by appending a start bit and stop bit(s) to the character before transmission. The transmission line is normally at high level (mark); a start bit, indicating the start of transmission, causes the line to go low (space).

Figure 4-9 shows the character format for the asynchronous mode. In this mode data is transmitted and received in units of characters which may be from 5 to 8 bits in length. When the character length is from 5 to 7 bits, each received character is extended to 8 bits by padding the higher order bit positions with 0s.

Data transmission begins with a start bit. The start bit is followed by the data, beginning with the least significant bit (LSB), that is, bit 0. The data may be optionally followed by a parity/MP bit. The data transmission ends with 1, 1.5, or 2 stop bits.

The PRTCL2-0 bits of MSCI mode register 0 (MMD0) specify the asynchronous mode. The TXCHR1-0 and RXCHR1-0 bits of MMD1 specify the bit length of characters, the STOP1-0 bits of MMD0 specify the length of the stop bit, and the PMPM1-0 bits of MMD1 specify the parity/MP bit setting. In the asynchronous mode, the coding type is only NRZ type. (For details, see section 4.2.1 "MSCI Mode Register 0" and section 4.2.2 "MSCI Mode Register 1.")

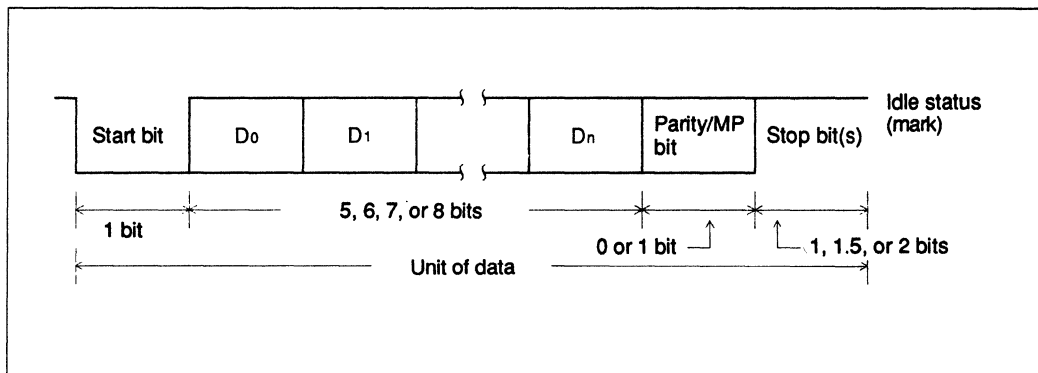
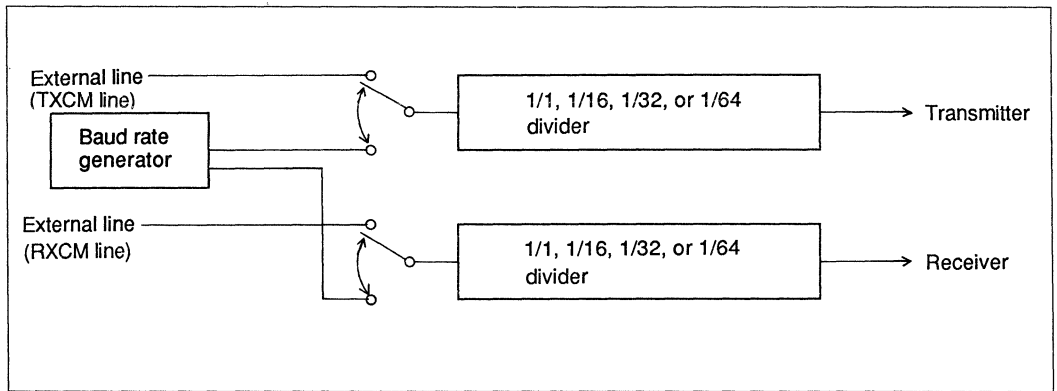
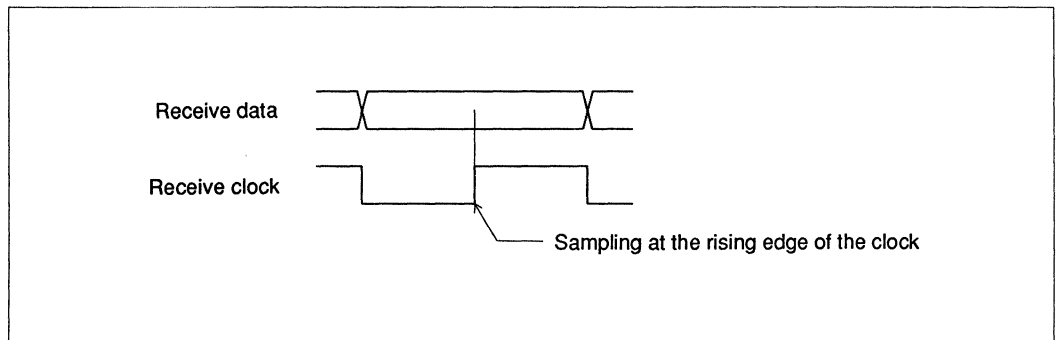


Figure 4-9. Character Format for Asynchronous Mode

The transmit and receive bit rates can be independently selected (see figure 4-10) from among input clock frequency ratios of: 1/1, 1/16, 1/32, or 1/64. Since data sampling occurs at the rising edge of the clock, bit synchronization is necessary when 1/1 is selected (see figure 4-11).



**Figure 4-10. Bit Rate Selection**



**Figure 4-11. Timing of Data Sampling (1/1 clock mode)**

The BRATE1-0 bits in MMD1 are used to specify the bit rate.

The external clock or internal baud rate generator output can be program-selected to serve as the input/output clock. The ADPLL clock extraction function is not available in the asynchronous mode. The MSCI RX clock source register (MRXS) and MSCI TX clock source register (MTXS) are used to specify the clock.

For details see section 4.2.5 "MSCI RX Clock Source Register" and section 4.2.6 "MSCI TX Clock Source Register."

See section 4.6 "Baud Rate Generator" for details about the on-chip baud rate generator.

**Transmission Operation:** Figure 4-12 shows the state transition diagram for asynchronous mode transmission.

- **Transmit disable state**

The transmitter is placed in the transmit disable state by a hardware or a channel reset, transmit reset, or a transmit disable command.

In this state, the TXDM line remains at mark and the TXRDY bit in MSCI status register 0 (MST0) is cleared to 0.

- **Idle state**

The transmitter enters the idle state from the transmit disable state after a transmit enable command. The idle state maintains the TXDM line at mark while waiting for transmit data to be written to the transmit buffer. Once the transmit data is written, the transmitter enters the start bit transmit state.

- **Start bit transmit state**

The TXDM line remains at space for one bit cycle and then enters the character transmit state.

- **Character transmit state**

The transmitter sends the character contained in the transmit buffer beginning with the LSB.

- **Parity/MP bit transmit state**

The transmitter sends a parity or MP bit as specified by the PMPM1-0 bits of MMD1. For details, see "Parity/MP Bit" below.

- **Stop bit transmit state**

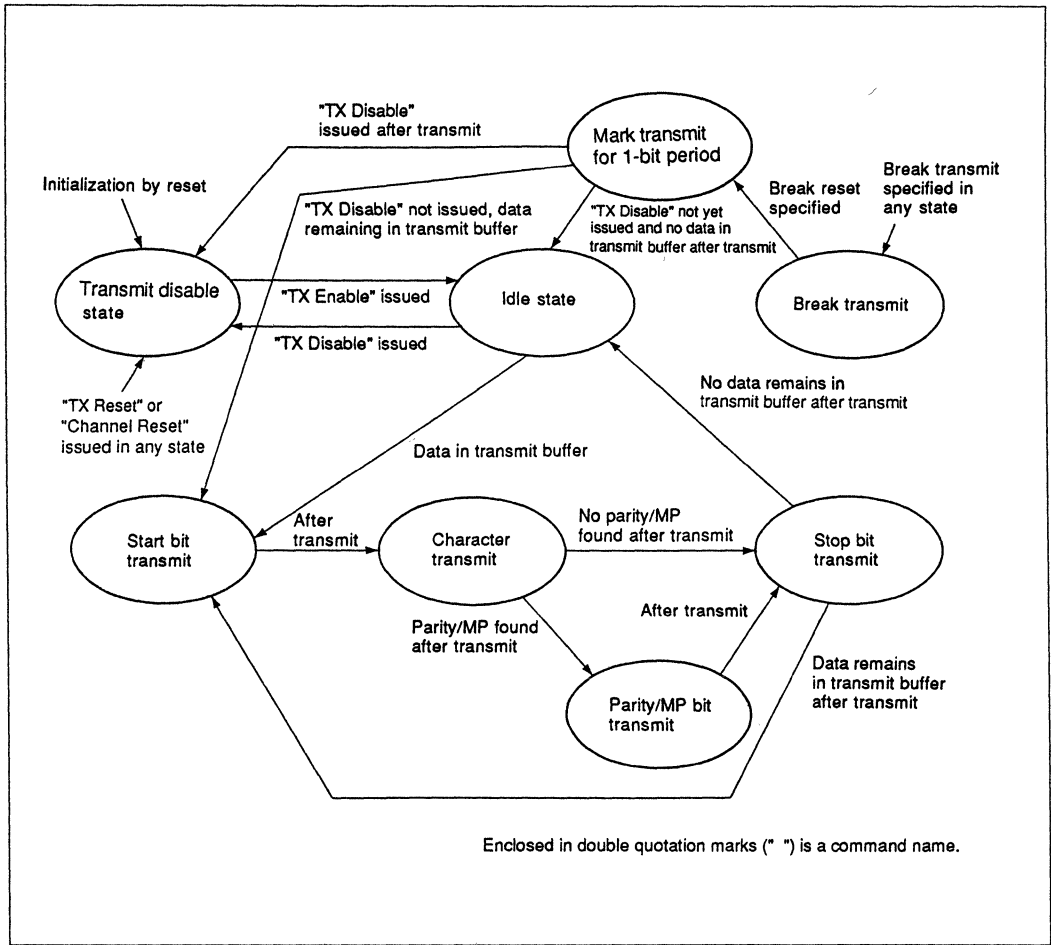
The transmitter sends a stop bit as specified by the STOP1-0 bits of MMD0, and then returns to the idle state.

- **Break transmit state**

The TXDM line remains at space. A break is transmitted when the BRK bit of the MSCI control register (MCTL) is set to 1. The break transmit state is maintained until it is reset (the BRK bit is cleared).

- **Mark transmit for one bit cycle state**

The TXDM line remains at mark for one bit cycle after the break transmit state is reset.



**Figure 4-12. State Transition Diagram for Asynchronous Mode Transmission**

Transmission starts when a transmit character is written into the transmit buffer in the idle state. The transmit line output changes at the falling edge of the transmit clock. In figures 4-13 (a) and (b) the character length is 8 bits, parity is used, and the stop bit length is 1 bit.

A stop bit length of 1, 1.5, or 2 can be specified in the 1/16, 1/32, or 1/64 clock mode. In the 1/1 clock mode, only a stop bit length 1 or 2 is available. If 1.5 is specified in this case, stop bit length is specified 1 or 2.

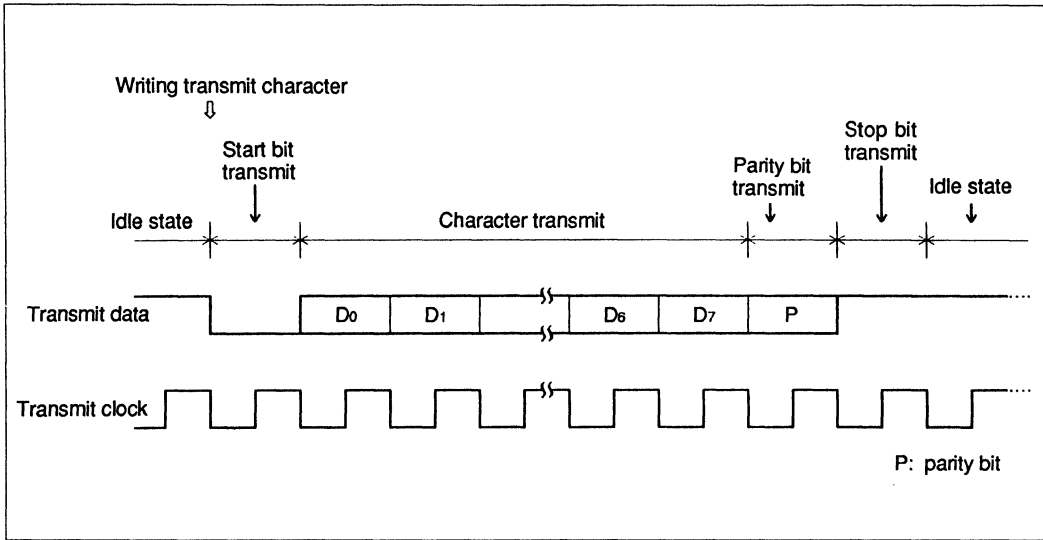


Figure 4-13. (a) Transmit Operation in 1/1 Clock Mode

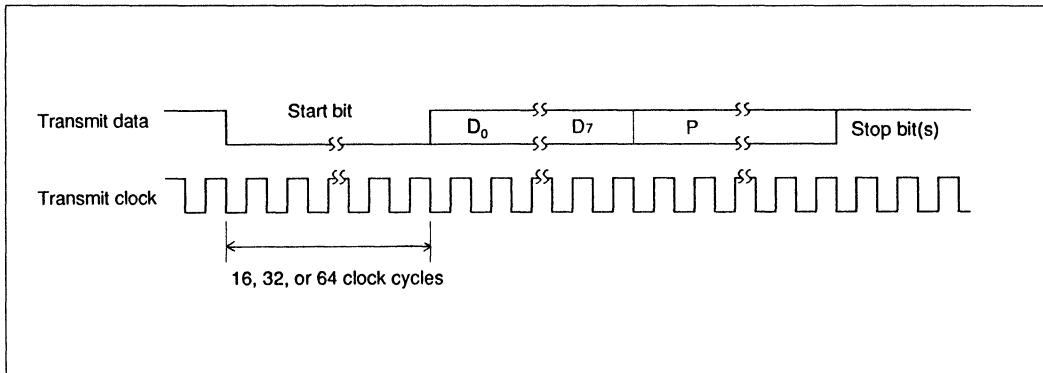


Figure 4-13. (b) Transmit Operation in 1/16, 1/32, or 1/64 Clock Mode

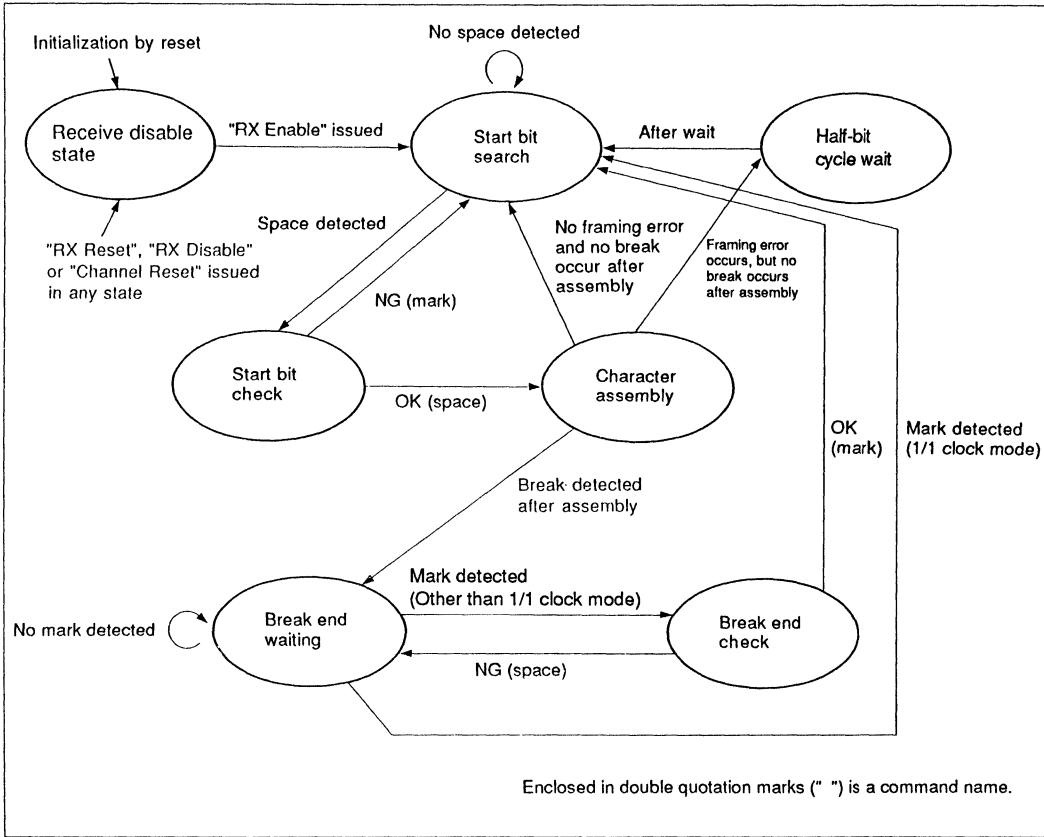
**Receive Operation:** Figure 4-14 shows the state transition diagram for asynchronous mode reception.

- Receive disable state

The receiver is placed in the receive disable state by a hardware or channel reset, receive reset, or receive disable command. In this state, input from the RXDM line is ignored and no reception operation occurs. The contents of the receive shift register are lost, but the value of the receive buffer is not changed.



- **Start bit search state**  
The receiver enters the start bit search state from the receive disable state after an receive enable command. In this state the receiver samples the RXDM line at the rising edge of each receive clock cycle until a space is detected.
- **Start bit check state**  
When the receiver detects a space in the start bit search state, it enters the start bit check state. After detecting a space, the receiver waits for half a bit cycle and samples the RXDM line again to verify that it remains at space. If the line does not remain at space, the receiver returns to the start bit search state. If the line remains at space, the receiver enters the character assembly state. In the 1/1 clock mode, this state is skipped; the receiver enters the character assembly state directly.
- **Character assembly state**  
The receiver samples the received data each bit cycle and assembles a character. Character assembling ends when the first stop bit is detected.
- **Half-bit cycle wait state**  
If a framing error occurs after character assembling has completed, the receiver waits for half a bit cycle in order to skip the stop bit associated with the framing error. It then enters the start bit search state.  
For details about framing errors, see "Error Checking."
- **Break end wait state**  
If a break is detected after character assembling, the receiver enters the break end wait state. The RXDM line is sampled each clock cycle until it goes to mark.  
For details about a break, see "Break Transmission and Detection."
- **Break end check state**  
When a mark is detected in the break end wait state, the receiver enters the break end check state. After detecting a mark, the receiver delays for half a bit cycle and again samples the RXDM line to verify that it remains at mark. If the line does not remain at mark, the receiver returns to the break end wait state. If the line remains at mark, the receiver enters the start bit search state. In the 1/1 clock mode, the break end check state is skipped and the receiver enters the start bit search state directly.



**Figure 4-14. State Transition Diagram for Asynchronous Mode Reception**

Figures 4-15 (a) and (b) show timing of sampling diagrams for receive data.

In these examples, the character length is 8 bits, parity is used, and the stop bit length is 1 bit.

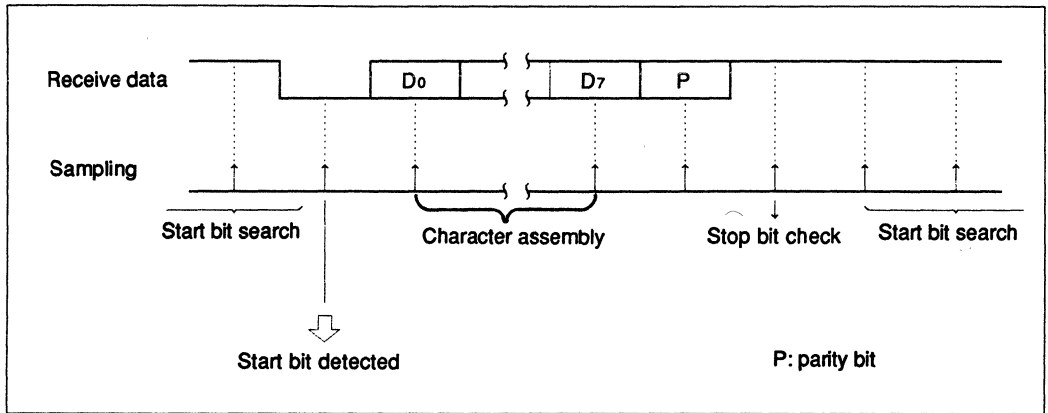


Figure 4-15. (a) Receive Data Sampling Timing (1/1 Clock Mode)

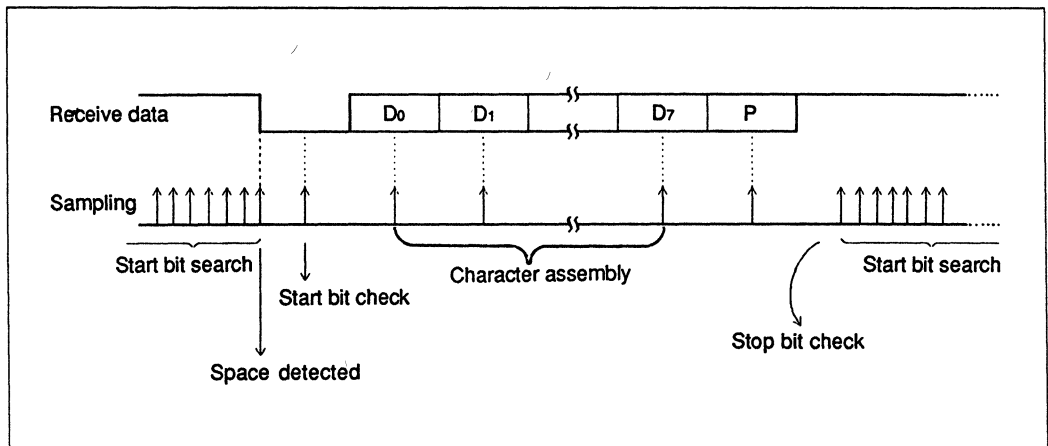


Figure 4-15. (b) Receive Data Sampling Timing (1/16, 1/32, or 1/64 Clock Mode)

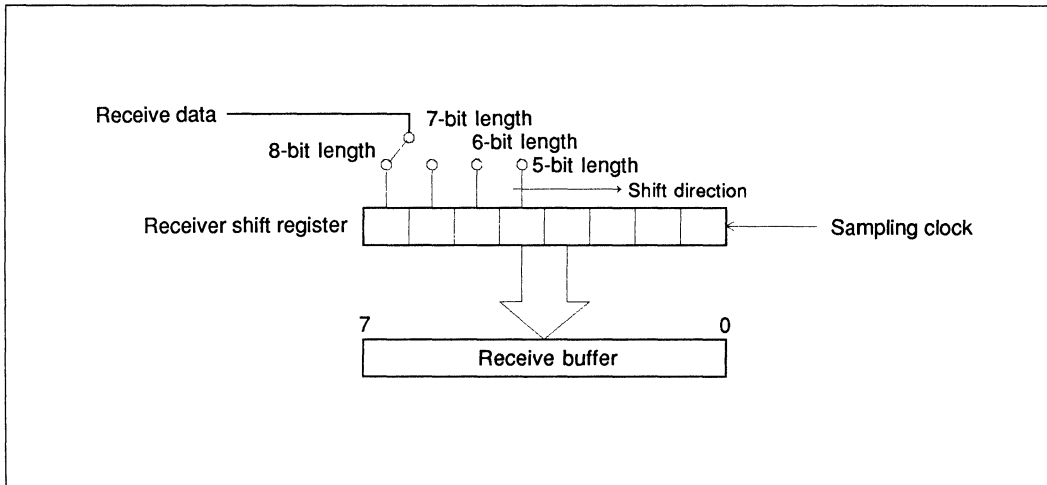
The receive operation starts when a receive enable command is issued.

In the 1/1 clock mode, the receiver searches for a start bit at the rising edge of each clock. If a space is detected, character assembling begins at the next clock rising edge.

Character assembling involves assembling a character by loading the bit sampled in each clock cycle into a receiver shift register, as shown in figure 4-16.

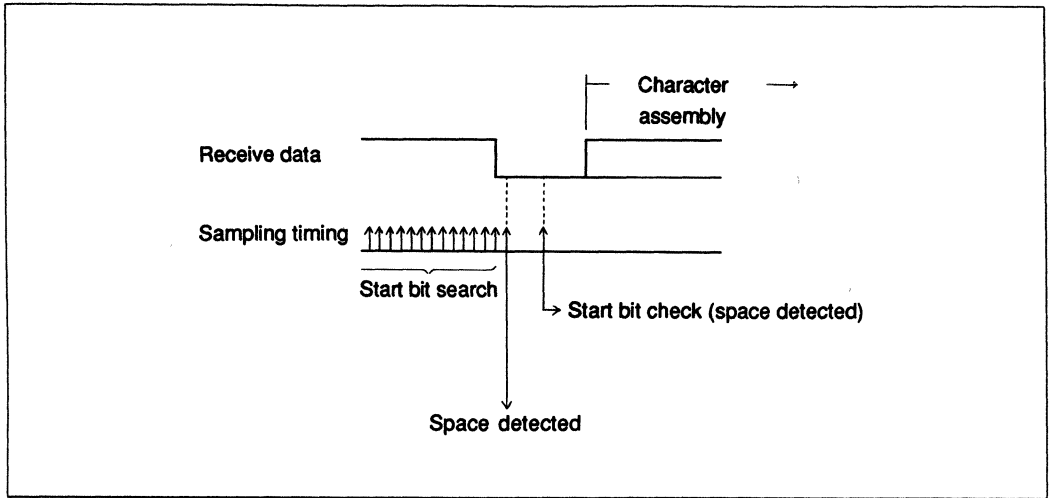
Data of the received character length, specified by the RXCHR1-0 bits of MMD1, is transferred to the receiver shift register, and then the parity/MP bit is sampled (if it exists). In the next clock cycle, the stop bit is sampled to complete the character assembly process. At this time, the receiver shift register value is loaded into the receive buffer.

One clock cycle after the completion of the character assembly process, the receiver resumes scanning for a start bit.

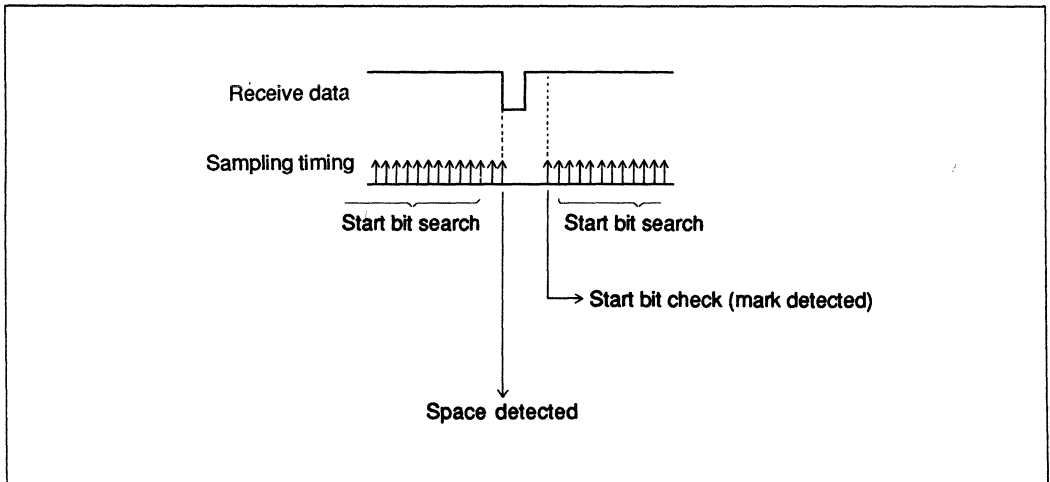


**Figure 4-16. Character Assembly in the Receiver Shift Register**

In the 1/16, 1/32, or 1/64 clock mode, start bit scanning involves sampling the input line at each clock rising edge. If a space is detected, the line level is checked again after a delay of half a bit cycle. If the line is still at space, character assembly will start after a delay of one bit cycle. If the transmission line is at mark, start bit scanning resumes because the previously detected space is assumed to have resulted from noise (see figures 4-17 (a) and (b)).



**Figure 4-17. (a) Start Bit Sampling (normal start bit is detected)**

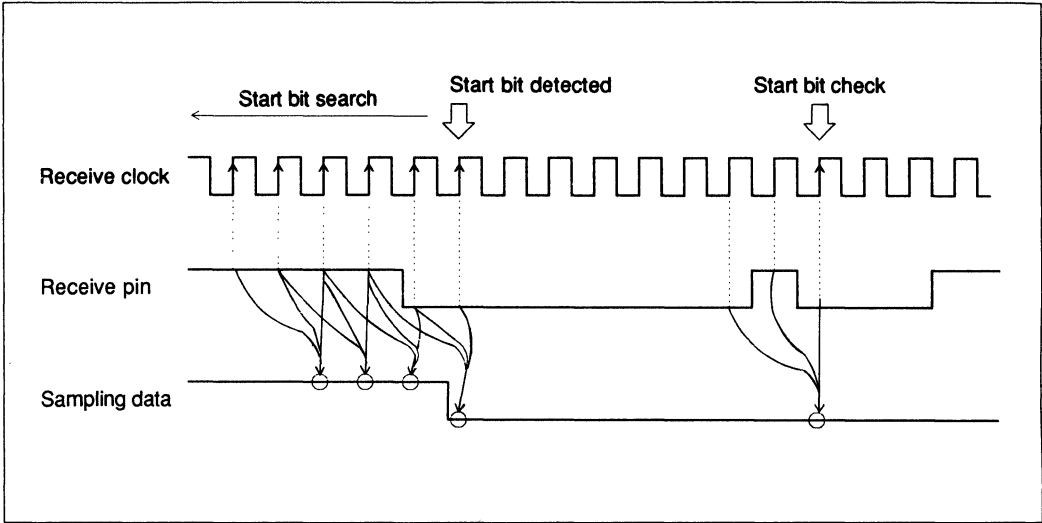


**Figure 4-17. (b) Start Bit Sampling (false start bit (noise) is detected)**

In the character assembly process, data is sampled every other bit cycle. When the most significant bit (MSB) or the parity bit (if present) has been detected, the stop bit is checked after a delay of one bit cycle. If at this time the RXDM line is at mark (normal), start bit scanning resumes immediately. If the line is at space (framing error), start bit scanning resumes after a delay of half a bit cycle.

In the 1/16, 1/32, or 1/64 clock mode, the noise suppressor function operates for the sampling of the start, parity, and stop bits, and character.

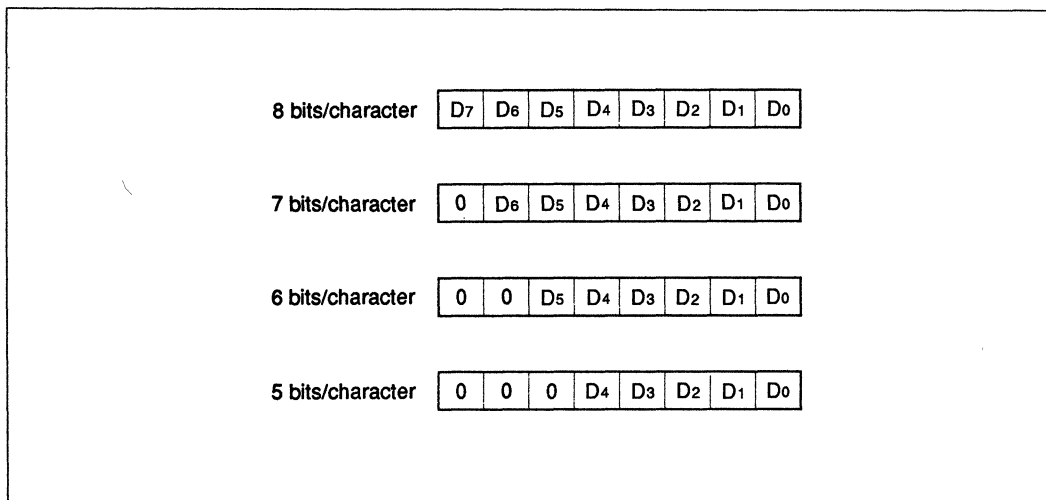
The noise suppressor function operates by applying majority rule to the current value and the two previously clocked values (see figure 4-18).



**Figure 4-18. Noise Suppressor Function**

In the asynchronous mode, the receivable character length is from 8 to 5 bits. The RXCHR1-0 bits in MMD1 are used to specify the character length.

Figure 4-19 shows the receive data format. When the character length is from 7 to 5 bits, the higher order bits are padded with 0s.



**Figure 4-19. Receive Character Format**

**Parity/MP Bit:** The PMPM1-0 bits of the MMD1 are used to specify whether or not an even/odd parity bit or an MP bit are to be appended.

When even parity is selected, the transmitter counts the number of 1s in the transmit character and appends a 0 if the number is even or a 1 if the number is odd. Thus, the total number of 1s actually transmitted is even. The receiver checks whether or not the number of 1s in the received character and parity bits is even.

Similarly, if an odd parity is selected, the value of the parity bit is set so that the total number of 1s transmitted is odd.

When the MP bit is selected, an MP bit is appended to the transmitted and received characters in order to support multiprocessor communications.

For details, see "Multiprocessor Support."

### **Error Checking**

- **Parity check**

The receiver verifies that received data has the proper parity bit.

If even parity is specified and an odd number of 1s are detected in the received characters and parity bits, the PE (parity error) bit in MSCI status register 2 (MST2) is set to 1 when the receive data

containing the parity error becomes ready to be read. The situation for odd parity is the same except that an even number of 1s triggers the error.

For details about the PE bit, see section 4.2.11 "MSCI Status Register 2."

Even if a parity error has occurred, subsequent data are normally received; however, the PE bit cannot be cleared until 1 is written to the PE bit by the CPU or the NPU is reset.

When the PE bit is set, an internal interrupt is generated (if enabled).

- Framing error

A space detected where a stop bit should be causes a framing error. Even if the stop bit length is 1.5 or 2 bits, only the first bit is checked.

When data containing a framing error becomes ready to be read, the FRME bit of MST2 is set.

For details about the FRME bit, see section 4.2.11 "MSCI Status Register 2."

A framing error does not stop the receive operation. In the 1/1 clock mode, start bit scanning resumes in the clock cycle following detection of the framing error. In the 1/16, 1/32, or 1/64 clock mode, scanning resumes after a delay of a half-bit cycle; this period allows invalid stop bit(s) to be skipped.

Once the FRME bit is set by a framing error, it is not cleared until 1 is written to the FRME bit by the CPU or the NPU is reset.

When the FRME bit is set, an internal interrupt is generated (if enabled).

- Overrun error

If the buffer is full when new data is transferred to the receive buffer, an overrun error occurs.

When an overrun error occurs, the new data is written into the top of the receive buffer (TRB), erasing the previous data. At the same time, the last stage of the receive status FIFO is overwritten with the status (including an overrun indication) of the new data. The OVRN bit in MST2 is set to 1 when the overwritten data becomes ready to be read.

For details about the OVRN bit, see section 4.2.11 "MSCI Status Register 2."



If an overrun error occurs, subsequent data will be received normally. However, the OVRN bit, once set, is not cleared even if the subsequent data causes no overrun error.

The OVRN bit can be cleared only when 1 is written to the OVRN bit by the CPU or the MSCI receiver is reset. When the OVRN bit is set, an internal interrupt is generated (if enabled).

**Break Transmission and Detection:** When the transmitter must suspend data transmission it transmits a break (space).

Normally, it issues a break transmission request after completing the current character transmission. The transmitter must continue to send the break signal for one or more character cycles. Break transmission is specified by BRK bit of the MCTL. This bit is set and TXDM line goes to space at the falling edge of the next transmit clock.

To cancel break transmission, clear the BRK bit. When the BRK bit is cleared, the TXDM line goes to mark at the falling edge of the next transmit clock. At this time, the receiver verifies that the mark level continues for one or more bit cycles before resuming start bit scanning.

When break transmission is requested, the output data in the transmit shift register is lost, but the transmit buffer is not affected.

The receiver detects a break as follows:

If the data and parity bits are all 0s and the data contains a framing error, it is assumed that this is the start of a break and the BRKD bit in MST1 is set. When the start of a break is detected, the null character containing the framing error is discarded (not transferred to the receive buffer).

Figure 4-20 shows break detection by the receiver. Therefore, if break transmission starts while transmitting a character, the break transmission must continue for two or more character cycles.

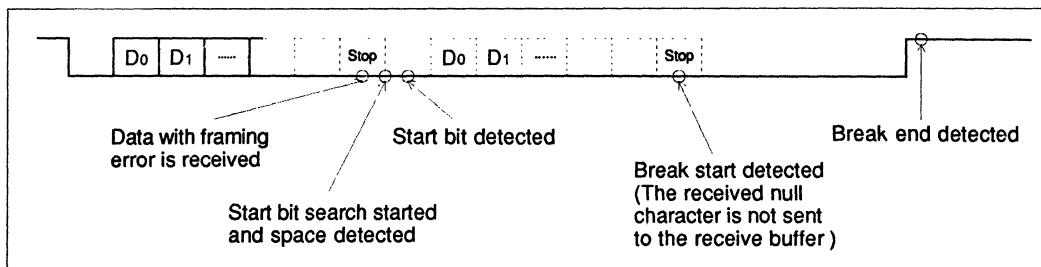


Figure 4-20. Break Detection by the Receiver

If a mark is detected for a half-bit cycle or longer after detecting the start of a break, this is assumed to be the end of a break and the BRKE bit in MST1 is set.

In the 1/1 clock mode, detection of the first mark causes the break to end.

When the BRKD or BRKE bit is set, an internal interrupt is generated (if enabled).

- **Supplementary explanation**

A break is generally transmitted using the following procedure:

- ① Wait for the end of transmission (idle status)
- ② Write 1 to the BRK bit
- ③ Wait one or more character cycles
- ④ Write 0 to the BRK bit

**Multiprocessor Support:** The MSCI supports a function which specifies whether a specific terminal should receive data in character unit or ignore it. This is useful for communications between multiple terminals.

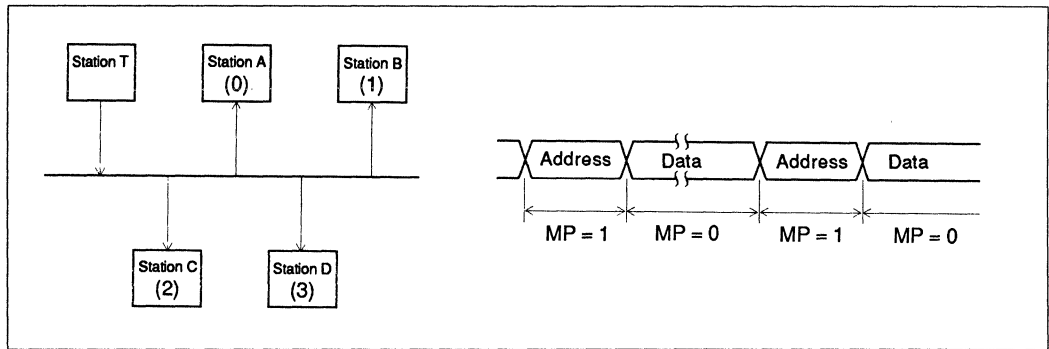
In multiprocessor mode (when using character format) an MP bit is used instead of a parity bit. The PMPM1-0 bits in MMD1 are used to specify this format.

In the multiprocessor mode, data is normally transmitted with the MP bit set to 0. The MP bit can be set to 1 by issuing an MP-bit-on-command immediately before transferring the transmit data to the transmit buffer. This command affects only one data transmission after issuing command.

On the receiver side, the MP bit in the receive data is transferred to the receive buffer together with other status information. When the receive data becomes ready to read, the value of the MP bit is set in MST2. Data whose MP bit = 0 can be ignored (not transferred to the receive buffer) by issuing a search MP bit command. This command is invalidated when data whose MP bit = 1 is received, and subsequent data is received in the normal manner.

For information on the "MP bit on" and "search MP bit" command, see section 4.2.8 "MSCI Command Register."

Figure 4-21 shows how communications are accomplished between multiprocessors using the MP bit.



**Figure 4-21. Sample MP Bit Operation**

In figure 4-21, T is a transmit station, and A, B, C, and D are receive stations. Receive stations A, B, C, and D are assigned addresses 0, 1, 2, and 3, respectively.

When transmitting data from T to B, transmit station T sends address (1) with MP bit = 1 to the communications path.

The receive stations all monitor the communications path. When they receive data with the MP bit = 1, they assume that the data is a station address and compare it with their own address. In this example, the received data matches the address of station B. Station B now assumes that subsequent data (data with the MP bit = 0) is destined for it. Other receive stations, A, C, and D, issue a search MP bit command and ignore the data (with the MP bit = 0). Thus, the transmit station can send data to a specific receive station by transmitting the destination address with MP bit = 1 and then transmitting the data with the MP bit = 0.

If the transmit station wants to send data to a different receive station, it transmits the new station address with the MP bit = 1 to clear the search MP bit command. The transmit station can use the procedure described above to communicate with the desired receive station.

### 4.3.2 Byte Synchronous Mode

In byte synchronous mode, each character is synchronized by adding a SYN character to the beginning of the transmit or receive data.

The MSC1 byte synchronous mode supports the mono-sync, bi-sync, and external synchronous modes. In the mono- and bi-sync modes, one and two SYN characters are used for synchronization,

respectively. In the external synchronous mode, the  $\overline{\text{SYNC}}$  line is asserted to achieve synchronization. The byte synchronous mode is specified using the PRTCL 2-0 bits of MMD0.

Figure 4-22 shows the character format for byte synchronous mode.

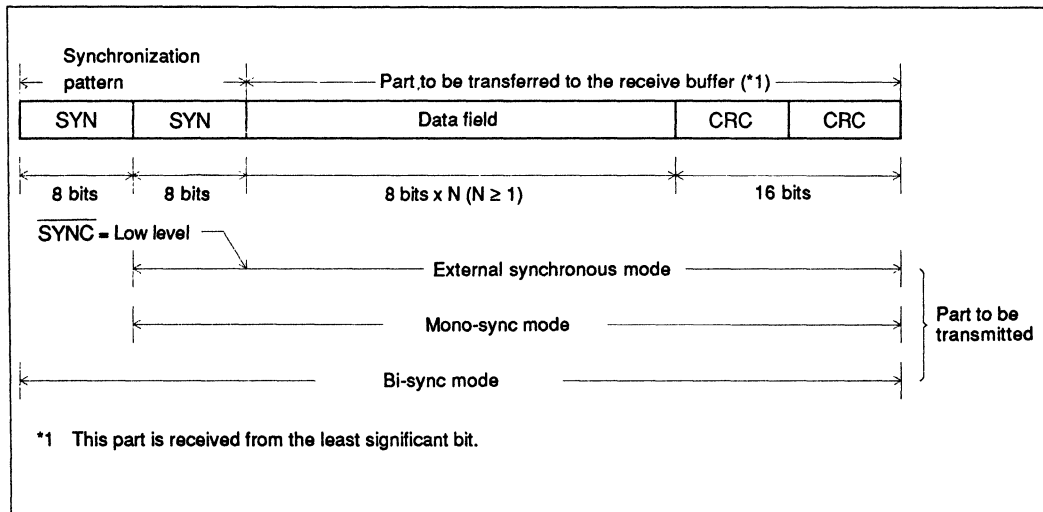


Figure 4-22. Character Format for Byte Synchronous Mode

Table 4-5 indicates the SYN character length for transmission and reception in the byte synchronous mode.

Table 4-5. SYN Pattern Length in Byte Synchronous Mode

Synchronous Mode	For Transmission	For Reception
Mono-sync	1 byte	1 byte
Bi-sync	2 bytes	2 bytes
External synchronous	1 byte	SYNC line used for synchronization

The SYN character pattern is set by MSCI synchronous/address registers 0 and 1 (MSA0-1).

When transmitting a header preceding the SYN character, write the header into the MSCI idle pattern register (MIDL) to delay data write to the transmit buffer. The transmitter keeps transmitting the header until data is written into the transmit buffer. (For details, see sections 4.2.4 "MSCI Control Register," 4.2.18 "MSCI Synchronous/Address Register 0," 4.2.19 "MSCI Synchronous/Address Register 1," and 4.2.20 "MSCI Idle Pattern Register.")

The receiver does not re-establish synchronization of the received data using SYN characters in the data field.

The SYN characters in the data field are automatically deleted or loaded into the receive buffer. The SYNC LD bit in the MSCI control register (MCTL) is used to make the selection. (For details, see section 4.2.4 "MSCI Control Register.")

**Transmission Operation:** Figure 4-23 shows the state transition diagram for byte synchronous transmission.

- Transmit disable state

The transmitter is placed in the transmit disable state by a hardware reset or a channel reset or transmit reset command.

In this state, the TXDM line remains at high level (mark), and the TXRDY bit in MST0 is cleared to 0.

- Idle state

The transmitter moves into the idle state from the transmit disable state with a transmit enable command.

In this state, the TXDM line transmits a different signal according to the value of the IDLC bit in MCTL. The signal is high (mark) when IDLC = 0; the contents of the MSCI idle pattern register (MIDL) are output when IDLC = 1. Once the transmit data is written, the transmitter enters SYN1 transmit state.

- SYN1 transmission state

Transmits the SYN characters set in MSA1. (For details, see section 4.2.18 "MSCI Synchronous/Address Register 0" and section 4.2.19 "MSCI Synchronous/Address Register 1.") After transmission, the transmitter enters the character transmission state in the mono-sync and external synchronous modes, or the SYN2 transmission state in the bi-sync mode.

- SYN2 transmission state

Transmits the SYN characters set in MSCI Synchronous/Address register 0 (MSA0) only in the bi-sync mode and enters the character transmission state. The transmitter does not enter this state in the mono-sync or external synchronous mode.

- Character transmission state

Transmits the data in the transmit buffer from the TXDM line.

- CRC transmission state

Transmits a 16-bit CRC code. If data remains in the transmit buffer after transmission, the transmitter enters the SYN1 wait state. If no data remains, it enters the idle state. (Set the CRC code in bits CRC1-0 of MMD0. Whether or not to perform CRC calculation and to send the result is specified by the CRCCC bit in MMD0. For details, see section 4.2.1 "MSCI Mode Register 0.")

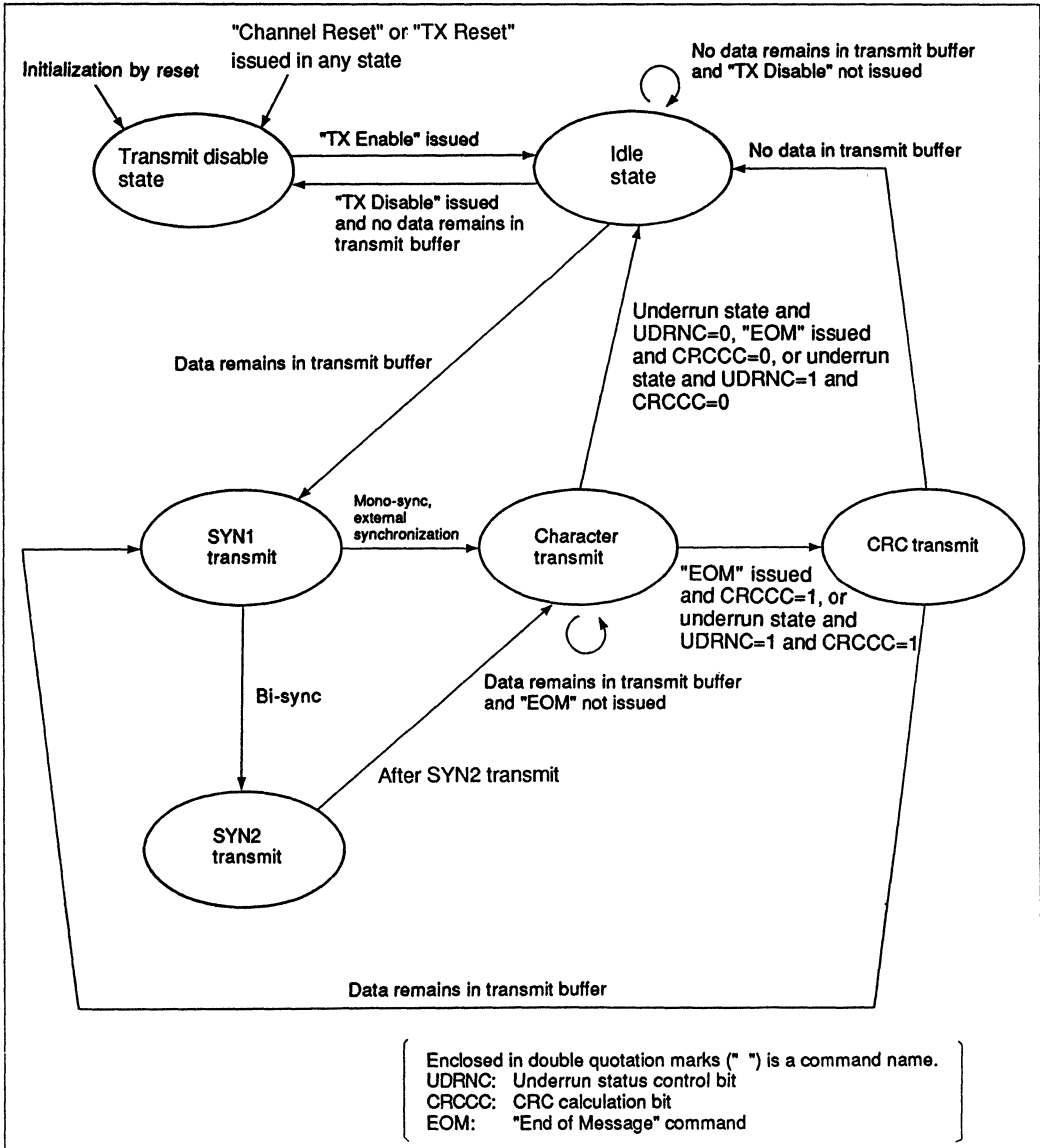
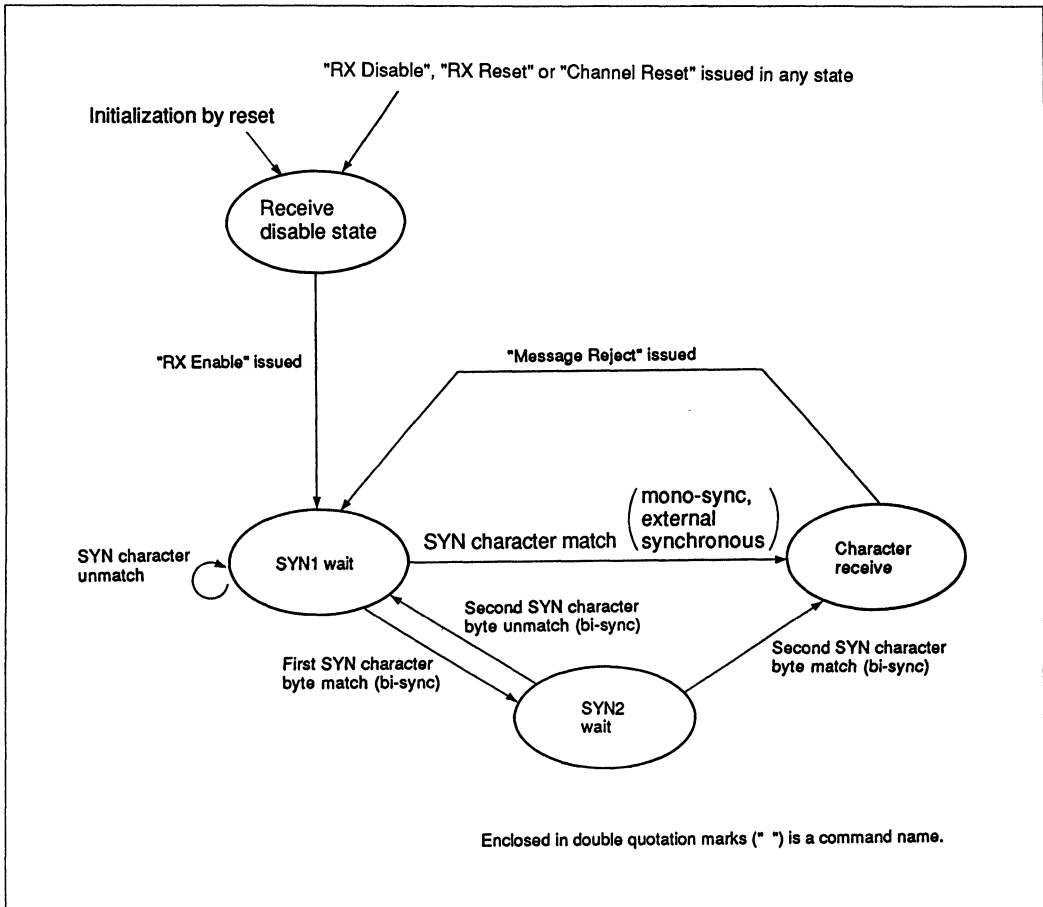


Figure 4-23. State Transition Diagram for Byte Synchronous Transmission

**Receive Operation:** Figure 4-24 shows the state transition diagram for reception.

- **Receive disable state**  
The receiver is placed in the receive disable state by a hardware reset, channel reset, receive reset or receive disable command.  
In this state, the receiver ignores input from the RXDM line and no reception operations are performed.
- **SYN1 wait state**  
Waits for the first SYN character byte to establish a character boundary.  
If the received data matches the SYN pattern set in MSCI Synchronous/Address register 0 (MSA0), the receiver enters the character reception state in the mono-sync mode, or the SYN2 wait state in the bi-sync mode. In the external synchronous mode, synchronization is established by the  $\overline{\text{SYNC}}$  line input.
- **SYN2 wait state**  
Waits for the second SYN character byte (only in the bi-sync mode). If the received data matches the SYN pattern set in MSA1, the receiver enters the character reception state. If it does not match, the receiver enters the SYN1 wait state. The MSCI does not enter this state in the mono-sync or external synchronous mode.
- **Character reception state**  
Transfers the received character to the receive buffer.  
The SYN character(s) in the data field may be transferred to the receive buffer, or not, as specified by the SYNCLD bit in MCTL.  
The receiver is placed in the SYN1 wait state by issuing a message reject command.



**Figure 4-24. State Transition Diagram for Byte Synchronous Reception**

## Error Checking

- CRC errors and CRC code transmission

The MSCI supports two CRC code types: CRC-16 and CRC-CCITT. The type to be used and the initial value (all 0s or 1s) are program-selectable. Use bits CRC1-0 of MMD0 for this purpose.

The CRC polynomial is  $X^{16} + X^{15} + X^2 + 1$  for CRC-16 and  $X^{16} + X^{12} + X^5 + 1$  for CRC-CCITT.

The MSCI transmitter and receiver both have a CRC calculator.



The TX CRC calculator is automatically initialized immediately before transmitting the data field. It can also be initialized by issuing a TX CRC initialization command.

During data transmission, synchronous patterns are excluded from the CRC calculation. Data can be excluded (in a character unit) from the CRC calculation by a TX CRC calculation exclusion command. Use the CRCCC bit in MMD0 and the end of message command to specify transmission of the CRC code. The CRC code is transmitted automatically when both the CRCCC bit and the UDRNC bit in MCTL are set to 1 and when the underrun state is detected.

For details, see section 4.2.1 "MSCI Mode Register 0," section 4.2.4 "MSCI Control Register," and section 4.2.8 "MSCI Command Register."

If an underrun occurs while UDRNC = 0 or CRCCC = 0, the MSCI directly enters the idle state without transmitting the CRC code.

The RX CRC calculator is automatically initialized immediately before receiving a data field. It can also be initialized by issuing an RX CRC initialization command.

During data reception, the characters not input to the receive buffer, such as SYN characters, are excluded from the CRC calculation. Data can be excluded (in a character unit) from the CRC calculation by specifying an RX CRC calculation exclusion command. The CRC code check is completed 15 system clock cycles after the character following the last check character has entered the receive buffer. If a CRC calculation forcing command is issued (the character following the last character does not enter the receive buffer), the check is completed 15 system clock cycles after issuing the command. In either case, the CRC error status is valid until the next character enters the receive buffer.

If a CRC error is detected, CRCE bit in MST2 is set to 1. See section 4.2.11 "MSCI Status Register 2."

When the CRCE bit is set, an internal interrupt is generated (if enabled).

- **Overrun error**

An overrun error occurs if the receive buffer is full when new data is sent to the buffer. When an overrun error occurs, the new data is written into the last stage of the receive buffer and the previous data is lost. The last stage of the status FIFO is overwritten with the status (including an overrun indication) of the new data.

The OVRN bit in MST2 is set to 1 when the overwritten data becomes available for read.

The OVRN bit status can be cleared to 0 only when 1 is written to it by the CPU or by a reset. When the OVRN bit is set, an internal interrupt is generated (if enabled).

Even after an overrun error is detected, character reception continues.

- Underrun error

An underrun error occurs if the transmit buffer becomes empty after data has been sent from the transmit shift register.

When an underrun error is detected\*1 the transmitter enters the idle state. Then the transmit line goes high (by clearing the IDLC bit of MCTL) or outputs an idle pattern (set by setting the IDLC bit.) At this time, the CRC code can be transmitted before entering the idle state by setting the UDRNC bit in MCTL to 1\*2.

When an underrun error is detected, the UDRN bit of MST1 is set to 1 and the TXRDY bit of MST0 is cleared to 0. The UDRN bit can be cleared to 0 only when 1 is written to it by the CPU or by a reset.

An internal interrupt is generated (if enabled) when the UDRN bit is set.

After entering the idle state, the MSCI enters the SYN1 transmit state when the UDRN bit is cleared and data is written into the transmit buffer.

\*1 An underrun error is assumed when the transmit shift register and transmit buffer are both empty and an end of message command has not been issued.

\*2 If an underrun error occurs while UDRNC = 0 or CRCCC = 0, the MSCI enters the idle state directly without transmitting the CRC code.

**End of message:** To signal the end of a message, use an end of message command. An end of message is also assumed if an underrun error occurs while the UDRNC bit in MCTL is set to 1.

When the message ends and the CRCCC bit of MMD0 is 1, the transmitter automatically transmits a CRC code and then enters the idle state. If the CRCCC bit is 0, the transmitter enters the idle state without CRC code transmission. This will generate an internal interrupt (if enabled).

During receive operations, the receiver does not perform end of message detection.

### 4.3.3 Bit Synchronous Mode

In the bit synchronous mode, the end of a frame is indicated by flag. The PRTCL2-0 bits of MMD0 are used to specify the bit synchronous mode.

Figure 4-25 shows the message format for the bit synchronous mode.

The A (address) and C+I (control and information) fields are configured in byte units and are sent to the receive buffer. Data, except the frame check sequence (FCS) field, is transmitted or received beginning with the least significant bit. (The FCS field data is transmitted and received beginning with the most significant bit.)

Residue bit frames cannot be transmitted. For reception, if residue bits exist at the end of receive data, the valid bits (residue bits) in the last character are justified to the upper positions and the lower bits are undefined. The undefined bits cannot be distinguished from valid bits. When a residue bit frame is received, the status of the last character indicates both the residue bit frame and end of receive frame status. (This status is indicated by the EOM and RBIT bits in MST2.)

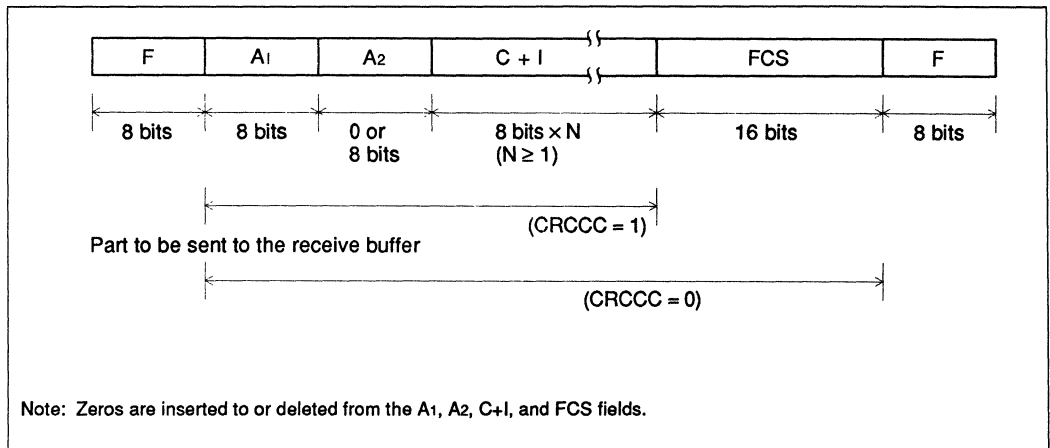


Figure 4-25. Message Format for Bit Synchronous Mode

#### Transmission Operation:

Figure 4-26 shows the state transition diagram for transmission in the bit synchronous HDLC mode.

- **Transmit disable state**

The transmitter is placed in the transmit disable state by a hardware reset, a channel reset or TX reset command.

The TXDM line goes to high level (mark), and the TXRDY bit of MST0 is cleared.

- **Idle state**

The transmitter is placed in the idle state from the transmit disable state by a transmit enable command.

In this state, the TXDM line behaves according to the setting of the IDLC bit in MCTL. A high-level signal (mark) is output when IDLC = 0; the contents of the MSCi idle pattern register (MIDL) are output when IDLC = 1. When transmit data is written, the transmitter enters the opening flag transmission state.

- **Opening flag transmission state**

Transmits one flag, then immediately enters the character transmission state.

- **Character transmission state**

Sequentially transmits the data in the transmit buffer.

- **FCS transmission state**

Transmits FCS (CRC), then enters the next state.

- **Closing flag transmission state**

Transmits one flag, then enters the next state.\*

- **Abort transmission state**

Transmits the abort pattern 11111111, then enters the next state.

\* When frames are sent in succession, they are automatically delimited by at least one closing flag and one opening flag.

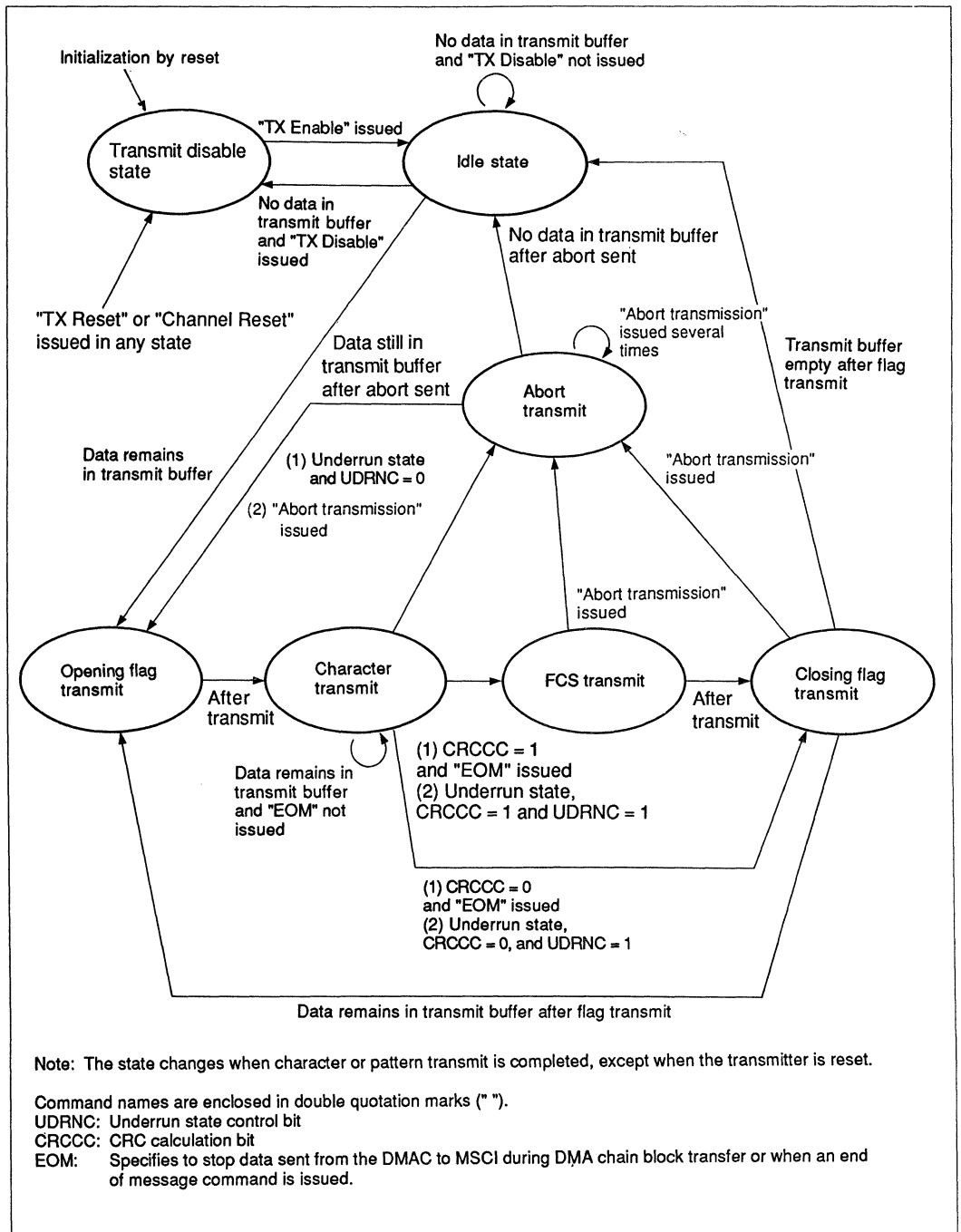


Figure 4-26. State Transition Diagram for Transmission in Bit Synchronous HDLC Mode

**Receive Operation:** Figure 4-27 shows the state transition diagram for reception in the bit synchronous mode.

- **Receive disable state**

The receiver is placed in the receive disable state by a hardware reset, a channel reset or a receive reset command.

The receiver ignores the input from the RXDM line and no reception operations are performed.

- **Flag wait state**

Compares the received bit string with the flag pattern until a match is detected.\*

When the flag pattern is detected, the character wait state is entered.

- **Character wait state**

Ignores successive flags which indicate a frame boundary and waits for a non-flag pattern.

When non-flag pattern is detected, the address field check state is entered.

- **Address field check state**

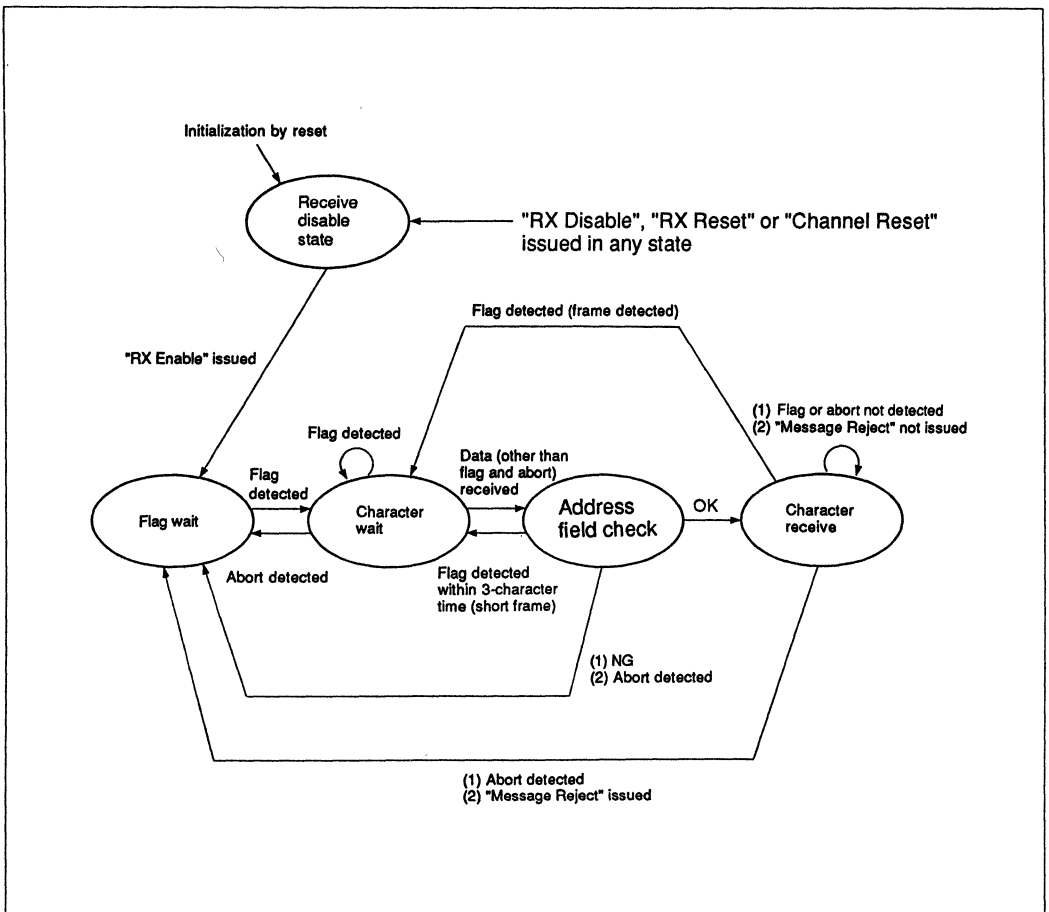
Checks the address field to determine whether or not to receive the associated frame. When the address field check is successful (the frame is accepted), the receiver enters the character reception state. When the address field check fails, the flag wait state is entered. In the address field no-check mode, this check is not performed and the character reception state is entered directly. If a flag is detected within three character cycles after the address field check, the received bit is assumed to be a short frame, and the character wait state is entered.

- **Character receive state**

Sends received characters to the receive buffer.

If the MSCI detects a flag in the character receive state, it sends characters to the receive buffer until the last character in the I field (CRCCC = 1) or the FCS (CRCCC = 0) has been sent. Then, the character wait state is entered.

\* Successive frames which share opening and closing flags can be received normally.



**Figure 4-27. State Transition Diagram for Bit Synchronous Reception**

**Error Checking:**

- CRC errors

In the bit synchronous HDLC mode, the initial value is normally set to all 1s using CRC-CCITT.

Use bits CRC1-0 in MMD0 for this purpose. (The CRC polynomial is  $X^{16} + X^{12} + X^5 + 1$  for CRC-CCITT.)

The transmitter and receiver both have a CRC calculator.

The CRC code is initialized immediately before transmitting or receiving the A field.

For transmission, CRC calculation is carried out on the data in the A, C, and I fields before zero insertion.

Use the CRCCC bit in MMD0 and the end of message command to enable CRC code transmission. The CRC code is transmitted automatically when both the CRCCC bit and the UDRNC bit in MCTL in underrun status are 1. (For details, see section 4.2.1 "MSCI Mode Register 0," section 4.2.4 "MSCI Control Register," and section 4.2.8 "MSCI Command Register.")

For reception, CRC calculation is carried out on the 0-deleted data in the A, C, and I fields. The CRC code check is completed when the last character in the I field enters the receive buffer with CRCCC bit in MMD0= 1. The error status is sent (via the status FIFO associated with the character) to the CRCE bit in MST2. When the CRCE bit is set, an internal interrupt request is generated (if enabled). If the CRCCC bit is 0, the CRCE bit is not set.

- **Overrun errors**

An overrun error occurs when the receive buffer is full when new data arrives. When an overrun error occurs, the new data is written into the last stage of the receive buffer (TRB) and previous data is lost. The last stage of the status FIFO is overwritten by the status (including an overrun status) of the new data (the OVRN bit in MST2 is set when overwritten data becomes available for read). The EOM bit is also cleared by overwriting.

The overrun status can be cleared only by writing 1 to the OVRN bit or by a reset. When the OVRN bit is set, an internal interrupt is generated (if enabled).

Character reception is not stopped by overrun detection.

- **Underrun errors**

An underrun error occurs if the transmit buffer becomes empty after sending data from the send shift register.

When an underrun error is detected and if abort transmission has been specified by the UDRNC bit in MCTL, the transmitter enters the idle state after sending an abort. In other cases, this is assumed to be the end of a message and the frame ends normally. Thus, MSCI transits to the idle state after sending FCS and a flag. (An underrun error is assumed when the transmit shift register and transmit buffer are empty and an end of message command has not been issued.)



The UDRN bit in MST1 is set to 1 when an underrun is detected. In this case, the transmit buffer is not full, but the TXRDY bit in MST0 is not set as long as the UDRN bit remains set. This prevents the remaining data from being transmitted as an ordinary frame when an underrun occurs during DMA transmission.

When the UDRN bit is 1, an internal interrupt is generated (if enabled).

**Message End Operation:** To signal the end of a message, use an end of message command. An end of message is assumed when either a DMA chain block transfer has been completed, or when an underrun occurs when the UDRNC bit in MCTL is 1.

The last character to be transmitted is the first character written into the transmit buffer after issuing the end of message command; for DMA chained block transfer, it is the last character received. If an underrun occurs, it is the character transmitted immediately before the underrun.

When the message transmission is complete, the MSCI enters the closing flag transmit state provided the CRCCC bit MMD0 is 0. If the CRCCC bit is 1, the MSCI enters the FCS transmit state.

For reception, the end of message is assumed when a flag is detected in the character receive state. While the CRCCC bit of MSCI mode register 0 (MMD0) is 1, characters up to and including the last character in the I field are sent to the receive buffer and FCS is deleted. The associated receive frame end status and CRC error status are sent to the status FIFO, and set to the EOM bit and CRCE bit of MSCI status register 2 (MST2) when the last character becomes available for read. At the same time the internal DMAC is informed of the end of frame and an internal interrupt is generated (if enabled).

When the CRCCC bit is 0, FCS is also sent to the receive buffer. In this case, its associated receive frame end status is transferred to the status FIFO.

To enable this control, characters are sent to the receive buffer and wait there for three character cycles after being received. When the closing flag is detected, the last character in the I field and the FCS have not yet been sent to the receive buffer.

**Address field check:** In the bit synchronous mode, data frames contain an address (A) field which specifies what secondary station(s) should receive the frame. The MSCI supports four address field check modes: address field no-check, single address 1, single address 2, and dual address. (See table 4-6.)

**Table 4-6. Address Field Check**

Mode	Function
Address field no-check	Receives all frames
Single address 1	Receives only frames whose A1 field has the specified value or global address (0FFH).
Single address 2	Receives only frames whose A2 field has the specified value or global address (0FFH).
Dual address	Receives only frames whose A1 and A2 fields have the specified value, global address (0FFFFH), or group address (A2 = specified value, A1 = 0FFH).

Use bits ADDR1-0 in MMD1 to specify the address field check and SA0 and 1 in MSA0 and MSA1 to specify the address. For details, see section 4.2.2 "MSCI Mode Register 1," section 4.2.18 "MSCI Synchronous/Address Register 0," and section 4.2.19 "MSCI Synchronous/Address Register 1."

**Short frame detection:** If a short frame is received, the action taken depends on the frame length, CRCCC bit value of MSCI mode register 0 (MMD0), and address field check as shown in table 4-7.

**Table 4-7. Reactions to Short Frame Detection**

Frame Length (excluding flag)	Mode Settings			
	CRCCC bit = 0		CRCCC bit = 1	
	Address Field No-Check	Single Address 2	Address Field No-Check	Single Address 2
Bits 1 – 8	Sends no data to the receive buffer.	Sends no data to the receive buffer.	Sends no data to the receive buffer.	Sends no data to the receive buffer.
Bits 9 – 23	Sends a part of the data to the receive buffer. Appends the short frame status to the last character and sets the SHRT bit in MST2.	Sends a part of the data to the receive buffer. Appends the short frame status to the last character and sets the SHRT bit in MST2.	Sends no data to the receive buffer.	Sends no data to the receive buffer.

**Table 4-7. Reactions to Short Frame Detection (cont.)**

Frame Length (excluding flag)	Mode Settings			
	CRCCC bit = 0		CRCCC bit = 1	
	Address Field No-Check Single Address 1	Single Address 2 Dual Address	Address Field No-Check Single Address 1	Single Address 2 Dual Address
Bits 24 – 31	Sends a part of the data to the receive buffer. Appends the short frame status to the last character and sets the SHRT bit in MST2.	Sends a part of the data to the receive buffer. Appends the short frame status to the last character and sets the SHRT bit in MST2.	Sends a part of the data to the receive buffer. Appends the short frame status to the last character and sets the SHRT bit in MST2.	Sends a part of the data to the receive buffer. Appends the short frame status to the last character and sets the SHRT bit in MST2.
Bits 32 – 39	Receives the data as normal data.	Sends a part of the data to the receive buffer. Appends the short frame status to the last character and sets the SHRT bit in MST2.	Receives the data as normal data.	Sends a part of the data to the receive buffer. Appends the short frame status to the last character and sets the SHRT bit in MST2.
Bits 40 and up	Receives the data as normal data.	Receives the data as normal data.	Receives the data as normal data.	Receives the data as normal data.

**Note:** When a short frame is detected and the SHRT bit of MSCI status register 2 (MST2) is set, the EOM bit is set, indicating the end of receive frame. At this time, an internal interrupt is generated (if enabled).

If data has not been transferred to the receive buffer, the SHRT bit is not set, even if a short frame is detected.

**Abort Transmission and Reception:** Use the abort transmit command to abort transmission. If abort transmission is selected (UDRNC = 0) using the UDRNC bit in MCTL, when an underrun occurs MSCI transmitter automatically enters the abort transmission state.

This state causes an abort pattern (eight 1s) to be transmitted in order to clear the transmit buffer. Thus, the contents of the transmit shift register and transmit buffer are lost. After transmitting the abort pattern, the MSCI enters the idle state.

During receive operations, the MSCI assumes 01111111 (0 followed by seven 1s) as an abort. When an abort is detected, an interrupt is generated (if enabled) and the receiver enters the flag wait state.

If the receiver is in the character receive state when the abort is detected, it carries out the following additional operation:

When the CRCCC bit is 0, data up to the position preceding 01111111 is sent to the receive buffer. When the CRCCC bit is 1, data, up to the character being assembled at detection, is sent to the receive buffer and 16 bits of data preceding 01111111 is truncated. (This operation is the same as for receive frame ending upon flag detection, except that the ABT bit of MSCI Status Register 2 is set to 1.)

#### 4.3.4 Bit Synchronous Loop Mode

The bit synchronous loop mode supports secondary stations for bit synchronous loop transmission of data. The PRTCL 2-0 bits of MSCI Mode Register 0 (MMD0) are used to specify this mode. The primary station is operated in the bit synchronous HDLC mode. Figure 4-28 shows the relationship between the primary and secondary stations in the bit synchronous loop mode.

The bit synchronous loop mode is the same as the HDLC mode except for some transmit operations and the address field check. See section 4.3.3 "Bit Synchronous Mode."

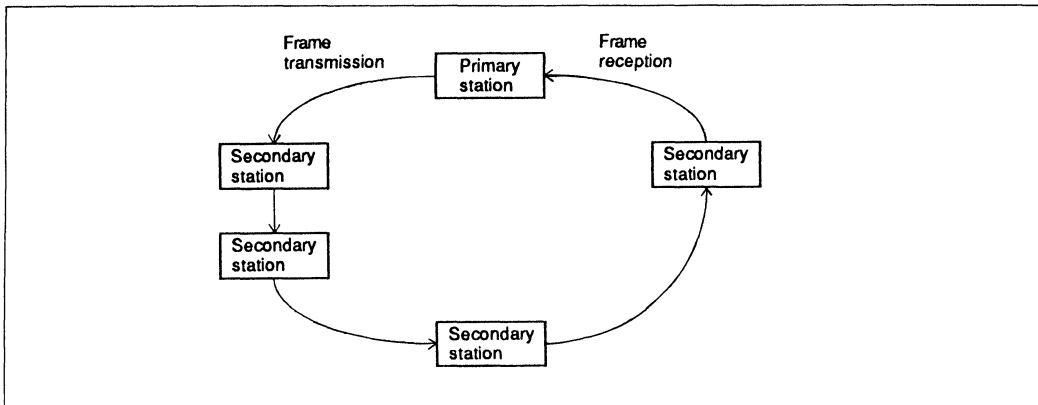


Figure 4-28. Relationship between Primary and Secondary Stations in Bit Synchronous Loop Mode

The primary station transmits a frame followed by an idle pattern, 11111111. It repeatedly transmits the idle pattern until the transmitted frame, the response frame from a secondary station, and an idle pattern are received from the secondary station.

Since the primary station's idle pattern is 11111111, a go-ahead (GA) pattern, shown in figure 4-29, is formed at the boundary of flag 01111110. This GA pattern requests the secondary station to transmit data.

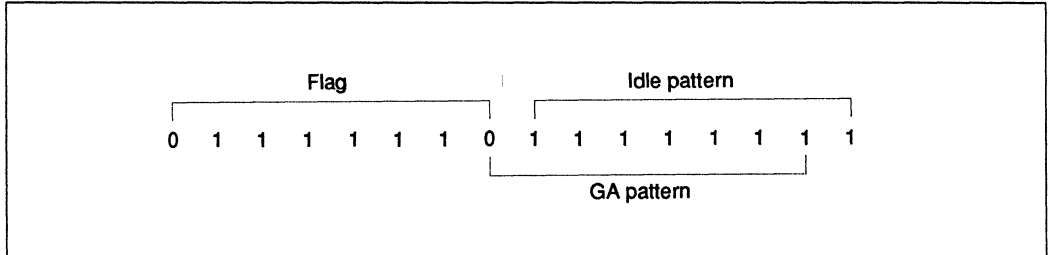
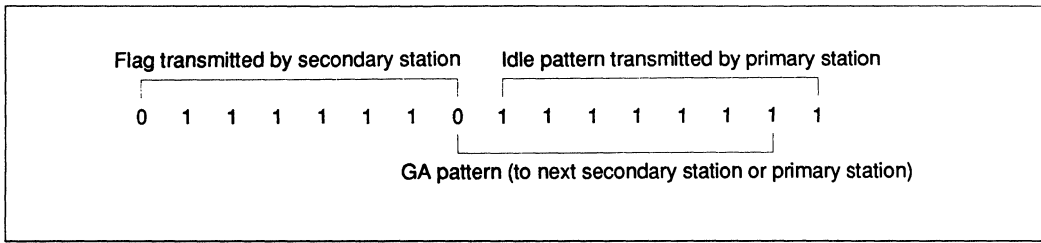


Figure 4-29. GA Pattern

The secondary station is usually in the retransmit idle state. It retransmits the received data after a 1-bit cycle delay. For transmission by polling from the primary station, a secondary station sets the GOP bit in MCTL to 1 (end of frame) and waits for the next GA pattern before starting transmission.

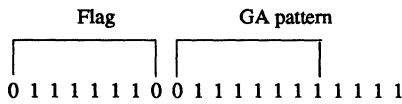
If the secondary station detects a GA pattern while the GOP bit is 1, it changes the last 1 of the GA pattern to 0, thus turning it into a flag pattern. The secondary station begins transmitting frames if transmit data exists. If no transmit data exists, the secondary station repeatedly transmits the idle pattern specified by the MSCI idle pattern register until transmit data is written. Therefore, the idle pattern must match that of a flag, 01111110.

When the secondary station completes data transmission, it transmits a flag and returns to the retransmit idle state. Thus, a GA pattern is automatically generated.\* See figure 4-30.



**Figure 4-30. GA Pattern Generation at the End of Data Transmission in the Secondary Station**

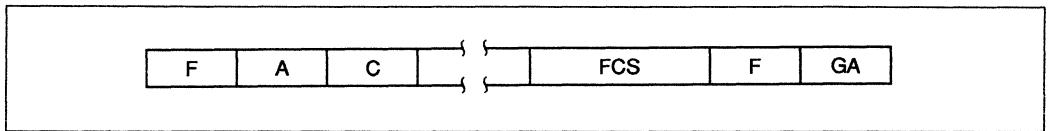
\* The following procedure can be used to avoid shared use of the 0s at the end of the flag and the beginning of the GA pattern.



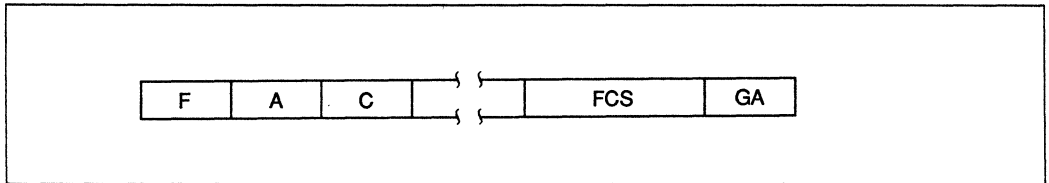
- ① When the primary (or secondary) station is transmitting, set the GA pattern (11111110) in the MSCI idle pattern register (MIDL).
- ② After the closing flag has been transmitted, place the transmitter in idle state; the idle pattern (GA pattern) will be output. (For the secondary station, it is necessary at this point that the GOP bit be left set to 1.)
- ③ While the GA pattern is being transmitted, write 11111111 to MIDL. This is then transmitted as the idle pattern after the GA pattern. (For the secondary station, it is necessary to clear the GOP bit to 0 after the idle pattern is transmitted in order to place the station in the retransmit idle state.)

When a flag is received during transmission, the secondary station must stop transmission. By polling the flag detection status or in response to an interrupt from the CPU, the secondary station can stop transmitting and return to the retransmit idle state. The primary station can also request the secondary station to stop transmitting. The primary station writes flag pattern 01111110 into the MSCI idle pattern register (MIDL) and transmits this flag by setting the IDLC bit in MCTL to 1. GA pattern and flag detection status are set in the GAPD and FLGD bits of MSCI Status Register 1 (MST1). At that time, an interrupt is generated (if enabled).

Figure 4-31 (a) shows a normal receive format in the bit synchronous mode. Figure 4-31 (b) shows an abort end frame format (the GA and abort patterns are identical).



**Figure 4-31. (a) Normal Receive Format**



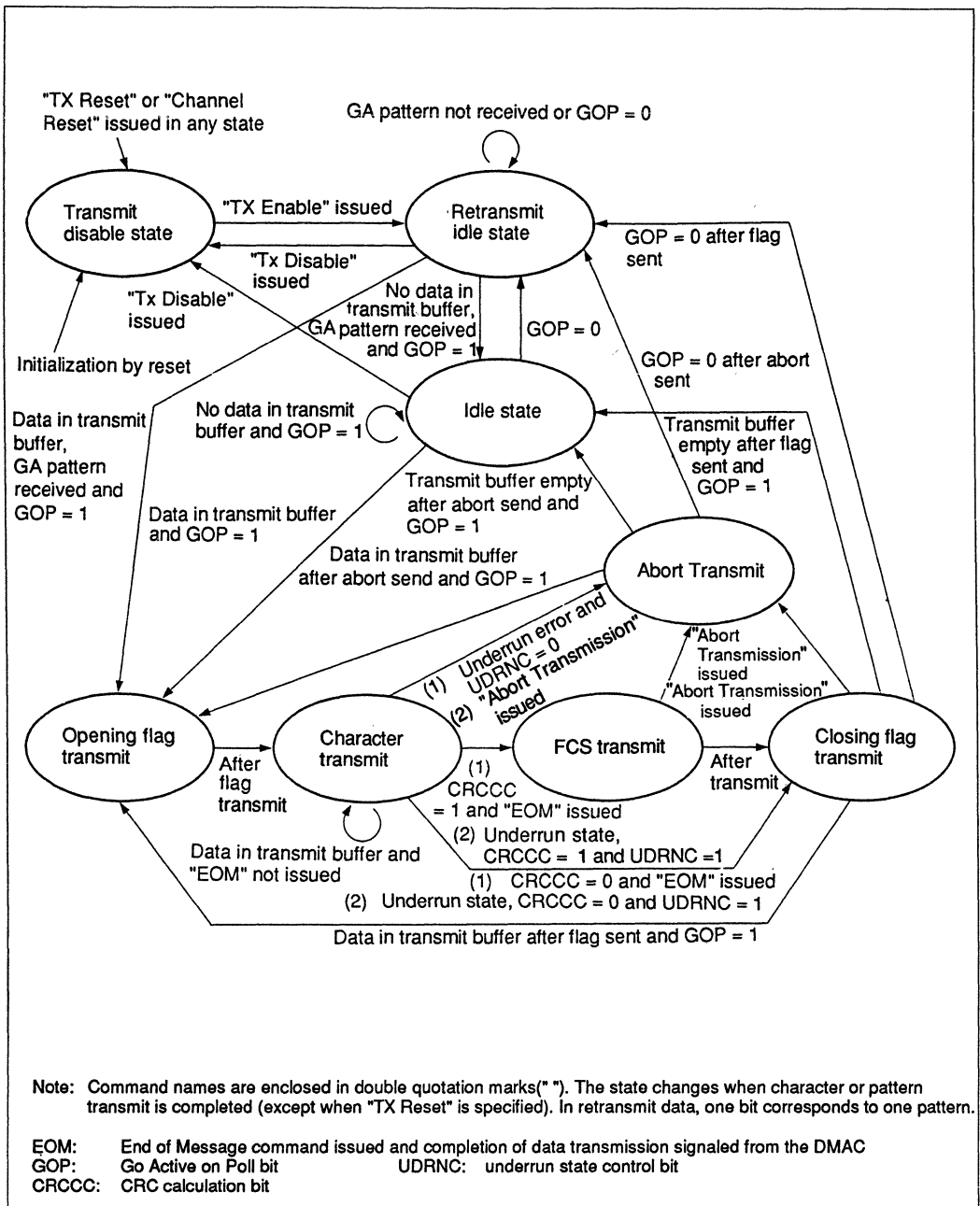
**Figure 4-31. (b) Abort End Frame Format**

**Transmission operation:** Transmission operations in the bit synchronous loop mode differ from those in the bit synchronous HDLC mode. Figure 4-32 shows the state transition diagram for transmission in the bit synchronous loop mode.

- **Transmit disable state**  
 The transmitter is placed in the transmit disable state by a hardware reset, a channel reset or a TX reset command.  
 The TXDM line goes to the high level (mark), and the TXRDY bit of MSCI status register 0 (MST0) is cleared to 0.
- **Retransmit idle state**  
 The transmitter enters the retransmit idle state from the transmit disable state by the TX enable command and retransmits received data after a delay of one bit cycle.
- **Idle state**  
 The transmitter repeatedly transmits high level (mark) signal (IDLC = 0) or the MSCI idle pattern register (MIDL) contents (IDLC = 1), from the TXDM line until data is written into the transmit buffer or the GOP bit of MCTL is cleared to 0.
- **Opening flag transmission state**  
 Transmits an opening flag and enters the character transmit.

- Character transmission state  
Sequentially transmits the data in the transmit buffer.
- FCS transmission state  
Transmits the FCS (CRC) and enters the next state.
- Closing flag transmission state  
Transmits a closing flag and enters the next state.
- Abort transmission state  
Transmits abort pattern 11111111 and enters the next state.





**Figure 4-32. State Transition Diagram for Transmission in Bit Synchronous Loop Mode**

**Address Field Check:** The bit synchronous loop mode supports four address field checks: address field no-check, single address 1, dual address, and 4-bit address. The first three modes are the same as those available in the bit synchronous HDLC mode. For details, see section 4.3.3 "Bit Synchronous Mode."

Table 4-8 shows the 4-bit address mode.

**Table 4-8. 4-bit Address Mode Field Check Function**

<b>Mode</b>	<b>Function</b>
4-bit address	Receives only frames whose four high order bits in the A1 field are set to the specified value or global address (FH).

## 4.4 Transmit/Receive Clock Selection

### 4.4.1 Overview

The MSCI transmit and receive clock sources are selected from among the following sources:

- Transmit clock sources
  - TXCM line input
  - Transmit baud rate generator output
  - Receive clock

The transmit clock source is selected using the TXCS 2-0 bits of the MSCI TX clock source register (MTXS).

- Receive clock sources
  - RXCM line input
  - Receive baud rate generator output
  - RXCM line input with noise suppression by the ADPLL (the ADPLL operating clock is the receive baud rate generator output).
  - Clock extracted from the receive data by the ADPLL (the ADPLL operating clock is the RXCM line input or the receive baud rate generator output).

The receive clock source is selected using the RXCS 2-0 bits of the MSCI RX clock source register (MRXS).

The internal baud rate generator (BRG) can provide independent outputs for transmission and reception by dividing the system clock. The on-chip ADPLL can perform (1) clock extraction from the receive data, (2) noise suppression for the receive data and (3) noise suppression for the receive clock.

The ADPLL operating clock employs the receive BRG output or RXCM line input for clock extraction and noise suppression in the receive data. It uses the receive BRG output for noise suppression of the receive clock.

Figure 4-33 shows how the MSCI clock is supplied.

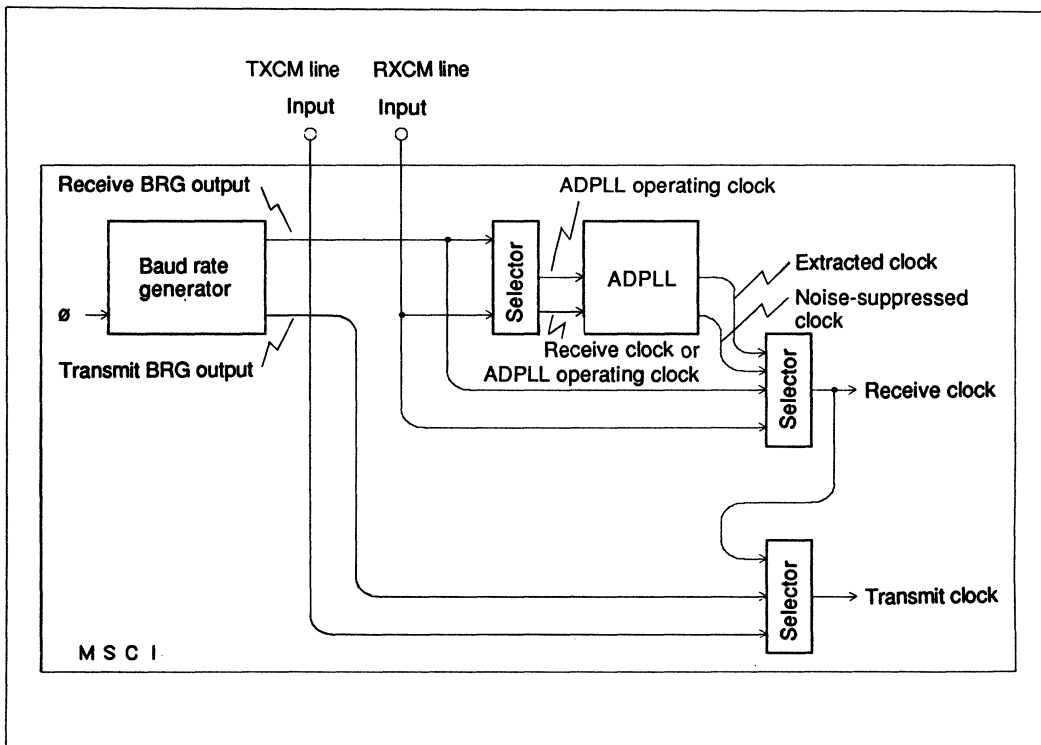


Figure 4-33. Selecting Transmit and Receive Clocks

#### 4.4.2 Supplying the Transmit Clock

Figure 4-34 shows transmit clock sources. When the transmit baud rate generator output is used as the transmit clock, the TXCM line functions as the transmit clock output.

The receive clock is used as the transmit clock in the following two cases:

- When the clock extracted by the ADPLL is used as the transmit clock.
- When the receive clock is used as the transmit clock in the bit synchronous loop mode.

In the asynchronous mode, the actual bit rate is determined by the clock mode (1/1, 1/16, 1/32 or 1/64).

In the byte or bit synchronous mode, 1/1 clock mode is automatically selected.

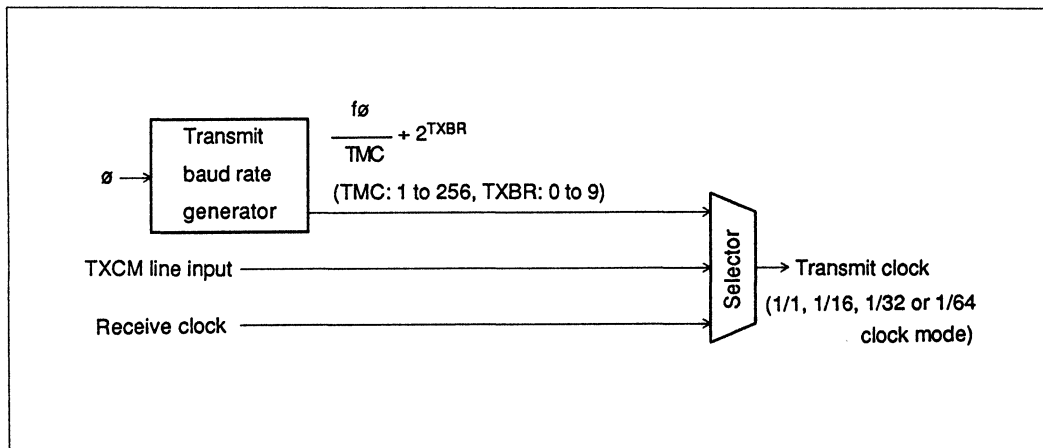


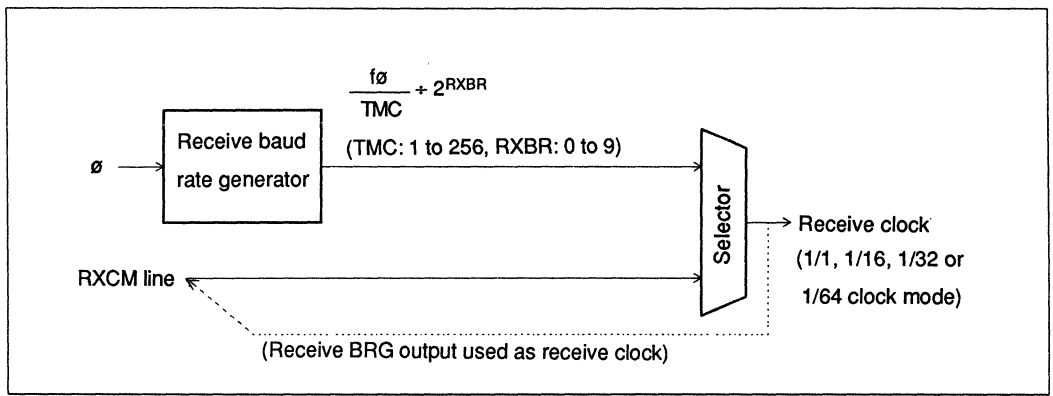
Figure 4-34. Transmit Clock Sources

#### 4.4.3 Supplying the Receive Clock

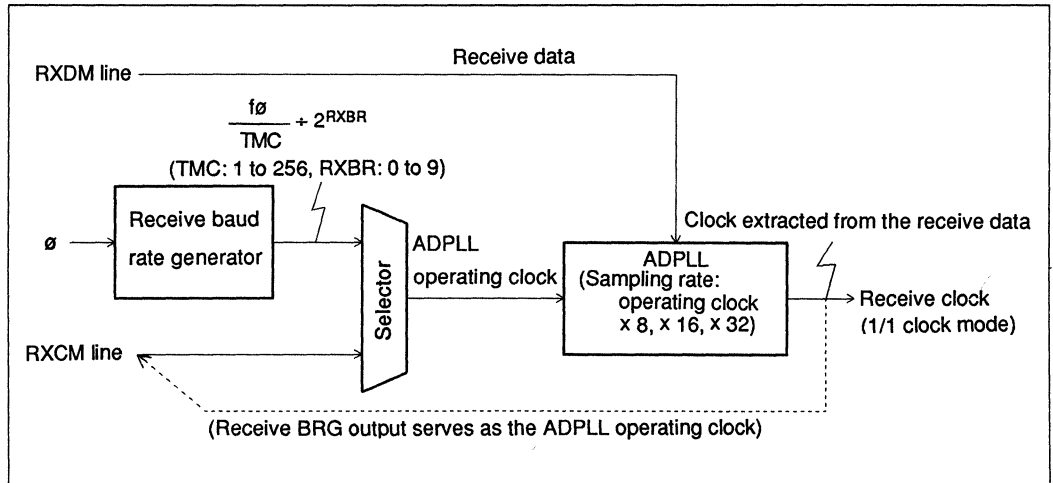
The receive clock sources are shown in figures 4-35 (a), (b) and (c).

When the RXCM line is not used as a clock source, it functions as the receive clock output.

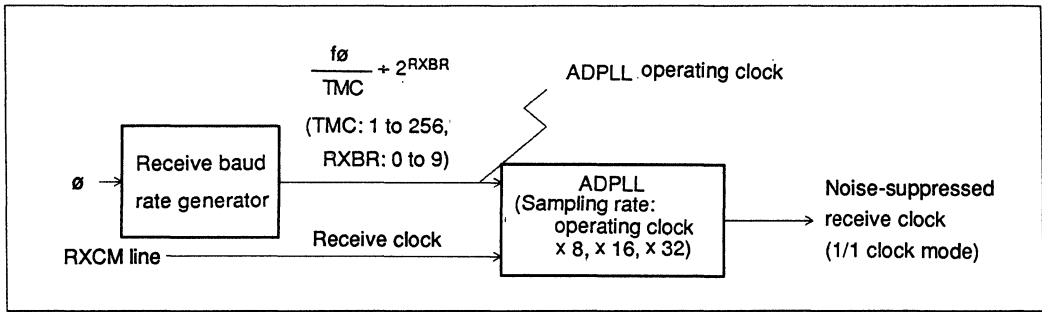
In the asynchronous mode, the actual bit rate is determined by the clock mode (1/1, 1/16, 1/32 or 1/64), and in the byte or bit synchronous mode, 1/1 clock mode is automatically selected.



**Figure 4-35. (a) Receive Clock Source**  
 (Receive BRG output or RXCM line input used as receive clock)



**Figure 4-35. (b) Receive Clock Source**  
 (clock extracted by ADPLL used as receive clock)



**Figure 4-35. (c) Receive Clock Source  
(Receive clock noise suppressed)**

#### 4.4.4 Baud Rate Generator

The output frequency of the baud rate generator for transmission and reception is obtained by the following equation:

$$f_{BRG} = \frac{f_{\phi}}{TMC} + 2^{BR}$$

where

}	f <sub>BRG</sub> : BRG output frequency
}	f <sub>ø</sub> : System clock frequency
}	TMC: MSCI time constant register value = 1 to 256
}	BR: TX clock source register TXBR 3-0 bit values (0 to 9) RX clock source register RXBR 3-0 bit values (0 to 9)

Frequencies determined by the above equation are independently output for transmission and reception from the baud rate generator.

#### 4.4.5 ADPLL

In byte or bit synchronous mode, the MSCI can use two kinds of receive clock : a clock extracted from the received data by ADPLL or RXCM input noise-suppressed by ADPLL.

The ADPLL has the following operating modes: ×8, ×16, and ×32 (ratio of the ADPLL operating clock rate to the bit rate).

To use the ADPLL clock extraction function, the operating clock frequency must be 8, 16, or 32 times the bit rate, regardless of whether the source of the operating clock is the RXCM line or the baud rate generator. The DRATE1-0 bits of MSCI mode register 2 are used to select the ADPLL operating mode.

## 4.5 ADPLL

### 4.5.1 Overview

The advanced digital PLL (ADPLL) functions to extract clock signals from the receive data and generate a decoding clock for the receive data.

The ADPLL has the following features:

- Clock extraction from five receive data transmission code types: NRZ, NRZI, Manchester, FM0 and FM1
- The bit rate of the ADPLL clock is selectable from among the following ratios:  $\times 8$ ,  $\times 16$ , and  $\times 32$
- Receive data noise suppressor function
- Receive clock noise suppressor function

Figure 4-36 shows the block diagram of the ADPLL.

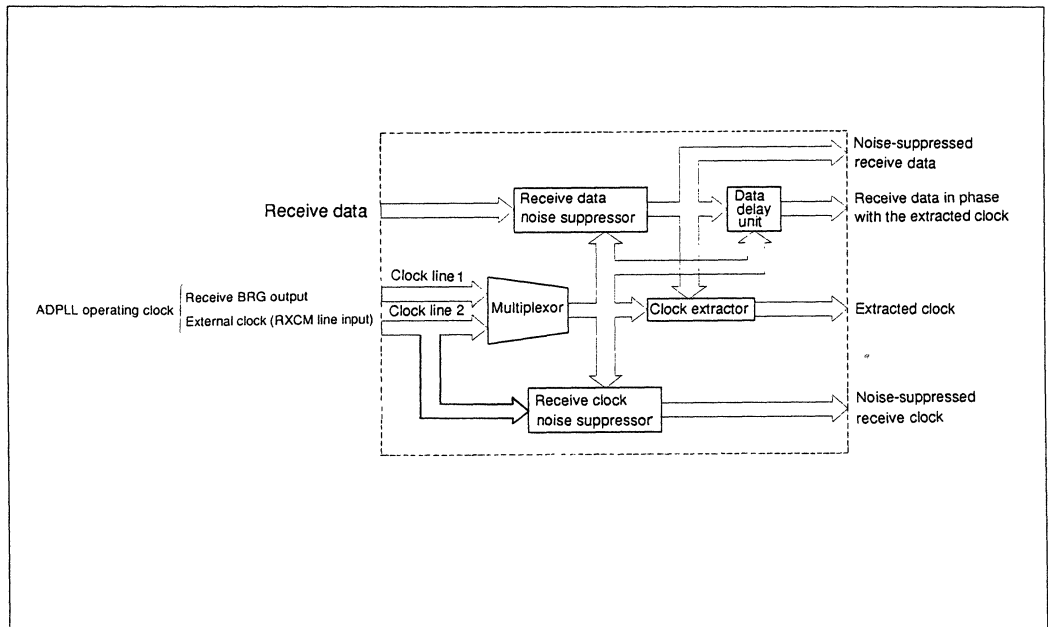


Figure 4-36. ADPLL Block Diagram

The ADPLL can selectively perform clock extraction for the receive data or noise suppression of the receive clock input from the RXCM line. In both cases, the receive data noise is suppressed.

The ADPLL receives the receive data and provides the operating clock. The ADPLL has two clock input lines: an input for the receive baud rate generator output and an input for the RXCM line input.

To extract the clock component from the receive data, operating the ADPLL uses the receive baud rate generator output or an external clock (RXCM line input) as the operating clock. The ADPLL clock is supplied to the receive data noise suppressor, clock extractor and the data delay unit, for use as a common operating clock. The extracted clock and the noise-suppressed receive data are sent from the ADPLL to the MSCI receiver. The extracted clock is used as the receive clock. When the output of the receive baud rate generator is used as the ADPLL clock, the RXCM line outputs this receive clock. (ADPLL operation is controlled by the RXCS 2-0 bits of the MSCI RX clock source register (MRXS).)

To function as a noise suppressor for the receive clock input from the RXCM line, the ADPLL uses the output of the receive baud rate generator as the operating clock. The ADPLL operating clock is supplied to the noise suppressors for the receive clock and the receive data, for use as a common operating clock. In this case, the clock extractor does not operate. The noise-suppressed receive data and the receive clock are sent from the ADPLL to the MSCI receiver.

The clock extraction from the receive data and noise suppression for the receive data and receive clock are based on the ADPLL operating clock. The ratio of the ADPLL clock to the bit rate can be selected from among  $\times 8$ ,  $\times 16$  and  $\times 32$  using the DRATE1-0 bits of MMD2.

Table 4-9 shows the relationship between the ADPLL clock and bit rates.

**Table 4-9. Relationship Between the ADPLL Operating Clock and Bit Rates**

Function	ADPLL Operating Clock Source	Operating Mode	Ratio of ADPLL Operating Clock Rate to Bit Rate
Clock extraction from receive data (receive data noise suppressed)	• RXCM line input	$\times 8$	8/1
	• Receiver BRG output	$\times 16$	16/1
		$\times 32$	32/1
Suppression of receive clock (receive data noise suppressed)	• Receiver BRG output	$\times 8$	8/1
		$\times 16$	16/1
		$\times 32$	32/1



The ADPLL supports a facility for adjusting the phase of the extracted clock. If the ADPLL clock is skewed by one or more cycles from the receive data that was passed via the data delay unit, this facility automatically adjusts it by  $\pm 1$  operation clock cycle. This compensation is repeated until the clock is synchronized.

ADPLL specifications are shown in table 4-10 and the transmission codes supported by the ADPLL are summarized in figure 4-37.

**Table 4-10. ADPLL Specifications**

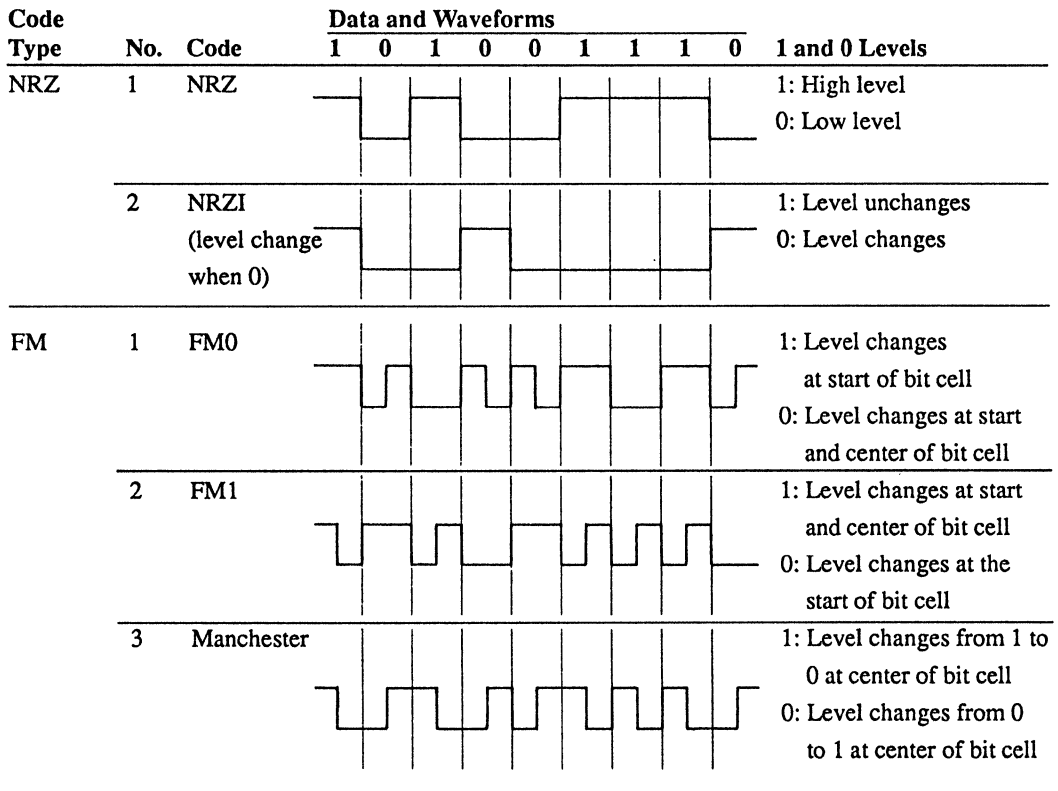
No.	Item	Mode		Specification	Remarks						
1	Maximum operating clock frequency			17.6 MHz							
2	Maximum bit rate	Operating mode		<table border="1"> <tr> <td>×8</td> <td>2.2 Mbps</td> </tr> <tr> <td>×16</td> <td>1.1 Mbps</td> </tr> <tr> <td>×32</td> <td>0.5 Mbps</td> </tr> </table>	×8	2.2 Mbps	×16	1.1 Mbps	×32	0.5 Mbps	
×8	2.2 Mbps										
×16	1.1 Mbps										
×32	0.5 Mbps										
3	Maximum number of level transitions necessary for synchronization	Code type	NRZ	<table border="1"> <tr> <td>×8</td> <td>4 level transitions</td> </tr> <tr> <td>×16</td> <td>8 level transitions</td> </tr> <tr> <td>×32</td> <td>16 level transitions</td> </tr> </table>	×8	4 level transitions	×16	8 level transitions	×32	16 level transitions	
×8	4 level transitions										
×16	8 level transitions										
×32	16 level transitions										
		FM	Normal mode	<table border="1"> <tr> <td>×8</td> <td>4 level transitions</td> </tr> <tr> <td>×16</td> <td>8 level transitions</td> </tr> <tr> <td>×32</td> <td>16 level transitions</td> </tr> </table>	×8	4 level transitions	×16	8 level transitions	×32	16 level transitions	
×8	4 level transitions										
×16	8 level transitions										
×32	16 level transitions										
			Search mode*	1 level transition	Sampling ratio must also be set						

\* The ADPLL enters search mode when the enter search mode command is issued. For details, see section 4.5.3 "Precautions for Using the ADPLL."

**Table 4-10. ADPLL Specifications (cont.)**

No.	Item	Mode	Specification			Remarks
4	Receive data noise suppression	Noise suppressor operation	On	Undefined	Off	
		Noise suppressor mode	×8	$x < 1/8$	$1/8 \leq x < 2/8$	$2/8 \leq x$
			×16	$x < 2/16$	$2/16 \leq x < 3/16$	$3/16 \leq x$
			×32	$x < 4/32$	$4/32 \leq x < 5/32$	$5/32 \leq x$
5	Receive clock noise suppression	Noise suppressor mode	×8	$x < 1/8$	$1/8 \leq x < 2/8$	$2/8 \leq x$
			×16	$x < 2/16$	$2/16 \leq x < 3/16$	$3/16 \leq x$
			×32	$x < 4/32$	$4/32 \leq x < 5/32$	$5/32 \leq x$
6	Maximum bit rate for receive clock noise suppression	Noise suppressor mode	×8	1.25 Mbps		during receive
			×16	0.62 Mbps		clock noise
			×32	0.31 Mbps		suppression

(x: Noise width/1-bit cell width)



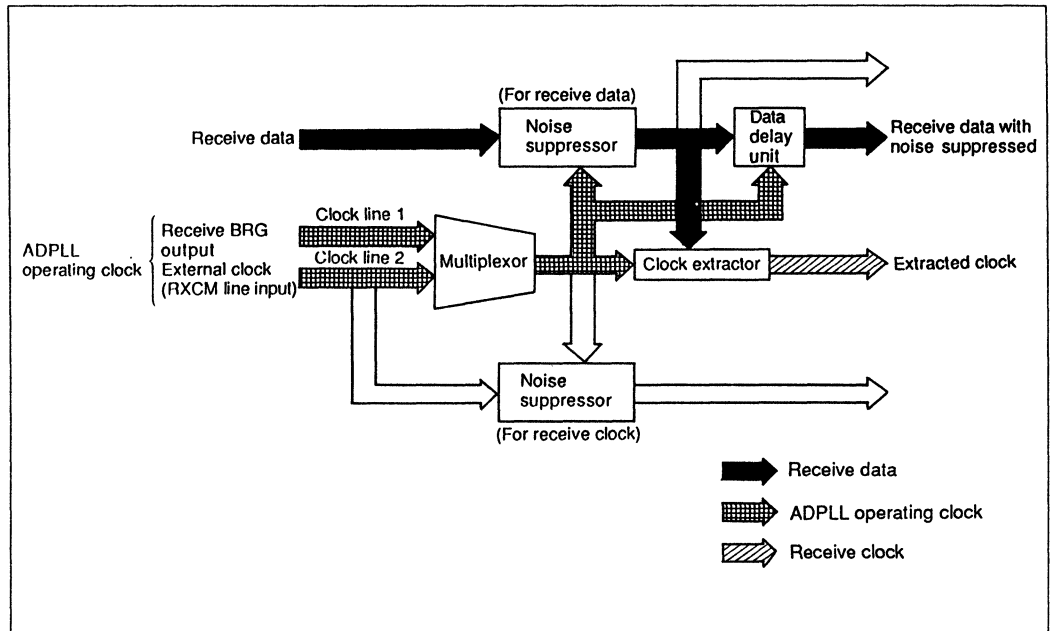
**Figure 4-37. Transmission Codes Supported by the ADPLL and Their Waveforms**

## 4.5.2 Operation

The ADPLL has two main functions: (1) extracting clock components from the noise-suppressed receive data, and (2) suppressing receive clock noise.

### Extracting Clock Components from Receive Data

Figure 4-38 shows the flow of receive data and the ADPLL operating clock signals when clock extraction is performed. Either the receive baud rate generator output from clock line 1 or the external clock (RXCM line input) from clock line 2 can be used as the ADPLL clock.



**Figure 4-38. Data Flow and Clocks When Extracting Clock from Receive Data**

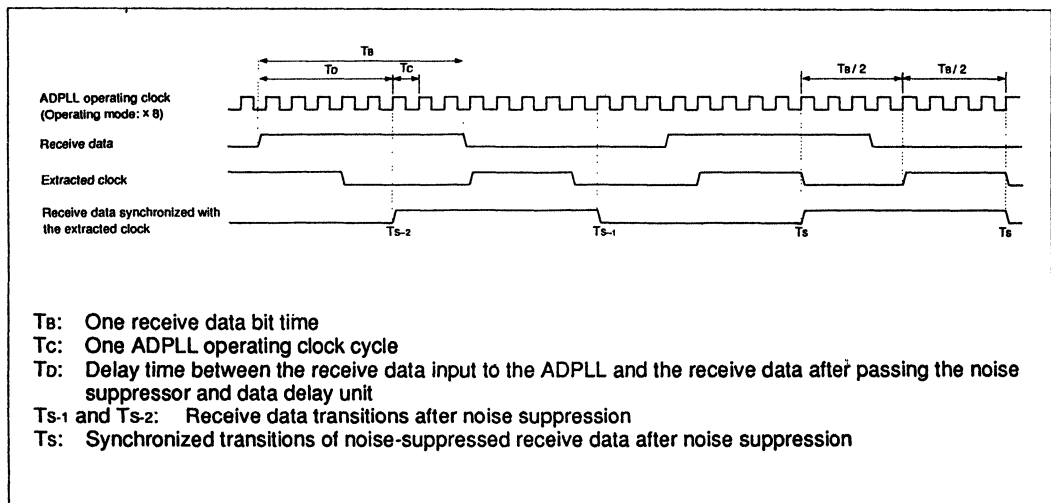
- ① The receive data is sent to the receive data noise suppressor.
- ② The noise-suppressed receive data is output to the clock extractor and data delay unit.
- ③ The noise-suppressed receive data is phase, matched with the extracted clock and output from the data delay unit.
- ④ The clock extractor extracts clock components from the noise-suppressed receive data and outputs the resulting clock.
- ⑤ The ADPLL operating clock (the receive baud rate generator output or external clock) passes through the multiplexor and is supplied to the clock extractor, receive data noise suppressor and data delay unit.

When the noise-suppressed receive data and extracted clock are output, their phases are matched using the ADPLL phase compensation function. Phase compensation timing for the NRZ code and FM0 code receive data is shown in figures 4-39 and 4-40.

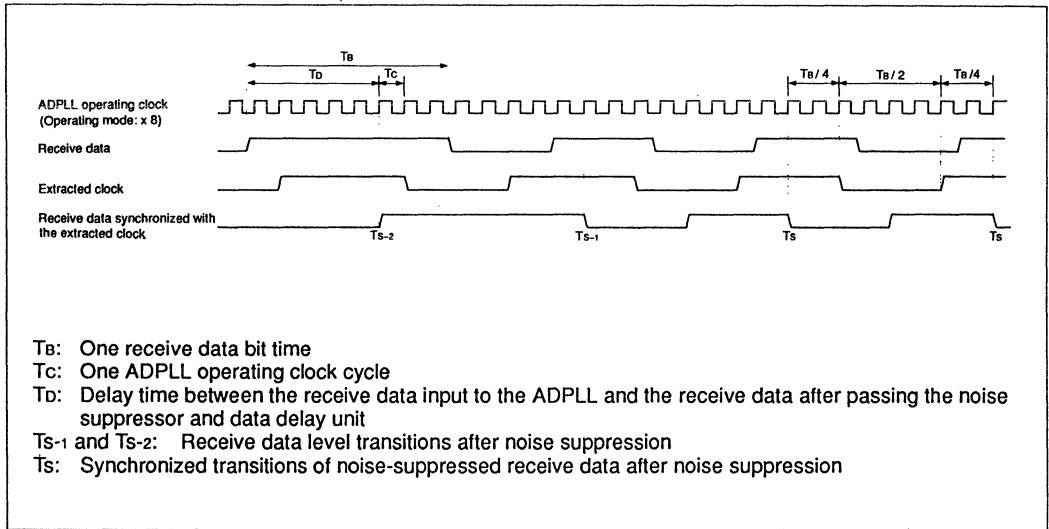
The noise-suppressed receive data input to the receive data noise suppressor is output to the data delay unit and clock extractor. The ADPLL samples the noise-suppressed receive data on the leading edge of the ADPLL operating clock and performs clock extraction.

The phases of the receive data and extracted clock are compared at level change points ( $T_s$ ,  $T_{s-1}$ ,  $T_{s-2}$ ) in the receive data output from the data delay unit. If there is a skew between the two phases, the extracted clock cycle is lengthened or shortened by one ADPLL operating clock cycle. In the examples shown in figures 4-39 and 4-40 (operating mode  $\times 8$ ), this synchronization can be established within a maximum of four change points. (For FM type codes (FM0, FM1 and Manchester), synchronization can be established with one level change by issuing the enter search mode command.)

The relationship between the extracted clock and the receive data bit cell differs depending on the receive data code. For NRZ and NRZI codes, the leading edge of the extracted clock is located at the mid point of the data bit cell width output from the data delay unit. For FM0, FM1, and Manchester codes, the leading edge of the extracted clock is located at the  $1/4$  point of the data bit cell width output from the data delay unit. This applies also to operating modes  $\times 16$  and  $\times 32$  with the difference that the maximum number of level changes required for synchronization is 8 and 16, respectively.



**Figure 4-39. NRZ Receive Data Phase Compensation in Operating Mode  $\times 8$**

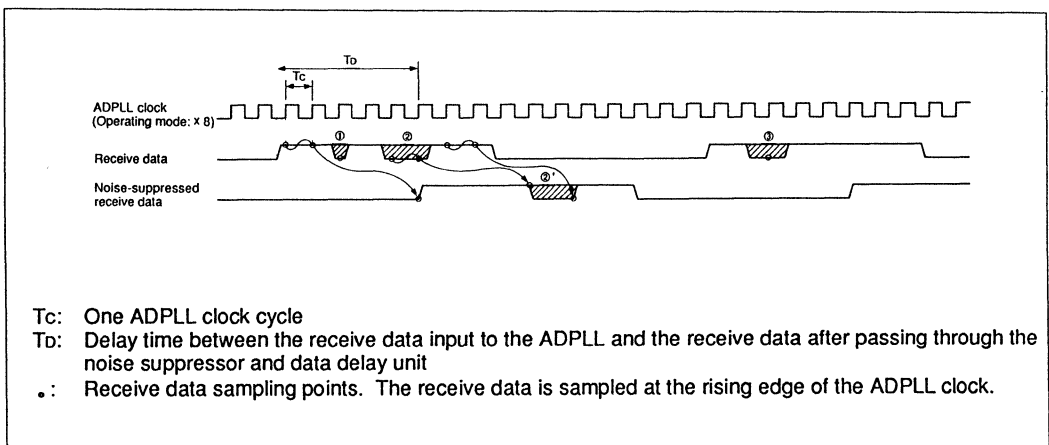


**Figure 4-40. FM0 Receive Data Phase Compensation in Operating Mode  $\times 8$**

The receive data noise suppression timing in the noise suppressor is shown in figure 4-41. NRZ code receive data is used in this example. The same basic timing also applies, however, to other codes.

The receive data is sampled at the rising edge of the ADPLL clock. In operation mode  $\times 8$ , receive data that can be sampled twice in succession is considered valid data. (Data sampled three times in succession in operation mode  $\times 16$  and five times in succession in operation mode  $\times 32$  is considered valid.) All other sampled data is suppressed as noise.

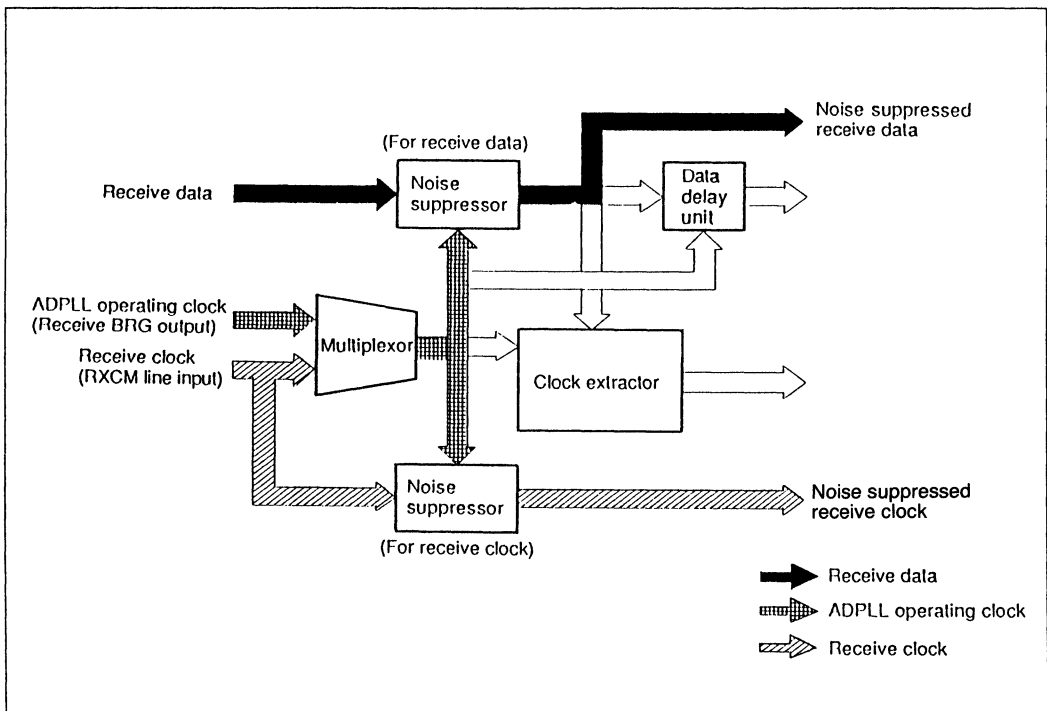
①, ②, and ③ in the figure correspond to "ON", "OFF" and "Undefined" in No. 4 of table 4-10 "ADPLL Specifications." The noise shown in ③ cannot be sampled twice in succession and is suppressed as noise.



**Figure 4-41. Noise Suppression in the Receive Data Noise Suppressor in Operating Mode  $\times 8$**

## Suppressing Receive Clock Noise

Figure 4-42 shows the flow of receive data, the ADPLL operating clock and receive clock signals when noise suppression is performed.



**Figure 4-42. Data and Clock Signal Flow When Suppressing Receive Clock Noise**

To suppress signal noise, the ADPLL operates as follows:

- ① The receive data is sent to the receive data noise suppressor for noise suppression.
- ② The ADPLL operating clock is supplied to the receive data noise suppressor and to the receive clock noise suppressor via the multiplexor.
- ③ The receive clock is input to the receive clock noise suppressor.

Noise suppression timing in the receive clock noise suppressor is shown in figure 4-43. In this example, operation mode  $\times 8$  is used. The same basic timing applies to other modes with the only difference being the number of consecutive sampling times. The receive clock is sampled at the rising edge of the ADPLL clock. In operation mode  $\times 8$ , receive data sampled twice in succession is considered valid. (Data is also considered valid when sampled three times in succession in operation mode  $\times 16$  and five times in succession in operation mode  $\times 32$ .) All other sampled data is suppressed as noise.\*

⓪ and Ⓜ in the figure correspond to "ON" and "OFF" in No. 5 of table 4-10 "ADPLL Specifications."

Receive data noise suppression is performed as described in "Extracting Clock Component from the Receive Data" above.

\* If noise occurs around the leading or falling edges of the receive clock, the leading or falling edges of the noise-suppressed receive clock may be shifted forward or back. The maximum shift widths in  $\times 8$ ,  $\times 16$  and  $\times 32$  modes are 2, 3, and 5 ADPLL operating clock cycles, respectively.

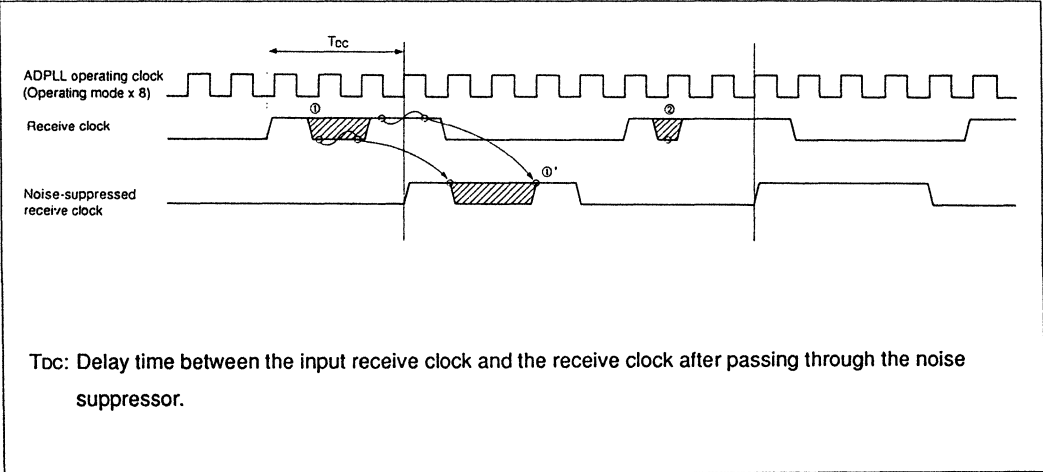


Figure 4-43. Noise Suppression in the Receive Clock Noise Suppressor

### 4.5.3 Precautions for Using the ADPLL

By issuing an enter search mode command, FM-coded receive data can be synchronized after only one transition. This command is effective in all operating modes ( $\times 8$ ,  $\times 16$ , or  $\times 32$ ).

When issuing an enter search mode command, the following synchronize patterns are recommended for each coding scheme:

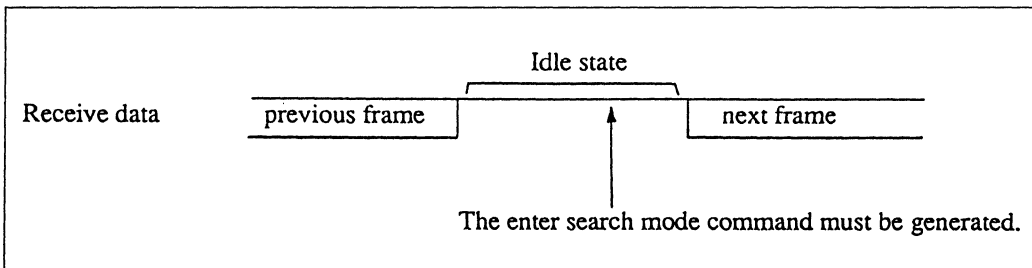
- FM0.....11111111
- FM1.....00000000
- Manchester.....10101010

The ADPLL needs the following precaution.

### 4.5.4 Precaution

When the MSCI receives continuous frames of the FM type code (FM0, FM1 or Manchester) serial data in the byte or bit synchronous mode, if the continuous frames are out of phase to each other, the enter search mode command must be generated between these frames to synchronize during the idle state.

The timing chart is shown below.



**Figure 4-44. The Enter Search Mode Command Timing between Continuous Frames**



### 4.5.5 Detailed Description

The table shown below is the maximum of level transitions necessary for synchronization of the receive data.

**Table 4-11. Maximum of Level Transitions**

Code type	Operating mode		Specification
NRZ		×8	4 level transitions
		×16	8 level transitions
		×32	16 level transitions
FM	Normal mode	×8	4 level transitions
		×16	8 level transitions
		×32	16 level transitions
	Search mode		1 level transition

Note the following restriction concerned with the table 4-11.

#### Restriction

The specification of the normal mode in FM type code on the table 4-11. is effective only when the receive data is synchronized by the enter search mode command.

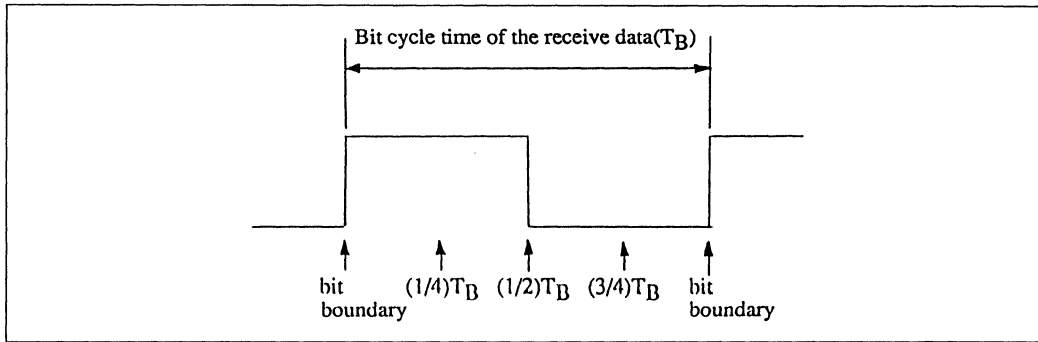
At FM type code, the phase of the receive data is adjusted by the level transitions in the specific window of the bit cell. The window defines the phase adjustment timing (the level transition at start or center of the bit cell).

Therefore, the restriction above is needed.

The table shown below is the phase adjustment of the ADPLL.

**Table 4-12. The Phase Adjustment of the ADPLL**

Code type	Level transition of the receive data	Phase adjustment
NRZ, NRZI	From bit boundary to $(1/2)T_B$	Delay phase by one ADPLL operating clock
	From $(1/2)T_B$ to bit boundary	Advance phase by one ADPLL operating clock
FM0, FM1	From bit boundary to $(1/4)T_B$	Delay phase by one ADPLL operating clock
	From $(3/4)T_B$ to bit boundary	Advance phase by one ADPLL operating clock
	Other than above	No phase adjustment
Manchester	From $(1/2)T_B$ to $(3/4)T_B$	Delay phase by one ADPLL operating clock
	From $(1/4)T_B$ to $(1/2)T_B$	Advance phase by one ADPLL operating clock
	Other than above	No phase adjustment



#### 4.5.6 Examples of Generating the Enter Search Mode Command

Examples of generating the enter search mode command between continuous frames are as follows.

##### Example 1.

Add an external circuit for detecting 'Logical High' state of the line during the idle state. This circuit notifies the logical high state to HD64180S by the HD64180S external interrupt. HD64180S generates the enter search mode command during the interrupt processing routine.

##### Example 2.

Add an external circuit for detecting the carrier of the next frame. This circuit notifies the carrier detection to HD64180S by the HD64180S external interrupt. HD64180S generates the enter search mode command during the interrupt processing routine.

### 4.6 Baud Rate Generator

#### 4.6.1 Overview

The MSCI has an internal baud rate generator (BRG) which is used to generate the MSCI transmit/receive clock. The BRG has the following main features:

- Output clock frequency range:  $f\phi$  to  $f\phi/2^{17}$  ( $2^{17} = 131,072$ )  
( $f\phi$ : CPU clock frequency)\*1
- Frequency accuracy within  $\pm 0.5\%$  for any frequency range from  $f\phi/100$  to  $f\phi/2^{17*2}$ .

\*1 When  $f_{BRG} = f\phi$ , the BRG output cannot be obtained from the TXDM and RXDM lines.

\*2 If  $-f_{BRG} \leq \frac{50}{\text{Time constant register value}} (\%)$

Where,  $f$  is the target frequency and  $f_{BRG}$  is the BRG output frequency set to the closest  $f$  value. ( $f \geq f \geq f/2^{17}$ )

Independent transmit and receive frequencies can be specified as the  $n$ th power of 2 (where  $n$  is a positive integer).

Figure 4-45 shows the block diagram of the baud rate generator.

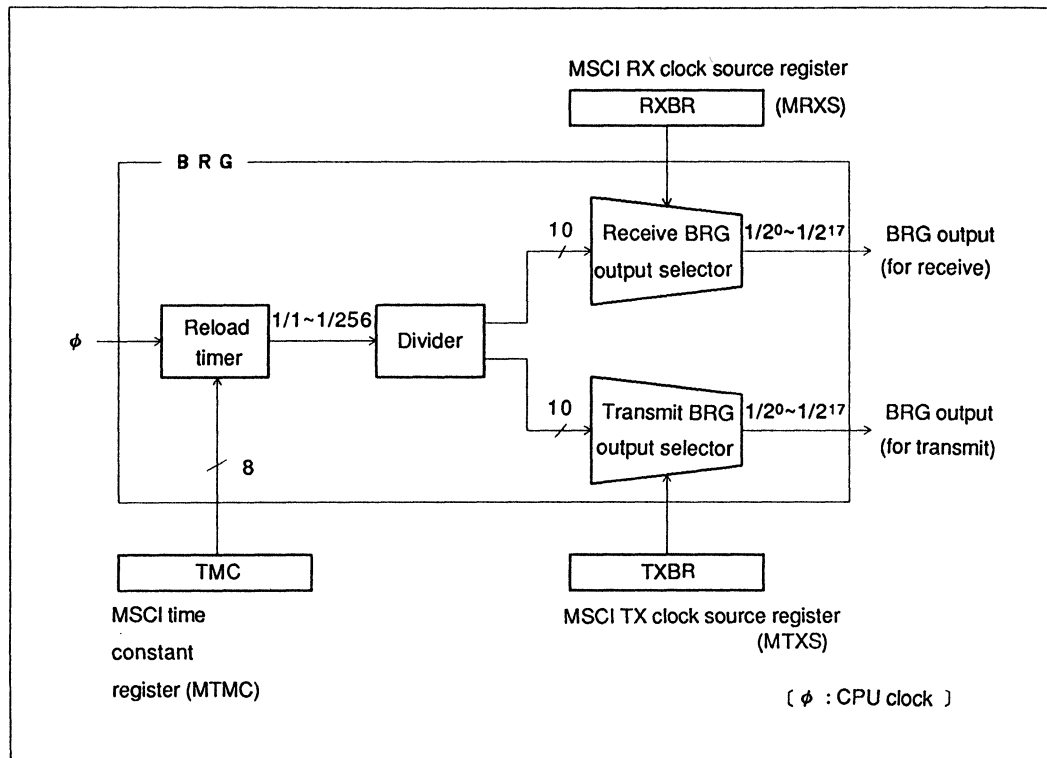
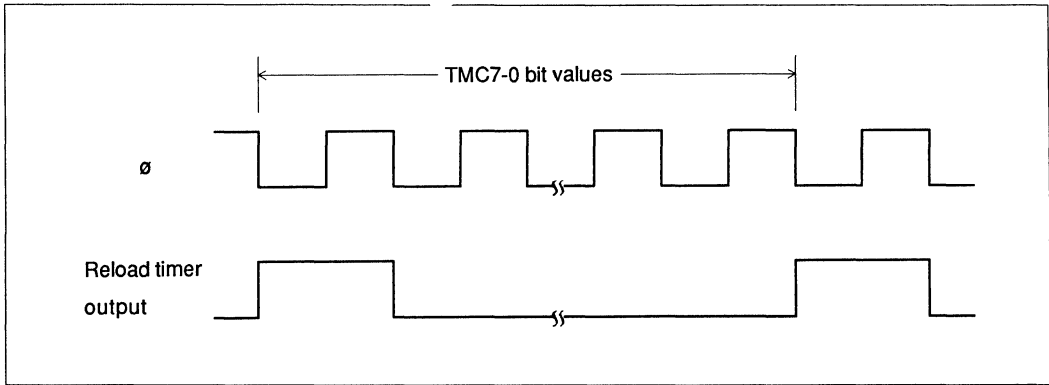


Figure 4-45. Block Diagram of the Baud Rate Generator

## 4.6.2 Functions

The MSCI baud rate generator clock is selected using the TMC7-0 bits in the MSCI time constant register (MTMC), the TXBR3-0 bits of the MSCI TX clock source register (MTXS), and the RXBR3-0 bits of the RX clock source register (MRXS).

MTMC is an 8-bit register for specifying the value to be loaded into the reload timer within the baud rate generator. The reload timer is decremented based on the CPU clock  $\phi$ , and the timer outputs a high level signal for one clock cycle each time the reload timer value equals 1. Thus, a high level signal is output once each time the number of CPU clock cycles specified in the TMC7-0 bits of MTMC elapses, as shown in figure 4-46. When 0 is specified, 256 is assumed, and when 1 is specified, this output will be the same as the CPU clock frequency.



**Figure 4-46. Reload Timer Output**

The reload timer output is input to a frequency divider. The transmit frequency division ratio is specified by the TXBR3-0 bits in the MTXS register and the receive frequency division ratio by the RXBR3-0 bits in the MRXS register.

In addition, the TXCS2-0 bits of the MTXS register and the RXCS2-0 bits of the MRXS register are used to specify whether or not to enable the output clock for the MSCI transmitter and receiver, respectively. The BRG output can be used for the transmit/receive clock or to the ADPLL operating clock. For details about these specifications, see sections 4.2.4 "MSCI Control Register," 4.2.5 "MSCI RX Clock Source Register," and 4.2.6 "MSCI TX Clock Source Register."

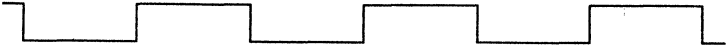


The relationship between the register set value and the generated clock frequency is given below.

$$f_{BRG} = \frac{f_{\phi}}{TMC} + 2BR$$

- f<sub>BRG</sub>: Frequency of the transmit (or receive) BRG output
- f<sub>φ</sub>: CPU clock frequency (Frequency f<sub>φ</sub> = f<sub>BRG</sub> can be used only as the ADPLL operating clock)
- TMC: Time constant register set value (1-256)
- BR: Value (0-9) of TXBR3-0 bits in the TX clock source register or RXBR3-0 bits in the RX clock source register

Table 4-13 gives clock widths and duty cycles (pulse width to pulse frequency) for BRG output clock waveforms along with the corresponding register set values.

**Table 4-13. BRG Output Waveform and Register Set Values**

BR Set Value	Waveform	
BR = 1 – 9	Duty cycle = 50%	
		
BR = 0 and TMC ≠ 1	Duty cycle = 50% when TMC = 2 Duty cycle ≠ 50% when TMC ≠ 2	Pulse width is 1 CPU clock cycle. 1 CPU clock cycle
		
BR = 0 and TMC = 1	Duty cycle = 50% Pulse width is 1 CPU clock cycle	1 CPU clock cycle
		

BR: Value of bits 3-0 in the TX (RX) clock source register

TMC: Value of bits 7-0 in the time constant register

#### 4.6.3 Register Set Values and Bit Rates

- Asynchronous mode

In asynchronous mode, the bit rate is selected using TMC7-0 bits in the MSCI time constant register (MTMC), the TXBR3-0 bits of the MSCI TX clock source register (MTXS), the RXBR3-0 bits of the MSCI RX clock source register (MRXS), and the BRATE7-6 bits of MSCI mode register 1 (MMD1).

Table 4-14 shows the relationship between the register set values and bit rates.

**Table 4-14. Register Values and Bit Rates in Asynchronous Mode**

Bit Rate (bps)	f $\phi$ (MHz)							
	1.7898				2.4576			
	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	–	–	–	–	1	1	1/32	0.00
19200	–	–	–	–	1	1	1/64	0.00
9600	–	–	–	–	1	2	1/64	0.00
4800	–	–	–	–	1	3	1/64	0.00
2400	47	0	1/16	–0.83	1	4	1/64	0.00
1200	93	0	1/16	–0.25	1	5	1/64	0.00
600	93	0	1/32	–0.25	1	6	1/64	0.00
300	93	0	1/64	–0.25	1	7	1/64	0.00
150	93	1	1/64	–0.25	1	8	1/64	0.00
110	127	1	1/64	0.10	175	1	1/64	–0.25

Bit Rate (bps)	f $\phi$ (MHz)							
	3.072				4			
	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	5	0	1/16	0.00	–	–	–	–
19200	5	0	1/32	0.00	13	0	1/16	0.16
9600	5	0	1/64	0.00	13	0	1/32	0.16
4800	5	1	1/64	0.00	13	0	1/64	0.16
2400	5	2	1/64	0.00	13	1	1/64	0.16
1200	5	3	1/64	0.00	13	2	1/64	0.16
600	5	4	1/64	0.00	13	3	1/64	0.16
300	5	5	1/64	0.00	13	4	1/64	0.16
150	5	6	1/64	0.00	13	5	1/64	0.16
110	109	2	1/64	0.08	71	3	1/64	0.03

TMC: Value of the TMC7-0 bits in the MSCI time constant register (in decimal)

BR: Value of the TXBR3-0 bits in the MSCI TX clock source register or value of the RXBR3-0 bits in the MSCI RX clock source register

CM: Clock mode in asynchronous mode (bit rate/clock frequency)

**Table 4-14. Register Values and Bit Rates in Asynchronous Mode (cont.)**

Bit Rate (bps)	f $\phi$ (MHz)							
	4.608				4.9152			
	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	–	–	–	–	1	1	1/64	0.00
19200	15	0	1/16	0.00	1	2	1/64	0.00
9600	15	0	1/32	0.00	1	3	1/64	0.00
4800	15	0	1/64	0.00	1	4	1/64	0.00
2400	15	1	1/64	0.00	1	5	1/64	0.00
1200	15	2	1/64	0.00	1	6	1/64	0.00
600	15	3	1/64	0.00	1	7	1/64	0.00
300	15	4	1/64	0.00	1	8	1/64	0.00
150	15	5	1/64	0.00	1	9	1/64	0.00
110	41	4	1/64	–0.22	175	2	1/64	–0.25

Bit Rate (bps)	f $\phi$ (MHz)							
	6				6.144			
	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	–	–	–	–	5	0	1/32	0.00
19200	–	–	–	–	5	0	1/64	0.00
9600	39	0	1/16	0.16	5	1	1/64	0.00
4800	39	0	1/32	0.16	5	2	1/64	0.00
2400	39	0	1/64	0.16	5	3	1/64	0.00
1200	39	1	1/64	0.16	5	4	1/64	0.00
600	39	2	1/64	0.16	5	5	1/64	0.00
300	39	3	1/64	0.16	5	6	1/64	0.00
150	39	4	1/64	0.16	5	7	1/64	0.00
110	213	2	1/64	0.03	109	3	1/64	0.08

TMC: Value of the TMC7-0 bits in the MSCI time constant register (in decimal)

BR: Value of the TXBR3-0 bits in the MSCI TX clock source register or value of the RXBR3-0 bits in the MSCI RX clock source register

CM: Clock mode in asynchronous mode (bit rate/clock frequency)

**Table 4-14. Register Values and Bit Rates in Asynchronous Mode (cont.)**

Bit Rate (bps)	f $\phi$ (MHz)							
	8				9.216			
	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	13	0	1/16	0.16	15	0	1/16	0.00
19200	13	0	1/32	0.16	15	0	1/32	0.00
9600	13	0	1/64	0.16	15	0	1/64	0.00
4800	13	1	1/64	0.16	15	1	1/64	0.00
2400	13	2	1/64	0.16	15	2	1/64	0.00
1200	13	3	1/64	0.16	15	3	1/64	0.00
600	13	4	1/64	0.16	15	4	1/64	0.00
300	13	5	1/64	0.16	15	5	1/64	0.00
150	13	6	1/64	0.16	15	6	1/64	0.00
110	71	4	1/64	0.03	41	5	1/64	-0.22

Bit Rate (bps)	f $\phi$ (MHz)							
	9.8304				10			
	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	2	1	1/64	0.00	-	-	-	
19200	2	2	1/64	0.00	-	-	-	
9600	2	3	1/64	0.00	65	0	1/16	0.16
4800	2	4	1/64	0.00	65	0	1/32	0.16
2400	2	5	1/64	0.00	65	0	1/64	0.16
1200	2	6	1/64	0.00	65	1	1/64	0.16
600	2	7	1/64	0.00	65	2	1/64	0.16
300	2	8	1/64	0.00	65	3	1/64	0.16
150	2	9	1/64	0.00	65	4	1/64	0.16
110	175	3	1/64	-0.25	89	4	1/64	-0.25

TMC: Value of the TMC7-0 bits in the MSCI time constant register (in decimal)

BR: Value of the TXBR3-0 bits in the MSCI TX clock source register or value of the RXBR3-0 bits in the MSCI RX clock source register

CM: Clock mode in asynchronous mode (bit rate/clock frequency)



**Table 4-14. Register Values and Bit Rates in Asynchronous Mode (cont.)**

Bit Rate (bps)	f $\phi$ (MHz)			
	12*			
	TMC	BR	CM	Deviation (%)
38400	–	–	–	–
19200	39	0	1/16	0.16
9600	39	0	1/32	0.16
4800	39	0	1/64	0.16
2400	39	1	1/64	0.16
1200	39	2	1/64	0.16
600	39	3	1/64	0.16
300	39	4	1/64	0.16
150	39	5	1/64	0.16
110	213	3	1/64	0.03

TMC: Value of the TMC7-0 bits in the MSCI time constant register (in decimal)

BR: Value of the TXBR3-0 bits in the MSCI TX clock source register or value of the RXBR3-0 bits in the MSCI RX clock source register

CM: Clock mode in asynchronous mode (bit rate/clock frequency)

\* The values for f $\phi$  = 12 MHz are given for reference purposes.

- Byte/Bit synchronous mode

In byte or bit synchronous mode, the bit rate is selected using the TMC7-0 bits of the MSCI time constant register (MTMC), the TXBR3-0 bits of the MSCI TX clock source register (MTXS), and the RXBR3-0 bits of the MSCI RX clock source register (MRXS).

Table 4-15 shows the register set values and the corresponding bit rates.

**Table 4-15. Register Set Values and Bit Rates (Byte/Bit Synchronous Mode)**

Bit Rate (bps)	f <sub>0</sub> (MHz)								
	2.4576			3.072			4		
	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)
38400	32	1	0.00	40	1	0.00	52	1	0.16
19200	32	2	0.00	40	2	0.00	52	2	0.16
9600	32	3	0.00	40	3	0.00	52	3	0.16
4800	32	4	0.00	40	4	0.00	52	4	0.16
2400	32	5	0.00	40	5	0.00	52	5	0.16
1200	32	6	0.00	40	6	0.00	52	6	0.16
600	32	7	0.00	40	7	0.00	52	7	0.16
300	32	8	0.00	40	8	0.00	52	8	0.16

Bit Rate (bps)	f <sub>0</sub> (MHz)								
	4.608			4.9152			6		
	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)
38400	60	1	0.00	64	1	0.00	78	1	0.16
19200	60	2	0.00	64	2	0.00	78	2	0.16
9600	60	3	0.00	64	3	0.00	78	3	0.16
4800	60	4	0.00	64	4	0.00	78	4	0.16
2400	60	5	0.00	64	5	0.00	78	5	0.16
1200	60	6	0.00	64	6	0.00	78	6	0.16
600	60	7	0.00	64	7	0.00	78	7	0.16
300	60	8	0.00	64	8	0.00	78	8	0.16

**TMC:** Value of the TMC7-0 bits in the MSCI time constant register (in decimal)

**BR:** Value of the TXBR3-0 bits in the MSCI TX clock source register or value of the RXBR3-0 bits in the MSCI RX clock source register

**Table 4-15. Register Set Values and Bit Rates (Byte/Bit Synchronous Mode) (cont.)**

Bit Rate (bps)	$f\phi$ (MHz)								
	6.144			8			9.216		
	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)
38400	80	1	0.00	104	1	0.16	120	1	0
19200	80	2	0.00	104	2	0.16	120	2	0
9600	80	3	0.00	104	3	0.16	120	3	0
4800	80	4	0.00	104	4	0.16	120	4	0
2400	80	5	0.00	104	5	0.16	120	5	0
1200	80	6	0.00	104	6	0.16	120	6	0
600	80	7	0.00	104	7	0.16	120	7	0
300	80	8	0.00	104	8	0.16	120	8	0

Bit Rate (bps)	$f\phi$ (MHz)								
	9.8304			10			12*		
	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)
38400	128	1	0	130	1	0.16	156	1	0.16
19200	128	2	0	130	2	0.16	156	2	0.16
9600	128	3	0	130	3	0.16	156	3	0.16
4800	128	4	0	130	4	0.16	156	4	0.16
2400	128	5	0	130	5	0.16	156	5	0.16
1200	128	6	0	130	6	0.16	156	6	0.16
600	128	7	0	130	7	0.16	156	7	0.16
300	128	8	0	130	8	0.16	156	8	0.16

TMC: Value of the TMC7-0 bits in the MSCI time constant register (in decimal)

BR: Value of the TXBR3-0 bits in the MSCI TX clock source register or value of the RXBR3-0 bits in the MSCI RX clock source register

\* The values for  $f\phi = 12$  MHz are given for reference purposes.

## 4.7 Internal Interrupts

### 4.7.1 Interrupt Types and Sources

The MSCI can issue four types of interrupt requests: TXRDY, RXRDY, TXINT, and RXINT.

These interrupts are initiated by the status bits (bits 7, 6, 1, and 0) in MST0 and are enabled/disabled by the enable bits (bits 7, 6, 1, and 0) in MIE0.

The TXINT and RXINT internal interrupts are also assigned status bits and corresponding enable bits for each source. The status bit and its enable bit are ANDed for each interrupt source. The interrupt sources are indicated by the TXINT bit (bit 7) or RXINT bit (bit 6) in MST0 regardless of the TXINTE bit (bit 7) or RXINTE bit (bit 6) in MIE0.

### 4.7.2 Interrupt Clear

The methods for clearing each interrupt are given below.

#### (1) TXRDY interrupt

Write data to the transmit buffer until it becomes full, or disable the transmitter. This interrupt can also be cleared by a channel or TX reset command.

#### (2) RXRDY interrupt

Read data from the receive buffer until it becomes empty. This interrupt can also be cleared by a channel or RX reset command.

#### (3) TXINT interrupt

If 1 is written in each status bits, the interrupt is cleared. When the interrupt source is idle transmitter, TXINT can be cleared by writing transmit data to the transmit buffer.

#### (4) RXINT interrupt

If 1 is written in each status bits, the interrupt is cleared.

When the interrupt source is a parity/MP or CRC error, the status bits can be reset by reading the receive data. In bit synchronous mode, when the last character to be transferred has been read from the receive buffer at completion of frame transfer, the MSCI status register 2 (MST2) bit values are transferred to the MSCI frame status register (MFST), and MST2 is reset. Table 4-16 shows interrupt types, sources and methods of clearing.

**Table 4-16. Internal Interrupts, Their Sources and Clearing Methods**

Interrupt Type	Interrupt		Interrupt Source	Source Status Bit	Enable Bit	Clear Procedure*1
	Status Bit	Enable Bit				
TXRDY interrupt	TXRDY	TXRDYE	Transmit ready	–	–	Write to transmit buffer until it becomes full; or, disable transmitter.
RXRDY interrupt	RXRDY	RXRDYE	Receive ready	–	–	Read data from receive buffer until it becomes empty.
TXINT interrupt	TXINT	TXINTE	① Underrun error	UDRN	UDRNE	① ③ Write 1 to status bits. ② Write transmit data to exit other state.
			② Transmitter idle	IDL	IDLE	
			③ CTSM line level change	CCTS	CCTSE	
RXINT interrupt	RXINT	RXINTE	① SYN pattern detection/flag detection	SYNCD/ FLGD	SYNCDE/ FLGDE	Set the status bit to 1.  PMP: Read the receive data thus making next data available to be read.*2  CRCE: Automatically cleared when the CRC calculation result is normal.*3
			② DCDM line level change	CDCD	CDCDE	
			③ Break start detection/abort detection/GA pattern detection	BRKD/ ABTD/ GAPD	BRKDE/ ABTDE/ GAPDE	
			④ Break stop detection/idle start detection	BRKE/ IDLD	BRKEE/ IDLDE	
			⑤ Receive frame end (MST2)	EOM*4	EOME	
			⑥ Parity or MP bit = 1/short frame detection	PMP/ SHRT*4	PMPE/ SHRTE	

**Table 4-16. Internal Interrupts, Their Sources (cont.)**

Interrupt						
Interrupt Type	Status Bit	Enable Bit	Interrupt Source	Source Status Bit	Enable Bit	Clear Procedure*1
RXINT interrupt	RXINT	RXINTE	⑦ Parity error/abort end frame detection	PE/ABT*4	PEE/ABTE	Set the status bit to 1.
			⑧ Framing error detection/residue bit frame detection	FRME/RBIT*4	FRMEE/RBITE	PMP: Read the receive data thus making next data available to be read.*2
			⑨ Overrun error	OVRN*4	OVRNE	
			⑩ CRC error	CRCE*4	CRCEE	
			⑪ Receive frame end (MFST)	EOMF	EOMFE	CRCE: Automatically cleared when the CRC calculation result is normal.*3

\*1 The RXINT interrupt source can also be cleared by issuing a channel or RX reset command. The TXRDY and TXINT interrupt sources can also be cleared by issuing a channel reset or TX reset command.

\*2 While the parity/MP bit of the next data is 0, this is cleared when the RXRDY bit is set to 1 after a read (when the next data is available to be read).

\*3 CRC calculation results can be read from CRCE bit when the CRCCC bit of MSCI mode register 0 is 1. For details of CRCE bit timing, see "Error Check" in section 4.3.2 "Byte Synchronous Mode" and "Error Check" in section 4.3.3 "Bit Synchronous Mode."

\*4 When the last character has been read from the receive buffer at completion of receive frame transfer, the MSCI status register 2 (MST2) bit values are transferred to the MSCI frame status register (MFST) and MST2 is reset.

### 4.7.3 Interrupt Enable Conditions

The conditions for the TXRDY, RXRDY, TXINT, and RXINT interrupt requests are listed below.

- (1) TXRDY interrupt request condition

$$\text{TXRDY} : \text{TXRDY} \cdot \text{TXRDYE}$$

- (2) RXRDY interrupt request condition

$$\text{RXRDY} : \text{RXRDY} \cdot \text{RXRDYE}$$

- (3) TXINT interrupt request condition

$$\text{TXINT} : \text{TXINT} \cdot \text{TXINTE}$$

$$\text{where TXINT} = \text{UDRN} \cdot \text{UDRNE} + \text{IDL} \cdot \text{IDLE} + \text{CCTS} \cdot \text{CCTSE}$$

- (4) RXINT interrupt request condition

$$\text{RXINT} : \text{RXINT} \cdot \text{RXINTE}$$

$$\begin{aligned} \text{where RXINT} = & (\text{SYNCD}/\text{FLGD}) \cdot (\text{SYNCDE}/\text{FLGDE}) + \text{CDCD} \cdot \text{CDCDE} \\ & + (\text{BRKD}/\text{ABTD}/\text{GAPD}) \cdot (\text{BRKDE}/\text{ABTDE}/\text{GAPDE}) \\ & + (\text{BRKE}/\text{IDLD}) \cdot (\text{BRKEE}/\text{IDLDE}) + \text{EOM} \cdot \text{EOME} \\ & + (\text{PMP}/\text{SHRT}) \cdot (\text{PMPE}/\text{SHRTE}) + (\text{PE}/\text{ABT}) \cdot (\text{PEE}/\text{ABTE}) \\ & + (\text{FRME}/\text{RBIT}) \cdot (\text{FRMEE}/\text{RBITE}) + \text{OVRN} \cdot \text{OVRNE} \\ & + \text{CRCE} \cdot \text{CRCEE} + \text{EOMF} \cdot \text{EOMFE} \end{aligned}$$

Figure 4-47 shows the logic flow for interrupt requests, and the status and enable bits of each register.

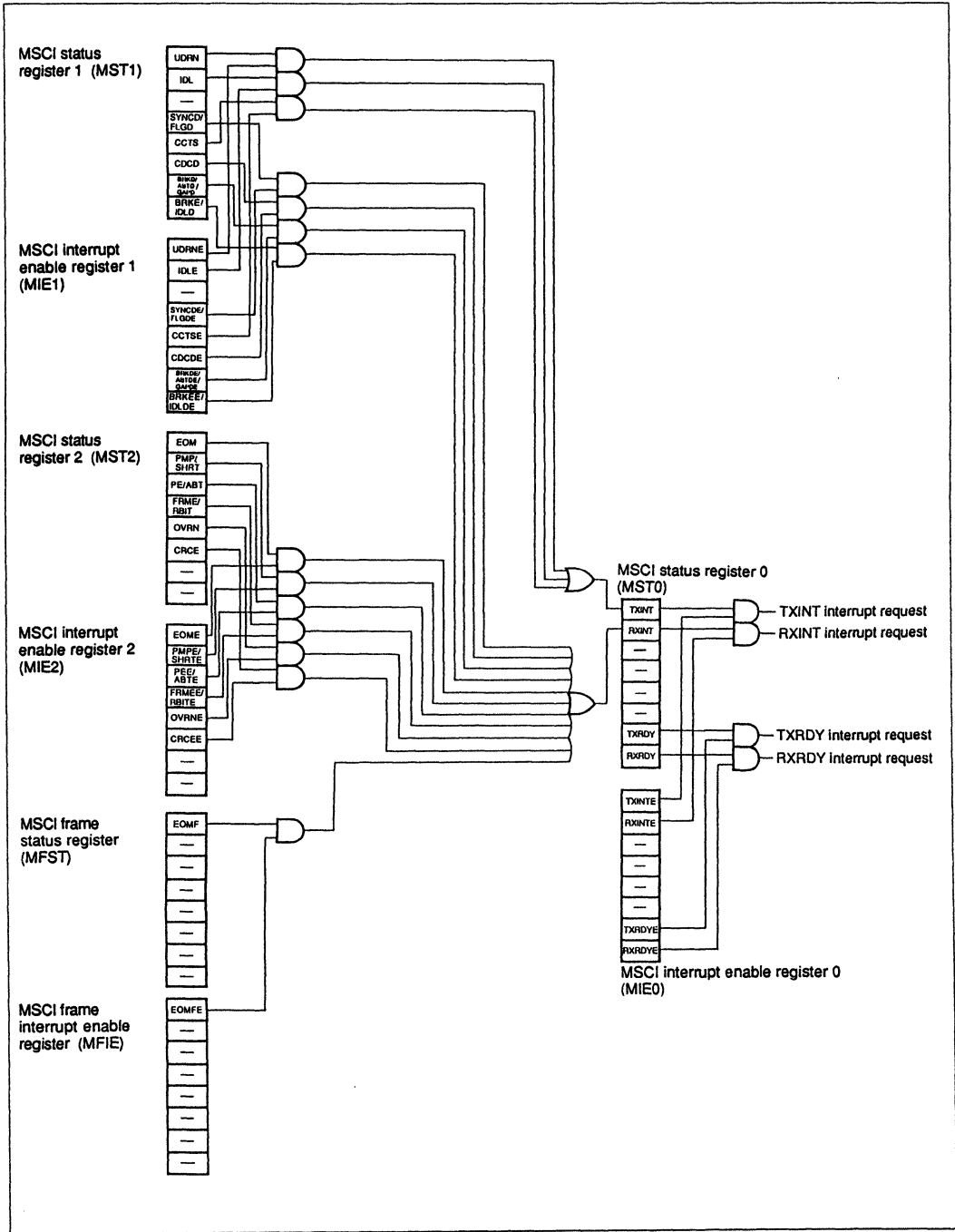


Figure 4-47. Logic Flow for Interrupt Requests, Status and Enable Bits



## 4.8. Application Examples

### 4.8.1 Serial Data Transfer by the CPU and DMAC

#### Transferring Transmit Data

- Polling

The CPU determines the transfer buffer write timing by monitoring the TXRDY bit of MST0. The TXRDY interrupt must be disabled.

- Interrupts

The CPU transfers data to the transmit buffer when a TXRDY interrupt is issued. The TXRDY interrupt is issued by setting the TXRDYE bit to 1 during the TX ready state (specified by the TXRDYC bit in MCTL). At this time, the on-chip DMAC must be disabled for transfer requests.

- DMA transfer

The on-chip DMAC controls data write to the transmit buffer using the DMA transfer request signal. This signal is asserted when the TXRDY bit is set. The TXRDYC bit must be 1 and the TXRDY interrupt must be disabled.

#### Transferring Receive Data

- Polling

The CPU determines the receive buffer data read timing by monitoring the RXRDY bit in MST0. The RXRDY interrupt must be disabled.

- Interrupts

The CPU transfers data to the receive buffer when an RXRDY interrupt is issued. The RXRDY interrupt can be enabled by setting the RXRDYE bit to 1. The on-chip DMAC must be disabled for transfer requests.

- DMA transfer

The on-chip DMAC controls data read from the receive buffer using the DMA transfer request signal. This signal is asserted when the RXRDY bit is set. The RXRDY interrupt must be disabled.

## 4.8.2 Maximum Bit Rate

Table 4-17 lists the maximum bit rates for the MSCI assuming a CPU clock frequency ( $f\phi$ ) = 10 MHz. When these bits rates are exceeded, normal data transfer is not guaranteed.

**Table 4-17. Maximum Bit Rates**

Protocol Mode	Maximum Bit Rate (bps)								
	Clock Mode	External Clock	BRG	Clock Extraction					
				Sampling Clock = External*4			Sampling Clock = BRG*5		
				×8	×16	×32	×8	×16	×32
Asynchronous	1/64	62.5k*1	78.1k*3	–	–	–	–	–	–
	1/32	125k*1	156.3k*3	–	–	–	–	–	–
	1/16	250k*1	312.5k*3	–	–	–	–	–	–
	1/1	4.0M*1 *6	4.0M*1	–	–	–	–	–	–
Byte synchronous	1/1	7.1M*2 *6	5M*3 *6	2.2M	1.1M	0.5M	1.25M	0.62M	0.31M
Bit synchronous (HDLC) mode	1/1	7.1M*2 *6	5M*3 *6	2.2M	1.1M	0.5M	1.25M	0.62M	0.31M
Bit synchronous (loop) mode	1/1	4.0M*1 *6	4.0M*1	2.2M	1.1M	0.5M	1.25M	0.62M	0.31M

( $\phi$  clock ( $f\phi$ ) = 10 MHz)

\*1  $f\phi + 2.5 \times$  (clock mode)

\*2  $f\phi + 1.4 \times$  (clock mode)

\*3  $f\phi + 2 \times$  (clock mode)

\*4 17.6 Mbps + (sampling clock rate)

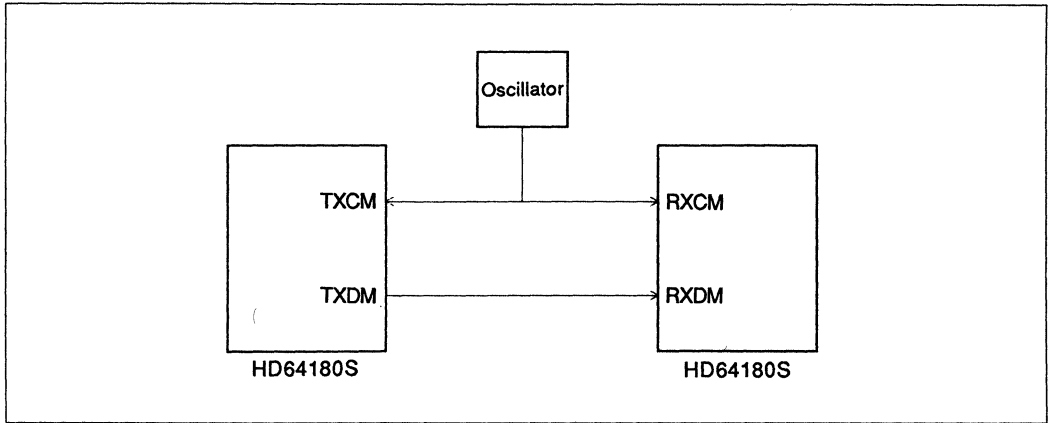
\*5  $f\phi +$  (sampling clock rate)

This is the same as the maximum rate for receive clock noise suppression.

\*6 See "Supplementary Explanation."

### Supplementary Explanation

The above table gives the maximum frequencies available when the transmitter and receiver are operating independently. In the configuration shown in figure 4-48, the maximum bit rates will be lower than those listed in the table.



**Figure 4-48. Transmission/Reception Between HD64180S Chips**

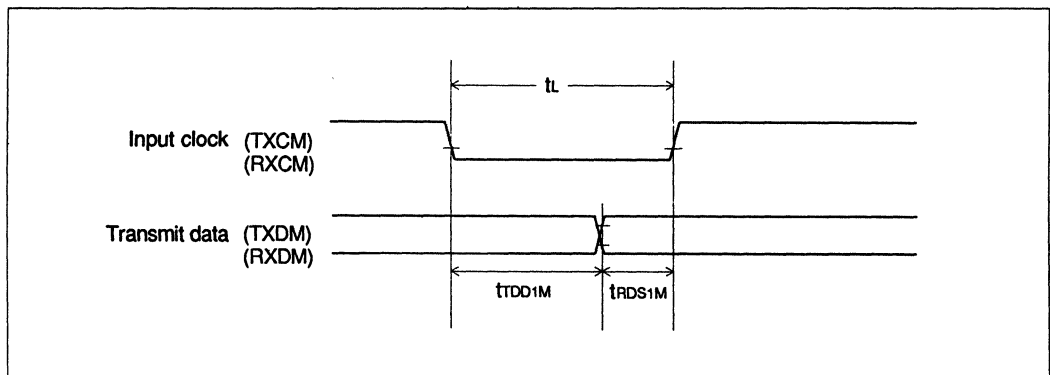
In the 1/1 clock mode, transmit data becomes stable after delay time  $t_{TDD1M}$  from the trailing edge of the input clock.

The receiver samples data at the leading edge of the input clock. The minimum low level time of the input clock is:

$$t_L = t_{TDD1M} + t_{RDS1M}$$

where,  $t_{RDS1M}$  is the receiver set-up time.

In the configuration shown in figure 4-48 the maximum frequency that can be used with this low-level width becomes the maximum bit rate.



**Figure 4-49. Input Clock and Transmit Data**

For example, when  $t_{\text{RDDIM}} = 310 \text{ ns}$  and  $t_{\text{RDSIM}} = 90 \text{ ns}$ , the low level time of the clock is:

$$t_L = 310 \text{ ns} + 90 \text{ ns} = 400 \text{ ns}$$

If the duty ratio of this clock is 50%, the clock frequency is:

$$400 \text{ ns} + 400 \text{ ns} = 800 \text{ ns}$$

Therefore, the maximum bit rate is:

$$\frac{1}{800 \text{ ns}} = 1.25 \text{ Mbps}$$

Accordingly, in the configuration shown in figure 4-48, to operate at the maximum bit rate in 1/1 clock mode, an external circuit must be provided to delay TXDM and to fix the level of TXD at the leading edge of TXC.

### 4.8.3 Example of Transmit by Programmed I/O (Bi-sync Mode)

#### (1) Initialization

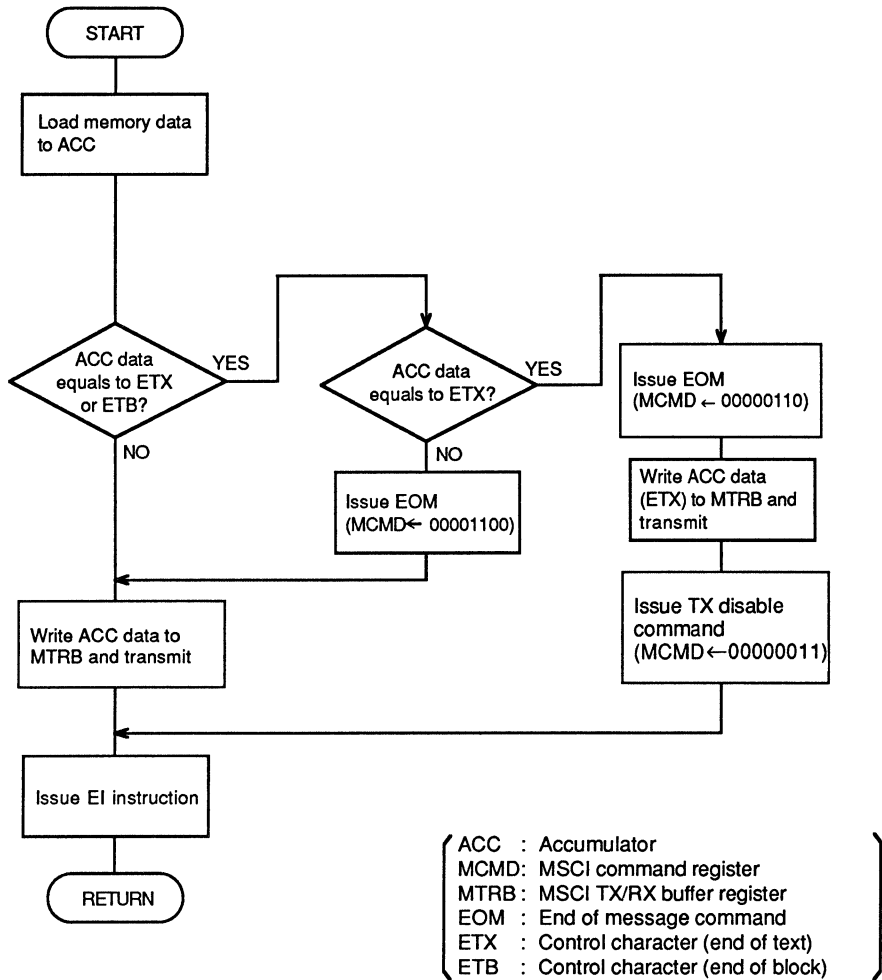
MCMD	←	21H.....	① Channel reset
MMD0	←	44H.....	① Sets bi-sync mode ② Inhibits auto-enable ③ Sets CRC-16 and initializes to all 0s
MMD2	←	00H.....	① Sets NRZ code ② Sets full duplex mode
MCTL	←	11H.....	① TXRDY bit = 1 when transmit buffer is empty ② Specifies idle pattern transmission ③ Sets $\overline{\text{RTSM}}$ line high level output
MTXS	←	00H.....	① Specifies transmit clock TXCM line input
MIE0	←	82H.....	① Enables TXINT interrupts ② Enables TXRDY interrupts
MIE1	←	80H.....	① Enables underrun interrupts
MSA0	←	16H.....	① Sets SYN character

MSA1	←	16H.....	① Sets SYN character
MIDL	←	XXH.....	① Sets leading pad or SYN character
MCMD	←	02H.....	① Enables transmit
MTRB	←	Transmit data	Transmits leading pad, then SYN character, then the transmit data.

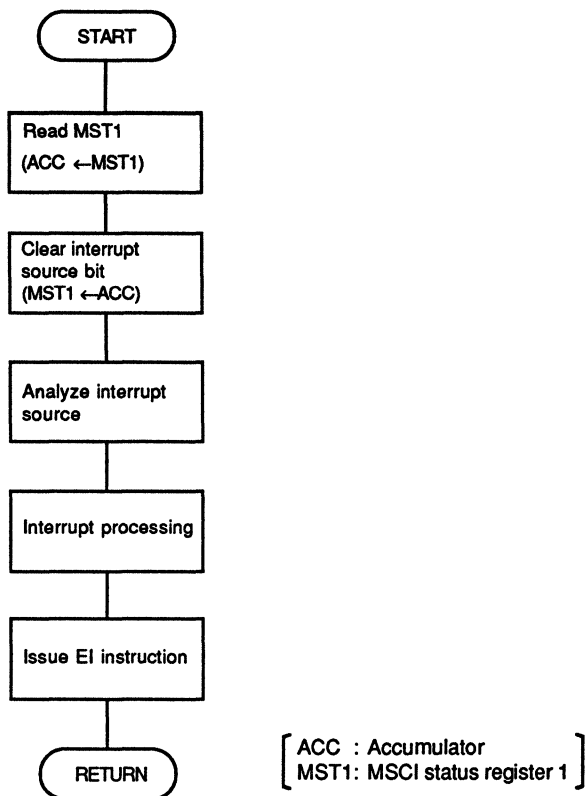
MCMD:	MSCI command register
MMD0:	MSCI mode register 0
MMD2:	MSCI mode register 2
MCTL:	MSCI control register
MTXS:	MSCI TX clock source register
MIE0:	MSCI interrupt enable register 0
MIE1:	MSCI interrupt enable register 1
MSA0:	MSCI sync/address register 0
MSA1:	MSCI sync/address register 1
MIDL:	MSCI idle pattern register

(2) Transmit processing routine

• TXRDY interrupt processing routine



- TXINT interrupt processing routine



#### 4.8.4 Example of Receive by Programmed I/O (Bi-sync Mode)

##### (1) Initialization

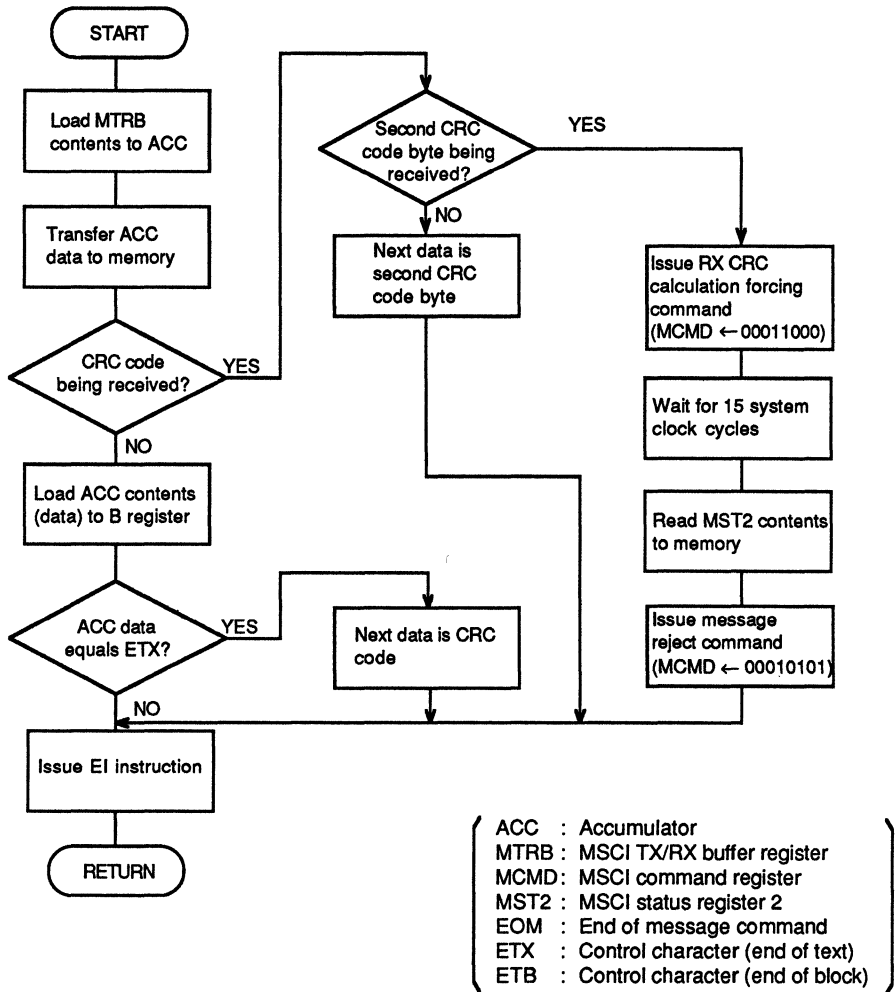
MCMD	←	21H.....	① Channel reset
MMD0	←	44H.....	① Sets bi-sync mode ② Inhibits auto-enable ③ Sets CRC-16 and initialization to all 0s
MMD2	←	00H.....	① Sets NRZ code ② Sets full duplex mode
MCTL	←	05H.....	① Specifies SYN character load
MRXS	←	00H.....	① Specifies receive clock RXCM input line
MIE0	←	41H.....	① Enables RXINT interrupts ② Enables RXRDY interrupts
MIE1	←	10H.....	① Enables SYNCD interrupts
MIE2	←	08H.....	① Enables overrun interrupts
MSA0	←	16H.....	① Sets SYN character
MSA1	←	16H.....	① Sets SYN character
MCMD	←	12H.....	① Enables receive

/	MCMD: MSCI command register	}
	MMD0: MSCI mode register 0	
	MMD2: MSCI mode register 2	
	MCTL: MSCI control register	
	MRXS: MSCI RX clock source register	
	MIE0: MSCI interrupt enable register 0	
	MIE1: MSCI interrupt enable register 1	
	MIE2: MSCI interrupt enable register 2	
	MSA0: MSCI sync/address register 0	
\	MSA1: MSCI sync/address register 1	}

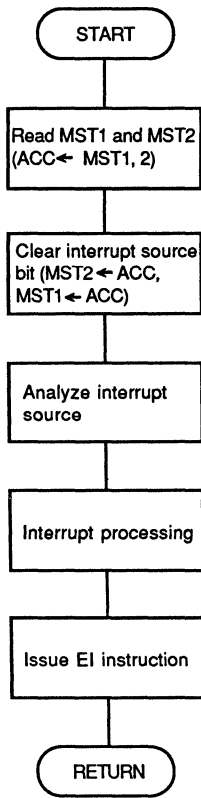


(2) Receive processing routine

- RXRDY interrupt processing routine.



- RXINT interrupt processing routine



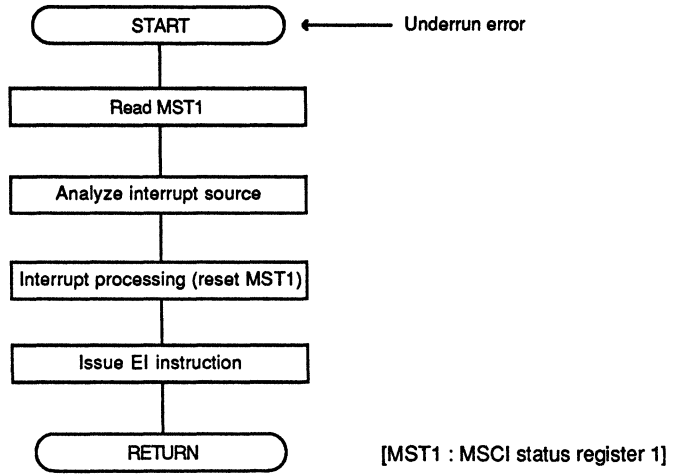
[ ACC : Accumulator  
MST1 : MSCI status register 1  
MST2 : MSCI status register 2 ]

#### 4.8.5 Example of Transmit in DMA Chained Block Transfer Mode (bit synchronous HDLC mode)

- Initialization

MCMD	←	21H.....	① Channel reset
MMD0	←	87H.....	① Sets bit synchronous HDLC mode ② Sets CRC-CCITT, and initializes to all 1s
MMD2	←	00H.....	① Sets NRZ code ② Sets full duplex mode
MCTL	←	91H.....	① TXRDY bit = 1 when transmit buffer is not full ② Specifies transmission of flag and idle ③ Sets $\overline{\text{RTSM}}$ line high level output
MTXS	←	00H.....	① Specifies transmit clock TXCM line input
MIE0	←	80H.....	① Enables TXINT interrupt
MIE1	←	80H.....	① Enables underrun interrupt
MIDL	←	XXH.....	① Sets leading pad or flag pattern (sets DMAC register)
MCMD	←	02H.....	① Enables transmit

- TXINT interrupt processing routine



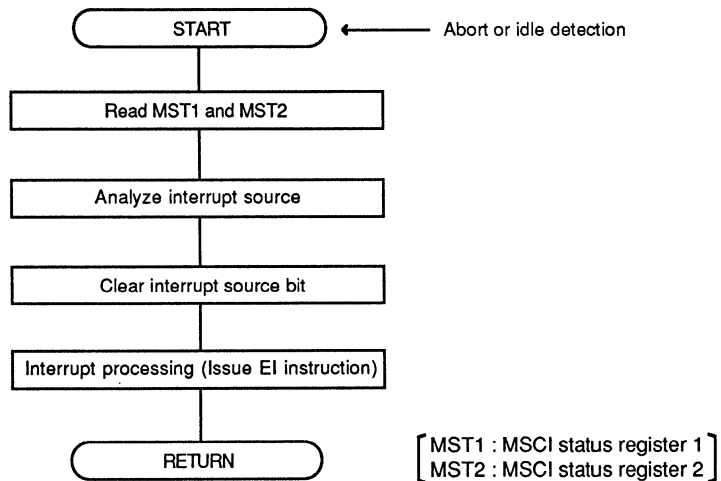
Note: An interrupt also occurs when the DMAC completes the transmission of a frame.

#### 4.8.6 Example of Receive in DMA Chained Block Transfer Mode (bit synchronous HDLC mode)

- Initialization

MCMD	←	21H	.....① Channel reset
MMD0	←	87H	.....① Sets bit synchronous HDLC mode ② Sets CRC-CCITT, initializes to all 1s
MMD1	←	40H	.....① Sets single address 1
MMD2	←	00H	.....① Sets NRZ code ② Sets full duplex mode
MCTL	←	01H	.....① Specifies FCS no-load
MRXS	←	00H	.....① Specifies receive clock RXCM input
MIE0	←	40H	.....① Enables RXINT interrupt
MIE1	←	03H	.....① Enables abort detection interrupt ② Enables idle detection interrupt
MSA0	←	XXH	.....① Secondary station address (Sets DMAC register)
MCMD	←	02H	.....① Enables receive

- RXINT interrupt processing routine



Note: An interrupt is also generated when the DMAC completes the reception of a frame.

## 4.9 Reset Operation

The MSCI is reset to the following condition:

- (1) The receiver and transmitter are disabled, and the transmit/receive buffers are cleared.
- (2) The input/output lines (RXCM and TXCM) are set for input, and the output lines (TXDM and  $\overline{\text{RTSM}}$ ) are placed in inactive.
- (3) All the internal registers are reset, and the following modes are selected.
  - Asynchronous mode (stop bit length of 1, character length of 8 bits, 1/1 clock rate, no parity).
  - Full-duplex communication with NRZ code is selected.
  - The transmit/receive status bits and interrupt enable bits are cleared.
  - The TXCM line input is selected for use as the transmit clock and the RXCM line input as the receive clock.
  - The ADPLL and baud rate generator are initialized.

# Section 5. Asynchronous Serial Communications Interface/ Clocked Serial I/O Port (ASCI/CSIO)

## 5.1 Overview

The asynchronous serial communications interface/clocked serial I/O port (ASCI/CSIO) supports asynchronous and clocked serial communications.

ASCI/CSIO functions in the asynchronous mode are a subset of MSCI functions in the asynchronous mode.\*1 In addition, the ASCII/CSIO can communicate with various asynchronous communications chips, such as the Universal Asynchronous Receiver/Transmitter (UART), Asynchronous Communications Interface Adapter (ACIA), HD64180, and the HD6301.

In the clocked serial mode, the ASCII/CSIO can interface with chips having a clocked serial communications capability, such as the HD64180 and the HD6301.

The operating mode (asynchronous or clocked serial) is selected using mode register 0 (MD0).\*2 MD0 is one of 16 internal registers dedicated to the ASCII/CSIO.

When operated in the asynchronous mode, the ASCII/CSIO and MSCI allow program portability. There are, however, some restrictions that must be observed.\*3

\*1 See section 5.3.1 "Asynchronous Mode."

\*2 For details of the MD0, see section 5.2.1 "ASCI Mode Register 0 (MD0)."

\*3 See section 5.8 "Generating MSCI-Compatible Programs."

### 5.1.1 Functions

Features of the ASCII/CSIO include:

- Choice of asynchronous or clocked serial mode
- NRZ coding
- Full duplex, auto-echo, and local loop-back mode
- Separate transmit and receive buffers (double buffer)

- Modem control signals automatically controlled by auto-enable function
- CTSA (Clear To Send): General-purpose line for input/transmit enable/transition-triggered interrupt
- DCDA (Data Carrier Detect): General-purpose line for input/receive carrier detection/transition-triggered interrupt
- RTSA (Request To Send): General-purpose line for output/transmit request
- Internal programmable baud rate generator
- Choice of external clock or internal baud rate generator as the clock source
- Data transmission rate selectable in the range of 0 to 4 Mbps (using a 10-MHz system clock)
- Four internal interrupt signals RXRDY, TXRDY, RXINT, and TXINT

The features of the asynchronous and clocked serial modes are outlined below:

#### (1) Asynchronous Mode

- Full duplex
- 7- or 8-bit characters (Transmit and receive lengths can be set independently.)
- Stop bit length: 1 or 2 bits
- Even, odd, or no parity
- Parity, overrun, and framing error detection
- Break generation and detection
- Multiprocessor communications support
- Bit rate: 1/1, 1/16, 1/32, or 1/64 of clock frequency

#### (2) Clocked Serial Mode

- Full duplex
- 7- or 8-bit characters
- Even, odd, or no parity
- Parity, overrun, and framing error detection
- Bit rate: 1/1 of clock frequency

### 5.1.2 Configuration and Operation

Figure 5-1 shows a block diagram of the ASCI/CSIO. Receive data is input to the RX buffer via the RXDA line, and bit status, such as parity error, framing error and MP bit, is checked. The receive data is then passed to the receive shift register, and the character is assembled for sending to the receive buffer. The receive status is set in ASCI status register 2 (ST2). The CPU can read the receive data and access this status via the internal data bus.

Transmit data is first written to the transmit buffer and then passed to the transmit shift register where the parity/MP bit, start bit, stop bit, etc., is appended to it. The data is output from the TXDA line. The TX controller controls shift operations in the transmit shift register and data loading from the transmit buffer. It also indicates transmit status, and issues TXRDY and TXINT interrupt request signals.

The RX controller controls shift operations in the receive shift register and data transfers to the receive buffer. It also indicates receive status, and issues RXRDY and RXINT interrupt request signals.



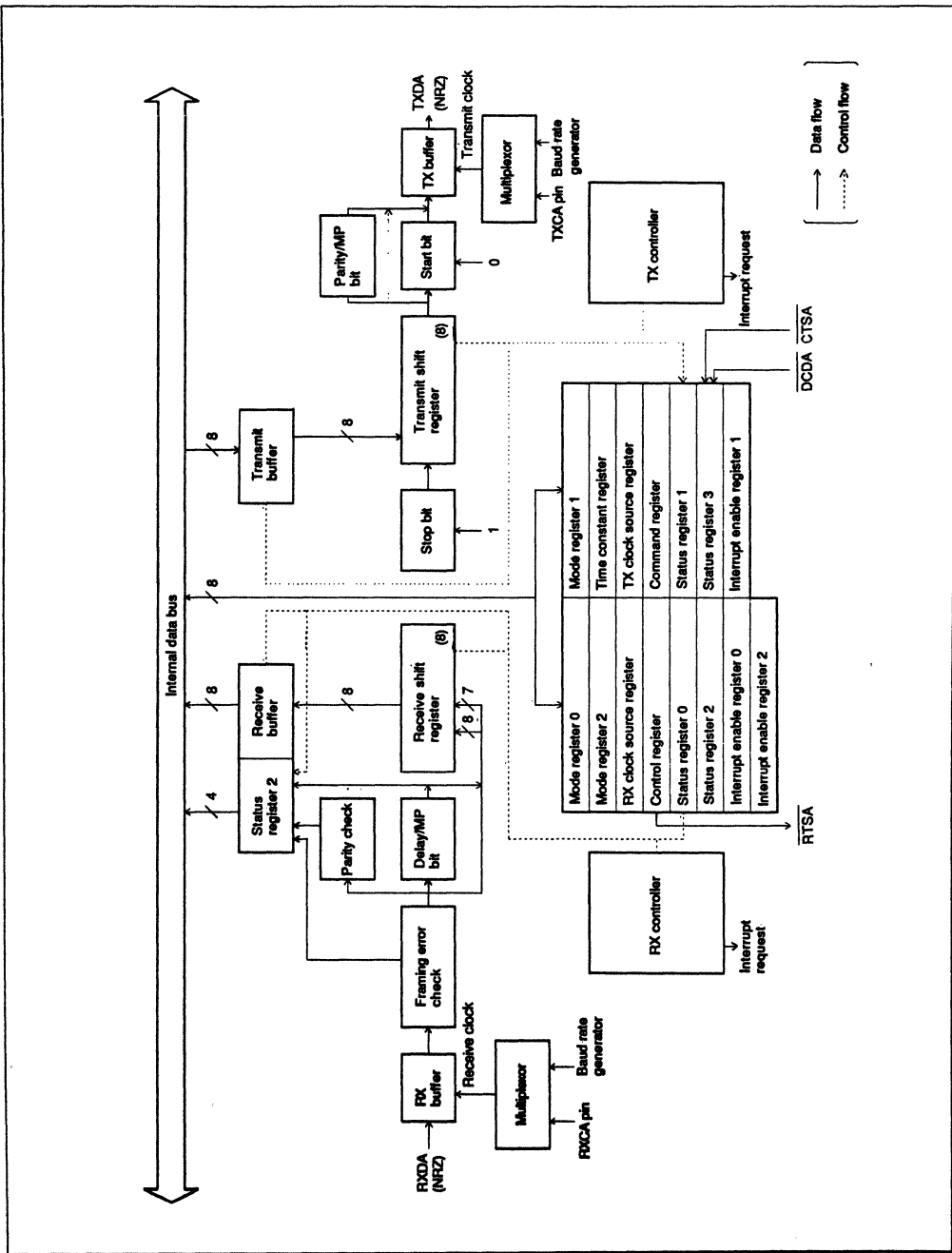


Figure 5-1. ASCII/CSIO Block Diagram

### 5.1.3 Registers

Details of the ASCI/CSIO's 16 dedicated registers are given in table 5-1. These registers are used to select the operating mode (asynchronous or clocked serial), communications protocol, bit rate, and transmit/receive control parameters.

**Table 5-1. ASCI Registers**

Register Name	Symbol	I/O Address	Initial Value <sup>*1</sup>	Read/ Write <sup>*2</sup>
			MSB↔LSB	
ASCI Mode register 0	MD0	0043H	00000000	R/W
ASCI Mode register 1	MD1	0044H	00000000	R/W
ASCI Mode register 2	MD2	0045H	00000000	R/W
ASCI Control register	CTL	0046H	00000001	R/W
ASCI RX clock source register	RXS	004BH	00000000	R/W
ASCI TX clock source register	TXS	004CH	00000000	R/W
ASCI Time constant register	TMC	004AH	00000001	R/W
ASCI Command register	CMD	0042H	—	W
ASCI Status register 0	ST0	0039H	00000000	R
ASCI Status register 1	ST1	003AH	00000000	R/W
ASCI Status register 2	ST2	003BH	00000000	R/W
ASCI Status register 3	ST3	003CH	0000XX <sup>*3</sup> 00	R
ASCI Interrupt enable register 0	IE0	003EH	00000000	R/W
ASCI Interrupt enable register 1	IE1	003FH	00000000	R/W
ASCI Interrupt enable register 2	IE2	0040H	00000000	R/W
ASCI TX/RX buffer register	TRB	0038H	XXXXXXXX	R/W <sup>*4</sup>

(X: Undefined value)

\*1 Registers are set to the initial value by a hardware reset or by a reset command.

\*2 The functions set in the registers differ depending on the operating mode (asynchronous or clocked serial). For details, see section 5.2 "Registers."

\*3 Bits 2 and 3 of ASCII status register 3 read the  $\overline{\text{CTS}}_A$  and  $\overline{\text{DCDA}}$  line levels.

\*4 The ASCII TX/RX buffer register serves as a receive buffer during read operations and as a transmit buffer during write operations.

## 5.2 Registers

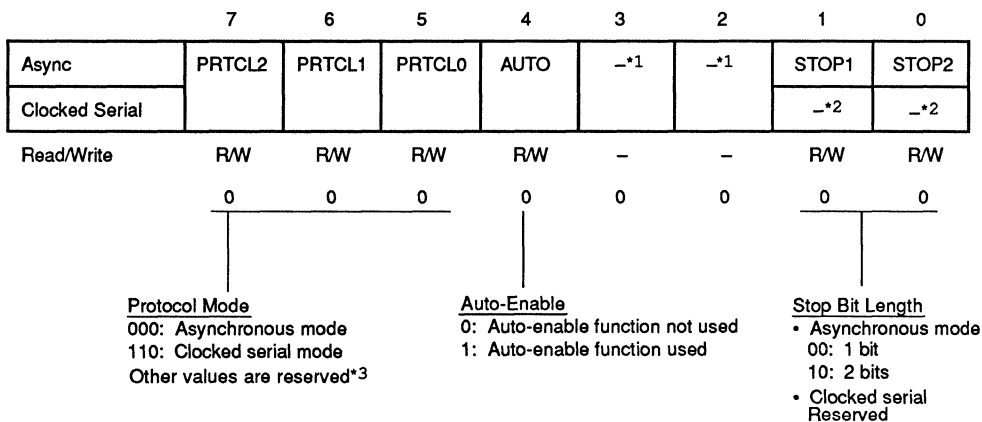
The ASCI/CSIO has 16 registers that are used to select the operating mode and to control the transceiver, receiver and baud rate generator. These registers can be accessed by CPU I/O instructions.

### 5.2.1 ASCI Mode Register 0 (MD0)

This register is used to specify the operating mode (asynchronous or clocked serial), use of the auto-enable function, and the stop bit length (1 or 2 bits) for the asynchronous mode.

The ASCI mode register 0 is reset at the following times:

- When a hardware reset or channel reset command is issued.



\*1 Reserved. These bits always read 0 and should be set to 0.

\*2 Reserved. Read values are undefined. These bits can be set to either 0 or 1.

\*3 Reserved. If these bits are selected, normal operation is not guaranteed.

#### Bits 7-5: PRTCL2-0 (protocol mode)

These bits specify the communications protocol (asynchronous or clocked serial). A channel reset command must be issued before rewriting these bits. If these bits are changed during operation, correct operation can not be guaranteed.

PRTCL2	PRTCL1	PRTCL0	Function
0	0	0	Specifies asynchronous mode
0	0	1	Reserved
⋮	⋮	⋮	
1	0	1	
1	1	0	Specifies clocked serial mode
1	1	1	Reserved

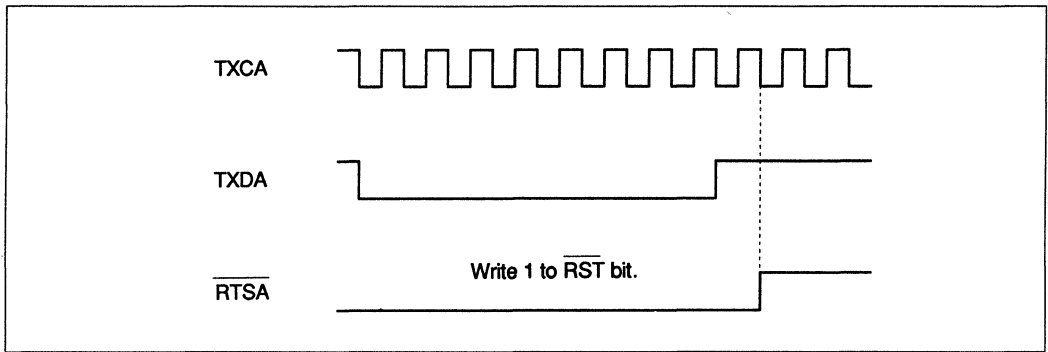
#### Bit 4: AUTO (auto-enable)

This bit specifies the function of the modem control signals ( $\overline{\text{CTSA}}$ ,  $\overline{\text{DCDA}}$ , and  $\overline{\text{RTSA}}$ ). This specification is same for both the asynchronous and clocked serial modes.

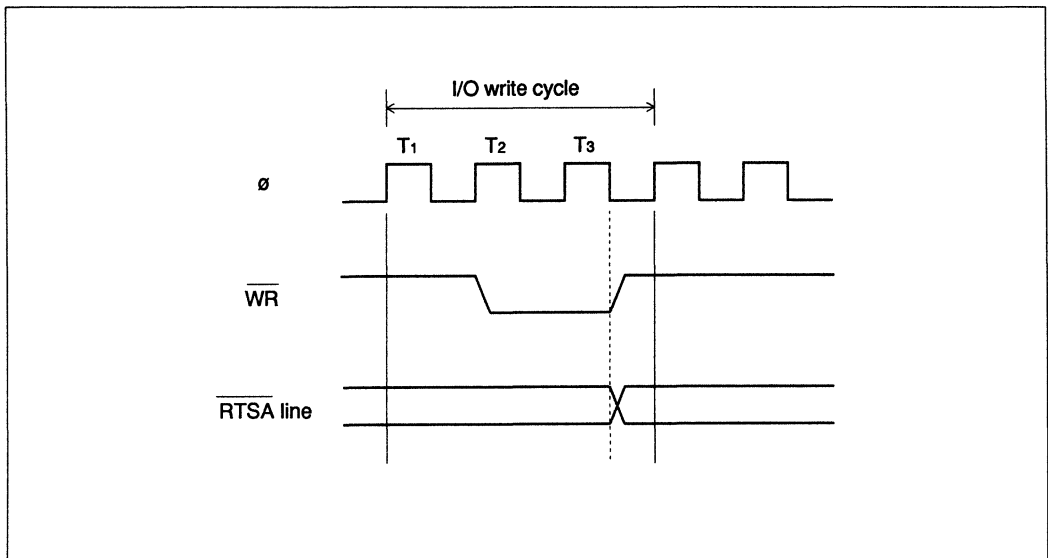
- Asynchronous/clocked serial mode

AUTO	Function
0	The $\overline{\text{CTSA}}$ and $\overline{\text{DCDA}}$ lines are used for general input and the $\overline{\text{RTSA}}$ line is used for general output. These lines function independently of the ASCII/CSIO.
1	Specifies the use of the auto-enable function. When the auto-enable function is used, the $\overline{\text{CTSA}}$ , $\overline{\text{DCDA}}$ , and $\overline{\text{RTSA}}$ lines can be used as modem control lines for an RS-232C interface, etc. $\overline{\text{CTSA}}$ is used to control transmission operations. If the $\overline{\text{CTSA}}$ input is high, data transfer from the transmit buffer to the transmit shift register is inhibited. After transmitting the contents of the transmit shift register, the transmitter enters the idle state. $\overline{\text{DCDA}}$ is used to control reception operations. When $\overline{\text{DCDA}}$ input is high, reception is inhibited. If $\overline{\text{DCDA}}$ goes high during character assembly, the data being assembled is lost. However, the data in the receive buffer is retained. The $\overline{\text{RTSA}}$ output is affected by transmission operation; it is held low regardless of the value of the $\overline{\text{RTS}}$ bit in CTL. When a transmission is not in progress (TX disable or idle status), the $\overline{\text{RTSA}}$ line outputs the $\overline{\text{RTS}}$ bit value.

Figures 5-2 (a) and (b) show modem control signal ( $\overline{\text{RTSA}}$ ) timing. The  $\overline{\text{RTSA}}$  output during writing to the transmit buffer (TRB) is provided at the falling edge of the T3 state. After data transmission, the  $\overline{\text{RTSA}}$  output is set to the high level one clock cycle after the TXDA line is set to mark.



**Figure 5-2. (a) Modem Control Signal Timing**  
 (Auto-enable, 7 bits/character, no parity, 1/1 clock mode)



**Figure 5-2. (b) Modem Control Signal Timing**

**Bits 3-2:** Reserved. These bits always read 0 and should be set to 0.

**Bits 1-0: STOP1-0 (Stop bit length)**

These bits specify the length of the stop bit appended to transmit data in the asynchronous mode. These bits can be rewritten during operation. In this case, the new value applies immediately to the character currently in the transmit buffer.

- Asynchronous mode

STOP1	STOP0	Function
0	0	Stop bit length is 1
0	1	Reserved
1	0	Stop bit length is 2
1	1	Reserved

- Clocked serial mode

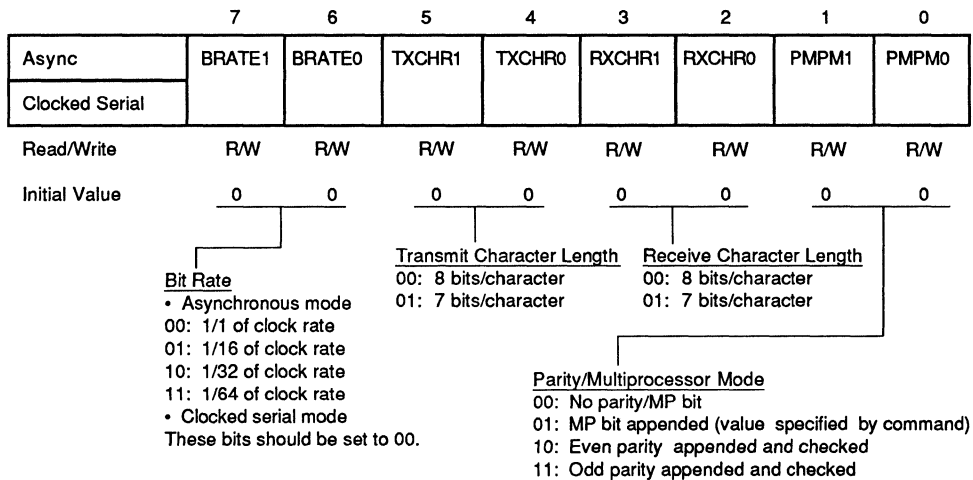
Reserved. Read values are undefined. These bits can be set to either 0 or 1.

### 5.2.2 ASCII Mode Register 1 (MD1)

This register is used to specify the relationship between the bit rate and the transmit clock, the receive clock, the transmit character length, the receive character length, and whether or not the parity/MP bit is to be used.

This register is reset at the following times:

- When a hardware reset or channel reset command is issued.



### Bits 7-6: BRATE1-0 (Bit rate)

These bits specify the relationship between the bit rate and the transmit/receive clock in the asynchronous mode.

In the clocked serial mode, these bits should be set to 00 (only the 1/1 clock is available).

- Asynchronous mode

BRATE1	BRATE0	Function
0	0	Bit rate = 1/1 of the clock rate
0	1	Bit rate = 1/16 of the clock rate
1	0	Bit rate = 1/32 of the clock rate
1	1	Bit rate = 1/64 of the clock rate

- Clocked serial mode

These bits should be set to 00.

### Bits 5-4: TXCHR1-0 (Transmit character length)

These bits specify the transmit character length. These bits can be changed during operation. In this case, the new value takes affect after the current character has been transmitted.

- Asynchronous/Clocked serial mode

TXCHR1	TXCHR0	Function
0	0	Transmit character length = 8 bits
0	1	Transmit character length = 7 bits
1	0	Reserved
1	1	Reserved

### Bits 3-2: RXCHR1-0 (Receive character length)

These bits specify the receive character length. These bits can be changed during operation. In this case, the new value takes affect after the current character has been received.

- Asynchronous/Clocked serial mode

<b>RXCHR1</b>	<b>RXCHR0</b>	<b>Function</b>
0	0	Receive character length = 8 bits
0	1	Receive character length = 7 bits
1	0	Reserved
1	1	Reserved

#### **Bits 1-0: PMPM1-0 (Parity/Multiprocessor mode)**

These bits specify whether to use the parity or multiprocessor (MP) bit in the asynchronous mode. These bits can be changed during operation. In this case, the new value takes effect from the next transmit/receive character.

- Asynchronous/Clocked serial mode

<b>PMPM1</b>	<b>PMPM0</b>	<b>Function</b>
0	0	Neither parity nor MP bit is appended, and parity check not performed
0	1	MP bit is appended (The actual MP bit value is specified by command)*
1	0	Even parity bit is appended and parity check performed
1	1	Odd parity bit is appended and parity check performed

\* See section 5.2.8 "ASCII Command Register (CMD)."

### **5.2.3 ASCII Mode Register 2 (MD2)**

This register is used to specify full duplex, auto-echo, or local loop-back mode for the channel connection. The mode determines how the transmit/receive data is handled on the TXDA and RXDA lines.

The register is reset at the following times:

- When a hardware reset or channel reset command is issued.



	7	6	5	4	3	2	1	0
Async	-*1	-*1	-*1	-*1	-*1	-*1	CNCT1	CNCT0
Clocked Serial								
Read/Write	-	-	-	-	-	-	RW	RW
Initial Value	0	0	0	0	0	0	0	0

Channel Connection  
 00: Full duplex  
 01: Auto-echo  
 10: Reserved\*2  
 11: Local loop-back

\*1 Reserved. These bits always read 0 and should be set to 0.

\*2 Reserved. If this setting is used, normal operating is not guaranteed.

**Bits 7-2:** Reserved. These bits always read 0 and should be set to 0.

**Bits 1-0: CNCT1-0 (Channel connection)**

- Asynchronous/Clocked serial mode

CNCT1	CNCT0	Function
0	0	Full duplex mode (normal transmit/receive operation)
0	1	Auto-echo mode The RXDA line input is output directly to the TXDA line; reception is allowed; transmission is not.
1	0	Reserved
1	1	Local loop-back mode The transmit shift register output is internally connected to the receive shift register input to loop-back the transmit data. The TXDA line echoes the RXDA line input and the TXCA line echoes the RXCA line input.

### 5.2.4 ASCII Control Register (CTL)

This register is used to send a break in the asynchronous mode and to specify the  $\overline{\text{RTSA}}$  line output level in the asynchronous and clocked serial modes.

This register is reset at the following times:

- When a hardware reset or channel reset command is issued.

The BRK bit is also cleared by a TX reset command.

	7	6	5	4	3	2	1	0
Async	—*	—*	—*	—*	BRK	—*	—*	$\overline{\text{RTS}}$
Clocked Serial								
Read/Write	—	—	—	—	R/W	—	—	R/W
Initial Value	0	0	0	0	0	0	0	1

Break Send

- Asynchronous mode
- 0: Off (Normal operation)
- 1: On (Break send)
- Clocked serial mode
- Set this bit to 0

Request to Send

- 0:  $\overline{\text{RTSA}}$  low level output
- 1:  $\overline{\text{RTSA}}$  high level output

\*1 Reserved. These bits always read 0 and should be set to 0.

**Bits 7-4:** Reserved. These bits always read 0 and should be set to 0.

#### Bit 3: BRK (Break send)

This bit is used to initiate a break in the asynchronous mode. In the clocked serial mode, this bit should be set to 0.

- Asynchronous mode

BRK	Function
0	No break sent (Normal operation)
1	When this bit is set to 1, the TXDA line goes low (space) at the falling edge of the next transmit clock. To send a break, this state must continue for two or more character cycles.

The BRK bit is cleared by a TX reset command. For details on break, see "Break send and detection" in section 5.3.1 "Asynchronous Mode."

- Clocked serial mode

This bit should be set to 0.

**Bits 2-1:** Reserved. These bits always read 0 and should be set to 0.

#### Bit 0: $\overline{\text{RTS}}$ (Request to send)

This bit specifies the  $\overline{\text{RTSA}}$  output level. For the  $\overline{\text{RTSA}}$  line level changes when auto-enable is used. See section 5.2.1 "ASCII Mode Register 0."

- Asynchronous/Clocked serial mode

$\overline{\text{RTS}}$	Function
0	$\overline{\text{RTSA}}$ line output is low.
1	$\overline{\text{RTSA}}$ line output is high.

### 5.2.5 ASCII RX Clock Source Register (RXS)

This register is used to specify the receive clock source in the asynchronous mode and to select the master/slave mode of the receiver in the clocked serial mode

The register is reset at the following times:

- When a hardware reset or channel reset command is issued.

	7	6	5	4	3	2	1	0
Async								
Clocked Serial	-*1	RXCS2	RXCS1	RXCS0	-*2	-*2	-*2	-*2
Read/Write	-	R/W	R/W	R/W	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0

Receive Clock Source

- Asynchronous mode
- 000: RXCA line input
- 100: Internal baud rate generator (BRG) output
- Others: Reserved\*3

RX Master/Slave Mode Select

- Clocked serial mode
- 000: Slave mode
- 100: Master mode
- Others: Reserved\*3

\*1 This bit always reads 0 and should be set to 0.

\*2 Reserved. These bits always read 0. Set these bits equal to the TXBR3-0 bits (bits 3-0) in TXS. See section 5.2.6 "ASCI TX Clock Source Register (TXS)."

\*3 Reserved. If any other settings are selected, normal operation is not guaranteed.

**Bit 7:** Reserved. This bit always reads 0 and should be set to 0.

**Bits 6-4: RXCS2-0 (Receive clock source)**

- Asynchronous mode

In the asynchronous mode, these bits are used to select the receive clock source.

RXCS2	RXCS1	RXCS0	Function
0	0	0	The clock input to the RXCA line is used.
0	0	1	Reserved
0	1	0	
0	1	1	
1	0	0	The internal BRG output is used. BRG parameters are set in the ASCI TX clock source register (TXS) (bits 3-0) and the time constant register (TMC). The RXCA line outputs the internally generated receive clock.
1	0	1	Reserved
1	1	0	
1	1	1	

- Clocked serial mode

These bits specify the master or slave mode. For details regarding each mode, see section 5.3.2 "Clocked Serial Mode." When the receiver is used in the master mode, the transmitter must also be in the master mode. Normal operation is not guaranteed if the transmitter is used in the slave mode. These bits should be changed when both the transmitter and receiver are in the disable or idle state.

<b>RXCS2</b>	<b>RXCS1</b>	<b>RXCS0</b>	<b>Function</b>
0	0	0	Slave mode
0	0	1	Reserved
0	1	0	
0	1	1	
1	0	0	Master mode (When the receiver is placed in the master mode, the transmitter must also be placed in the master mode.)
1	0	1	Reserved
1	1	0	
1	1	1	

**Bits 3-0:** Reserved. Set these bits equal to the TXBR3-0 bits (bits 3-0) in the ASCI TX clock source register. See section 5.2.6 "ASCI TX Clock Source Register (TXS)."

### 5.2.6 ASCI TX Clock Source Register (TXS)

This register is used to select the transmit clock source, the internal baud rate generator (BRG) clock rate in the asynchronous mode, the transmit mode (slave or master), and the internal BRG clock rate in the clocked serial mode.

The register is reset at the following times:

- When a hardware reset or channel reset command is issued.

	7	6	5	4	3	2	1	0
Async	..* 1	TXCS2	TXCS1	TXCS0	TXBR3	TXBR2	TXBR1	TXBR0
Clocked Serial								

Read/Write      -      R/W      R/W      R/W      R/W      R/W      R/W      R/W

Initial Value    0      0      0      0      0      0      0      0

Transmit Clock Source

- Asynchronous mode
- 000: TXCA line input
- 100: Internal baud rate generator (BRG) output
- Others: Reserved\*2

TX Master/Slave Mode Select

- Clocked serial mode
- 000: Slave mode
- 100: Master mode
- Others: Reserved\*2

Baud Rate

- Clock division ratio
- 0000: 1/1    0101: 1/32
- 0001: 1/2    0110: 1/64
- 0010: 1/4    0111: 1/128
- 0011: 1/8    1000: 1/256
- 0100: 1/16   1001: 1/512
- Others: Reserved\*2

\*1 Reserved. This bit always reads 0 and should be set to 0.

\*2 Reserved. If set to other values, normal operation is not guaranteed.

**Bit 7:** Reserved. This bit always reads 0 and should be set to 0.

**Bits 6-4: TXCS2-0 (Transmit clock source)**

- Asynchronous mode

In the asynchronous mode, these bits are used to select the transmit clock source.

TXCS2	TXCS1	TXCS0	Function
0	0	0	The clock input to the TXCA line is used.
0	0	1	Reserved
0	1	0	
0	1	1	
1	0	0	The internal BRG output is used. BRG parameters are set in the ASCII TX clock source register (TXS) (bits 3-0) and the time constant register (TMC). The TXCA line outputs the internally generated transmit clock. For details, see section 5.5 "Baud Rate Generator."
1	0	1	Reserved
1	1	0	
1	1	1	

- Clocked serial mode

In the clocked serial mode, these bits are used to select the master or slave mode for the transmitter.

These bits should be changed only when both the transmitter and the receiver are in the disable or idle state.

TXCS2	TXCS1	TXCS0	Function
0	0	0	Slave mode
0	0	1	Reserved
0	1	0	
0	1	1	
1	0	0	Master mode
1	0	1	Reserved
1	1	0	
1	1	1	

For the details of each mode, see section 5.3.2 "Clocked Serial Mode."

#### Bits 3-0: TXBR3-0 (Baud rate )

These bits specify the transmit/receive clock rate output by the internal BRG. The transmit/receive BRG output is generated by dividing the reload timer output frequency. These bits specify the division ratio. Setting values and division ratios are given below. For details, see section 5.5 "Baud Rate Generator."

- Asynchronous/Clocked serial mode

TXBR3	TXBR2	TXBR1	TXBR0	Division Ratio
0	0	0	0	1/1
0	0	0	1	1/2
0	0	1	0	1/4
0	0	1	1	1/8
0	1	0	0	1/16
0	1	0	1	1/32
0	1	1	0	1/64
0	1	1	1	1/128
1	0	0	0	1/256
1	0	0	1	1/512
1	0	1	0	Reserved
	⋮			
1	1	1	1	

### 5.2.7 ASCII Time Constant Register (TMC)

This register is used to specify the value to be loaded into the reload timer (inside the internal BRG).

The specified value is used for both transmit and receive.

This register is reset at the following times:

- When a hardware reset or channel reset command is issued.

	7	6	5	4	3	2	1	0
Async	TMC7	TMC6	TMC5	TMC4	TMC3	TMC2	TMC1	TMC0
Clocked Serial								
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	1


  
 Reload Timer Value (1 - 256)



## Bits 7-0: TMC 7-0 (Time Constant)

- Asynchronous/Clocked serial mode

These bits are used to specify the value (1 – 256) to be loaded into the reload timer (inside the internal BRG). The specified value is used for both transmit and receive. (When 0 is specified, 256 is assumed.) The reload timer output frequency is:

$$f\phi/TMC$$

where,  $f\phi$  is the CPU clock frequency and TMC is the value of the time constant register.

In the asynchronous mode, the actual baud rate is determined by the values of: TMC (1-256), TXBR (0-9 determined by the TXBR3-0 bits (bits 3-0) of the TX clock source register), and CM\* (clock mode = bit rate/clock rate). In the clocked serial mode, this is determined by the values of TMC and TXBR. For details, see section 5.5 "Baud Rate Generator."

\* The clock mode is specified by the BRATE1-0 bits (bits 7 and 6) in MD1.

### 5.2.8 ASCII Command Register (CMD)

This register is used in the asynchronous or clocked serial mode to specify the commands that control various ASCII transmission and reception operations.

This register is a write-only register and always reads 00H.

	7	6	5	4	3	2	1	0
Async	- *1	- *1	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
Clocked Serial								
Read/Write	-	-	W	W	W	W	W	W
Initial Value	-	-	-	-	-	-	-	-

#### Command

- |  |  |   |
|--|--|---|
| <ul style="list-style-type: none"> <li>Transmit commands</li> <li>000001: TX reset</li> <li>000010: TX enable</li> <li>000011: TX disable</li> <li>001000: MP bit on</li> <li>001001: TX buffer clear</li> </ul> | <ul style="list-style-type: none"> <li>Receive commands</li> <li>010001: RX reset</li> <li>010010: RX enable</li> <li>010011: RX disable</li> <li>010110: Search MP bit</li> </ul> | <ul style="list-style-type: none"> <li>Other commands</li> <li>100001: Channel reset</li> <li>000000: No operation</li> <li>Others: Reserved*2</li> </ul> |
|--|--|---|

\*1 Reserved. These bits always read 0 and should be set to 0.

\*2 Reserved. If other settings are used, normal operation is not guaranteed.

**Bits 7 and 6:** Reserved. These bits always read 0 and should be set to 0.

**Bits 5-0: CMD5-0 (Command)**

- Asynchronous/Clocked serial mode

The command set includes transmit and receive commands as well as a channel reset command and a no operation code. The codes and functions for each command are given in tables 5-2 through 5-4.

(1) Transmit commands

**Table 5-2. Transmit Commands**

<b>Command Name (set value)</b>	<b>Function</b>
TX reset (01H)	Immediately places the transmitter in the TX disable state (the transmit line goes high (mark)). The transmit buffer is cleared, bit 1 of ST0, and the BRK bit in CTL are cleared. Other registers are not affected.
TX enable (02H)	Places the transmitter in the idle state (the transmit line goes to mark if it is in the TX disable state).
TX disable (03H)	Forcibly disables the TXRDY state (TXRDY bit in ST0 reset). The contents of the transmit buffer and transmit shift register are transmitted, then the transmitter enters the disable state.
MP bit on (08H)	An MP bit (value 1) is appended to the transmit data. This command is valid only for the next character loaded into the transmit buffer.
TX buffer clear (09H)	Clears the transmit buffer. Other registers are not affected.

## (2) Receive commands

**Table 5-3. Receive Commands**

<b>Command Name (set value)</b>	<b>Function</b>
RX reset (11H)	Stops the receive shift register and places the receiver in the RX disable state. The receive buffer is cleared and bit 0 in ST0 is reset. Other registers are not affected.
RX enable (12H)	If the receiver is in the RX disable state when this command is issued, the receiver enters the start bit search state.
RX disable (13H)	Immediately places the receiver in the disable state. The receive shift register contents are lost. However, the receive buffer is not affected.
Search MP bit (16H)	Inhibits the transfer of characters with MP bit = 0 to the receive buffer. This function continues until a character with MP bit = 1 is received.

## (3) Other commands

**Table 5-4. Other Commands**

<b>Command Name (set value)</b>	<b>Function</b>
Channel reset (21H)	Immediately places the transmitter and receiver in the disable state, clears the transmit and receive buffers, and resets all registers to their initial values (same as hardware reset).
No operation (00H)	The transmitter and receiver operate normally.

### 5.2.9 ASCI Status Register 0 (ST0)

This register is used in the asynchronous and clocked serial mode to indicate the status of interrupts TXINT and RXINT, as well as the status of the transmit and receive buffers.

This register is reset at the following times:

- When a hardware reset or channel reset command is issued.
- When the system stop mode is entered.

	7	6	5	4	3	2	1	0
Async	TXINT	RXINT	-*	-*	-*	-*	TXRDY	RXRDY
Clocked Serial								

Read/Write      R      R      -      -      -      -      R      R

Initial Value      0      0      0      0      0      0      0      0

**TXINT Interrupt**

0: TX interrupt not requested  
1: TX interrupt requested

**RXINT Interrupt**

0: RX interrupt not requested  
1: RX interrupt requested

**RX Ready**

0: Receive data does not exist  
1: Receive data exists

**TX Ready**

0: Transmit buffer not empty  
1: Transmit buffer empty

\* Reserved. These bits always read 0.

### Bit 7: TXINT (TXINT interrupt)

This bit indicates the TXINT interrupt status. When both this bit and the TXINTE bit in IE0 are 1, a TXINT internal interrupt request is issued to the CPU.

• Asynchronous/Clocked serial mode

TXINT	Function
0	A TXINT interrupt request has not been issued.
1	A TXINT interrupt request has been issued. The TXINT value is determined by $TXINT = IDL \cdot IDLE + CCTS \cdot CCTSE$ IDL and CCTS are bits 6 and 3 of ST1 IDL and CCTSE are bits 6 and 3 of IE1 This bit is set to 1 when: <ul style="list-style-type: none"> <li>• The IDLE bit is set to 1 and idle state is entered.</li> <li>• The CCTSE bit is set to 1 and the <math>\overline{CTSA}</math> line level is changed.</li> </ul>

### Bit 6: RXINT (RXINT interrupt)

This bit indicates the RXINT interrupt status. When both this bit and the RXINTE bit in IE0 are 1, an RXINT interrupt request is issued to the CPU.

- Asynchronous/Clocked serial mode

<b>RXINT</b>	<b>Function</b>
0	An RXINT interrupt request has not been issued.
1	An RXINT interrupt request has been issued. The RXINT bit is set when: <ol style="list-style-type: none"> <li>(1) The CDCDE bit is set and <math>\overline{\text{DCDA}}</math> line level is changed.</li> <li>(2) The BRKDE bit is set and start of break is detected.</li> <li>(3) The BRKEE bit is set and end of break is detected.</li> <li>(4) The PMPE bit is set and parity bit, MP bit or receive data MSB is set to 1. (When the PMPM1-0 bits of MD1 is 10 or 11, the parity bit is set to 1, or When the PMPM1-0 bits of MD1 is 01, the MP bit is set to 1, or When the PMPM1-0 bits of MD1 is 00, the MSB is set to 1.)</li> <li>(5) The PEE bit is set and a parity error has occurred.</li> <li>(6) The FRMEE bit is set and a framing error is detected.</li> <li>(7) The OVRNE bit is set and an overrun error is detected.</li> </ol> The RXINT value is determined by: $\text{RXINT} = \text{CDCD} \cdot \text{CDCDE} + \text{BRKD} \cdot \text{BRKDE} + \text{BRKE} \cdot \text{BRKEE} + \text{PMP} \cdot \text{PMPE} + \text{PE} \cdot \text{PEE} + \text{FRME} \cdot \text{FRMEE} + \text{OVRN} \cdot \text{OVRNE}$ The CDCD, BRKD, and BRKE bits are bits 2, 1 and 0 in ST1. The PMP, PE, FRME, and OVRN bits are bits 6, 5, 4 and 3 in ST2. The CDCDE, BRKDE, and BRKEE bits are bits 2, 1 and 0 in IE1. The PMPE, PEE, FRMEE, and OVRNE bits are bits 6, 5, 4 and 3 in IE2.

**Bits 5-2:** Reserved. These bits always read 0.

**Bit 1: TXRDY (TX ready)**

This bit is a read-only status flag that indicates whether or not data can be loaded into the transmit buffer. It is set to 1 when the transmitter is in the TX enable state and the transmit buffer is empty. At all other times this bit is 0.

When this bit and the TXRDYE bit (bit 1 in IE0) are 1, a TXRDY interrupt request is issued.

- Asynchronous/Clocked serial mode

<b>TXRDY</b>	<b>Function</b>
0	Indicates that the transmit buffer is not empty (data cannot be written to the transmit buffer).
1	Indicates that the transmit buffer is empty (data can be written to the transmit buffer).

### Bit 0: RXRDY (RX ready)

This bit is a read-only status flag that indicates whether or not the receive buffer has data waiting to be read. It is set to 1 when the receive buffer has data, regardless of the RX enable/disable state. It reads 0 when the receive buffer has no data.

When this bit and the RXRDYE bit (bit 0 in IE0) are 1, an RXRDY interrupt request is issued.

- Asynchronous/Clocked serial mode

RXRDY	Function
0	Indicates that the receive buffer has no data.
1	Indicates that the receive buffer has data.

### 5.2.10 ASCI Status Register 1 (ST1)

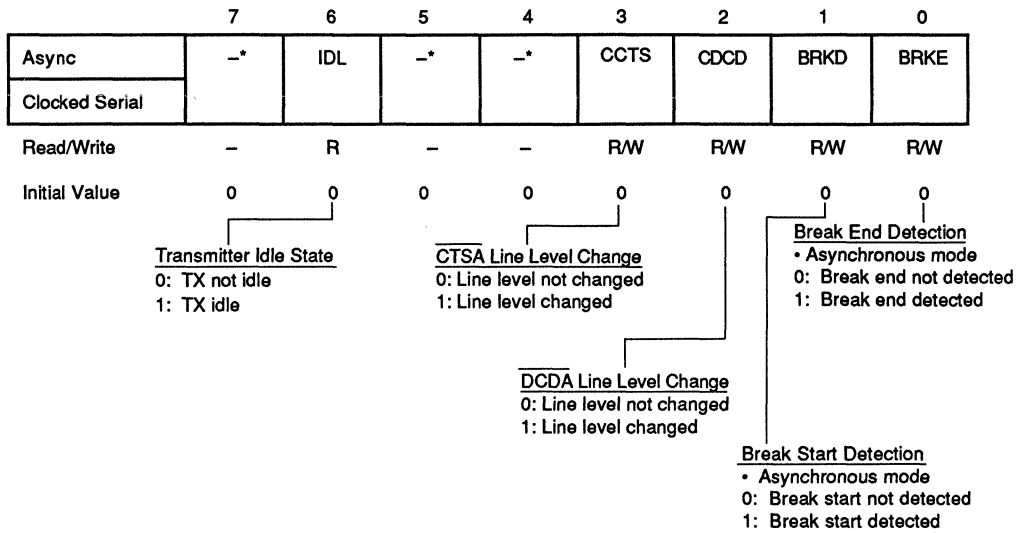
This register indicates the transmitter idle state,  $\overline{\text{CTSA}}$  and  $\overline{\text{DCDA}}$  line level changes, and the detection of a break start or end.

When bits 3 and 6 in this register go to 1, they can be used to generate a TXINT interrupt.

When bits 2-0 of this register go to 1, they can be used to generate an RXINT interrupt.

The register is reset under the following conditions:

- When 1 is written to bit positions 3, 2, 1, or 0 of this register, the bit is cleared.
- Issuing a TX reset command clears bits 6 and 3.
- Issuing an RX reset command clears bits 2, 1, and 0.
- Issuing a channel reset command or entering system stop mode clears all the bits of this register.



- Reserved. These bits always read 0.

**Bit 7:** Reserved. This bit always reads 0 and can be set to either 0 or 1.

**Bit 6: IDL (Transmitter idle state)**

This bit indicates whether or not the transmitter is in the idle state.\*

When this bit and the IDLE bit (bit 6 in IE1) are 1, a TXINT interrupt request is issued.

\* The idle state is explained below.

- Idle State {
- Asynchronous mode — No transmit data is available and the TXDA line is set to mark.
  - Clocked serial mode — The MSB of the previously transmitted data is retained.

- Asynchronous/Clocked serial mode

IDL	Function
0	The transmitter is not in the idle state.
1	The transmitter is in the idle state.

This bit is a read-only flag. It is cleared to 0 only when the transmitter leaves the idle state (for example, when data is loaded into the transmit buffer).

**Bits 5 and 4:** Reserved. These bits always read 0 and can be set to either 0 or 1.

**Bit 3: CCTS ( $\overline{\text{CTSA}}$  line level change)**

This bit is set to 1 when the  $\overline{\text{CTSA}}$  line level changes. It is cleared when 1 is written to this bit position.

When this bit and the CCTSE bit (bit 3 in IE1) are equal to 1, a TXINT interrupt request is issued to the CPU.

- Asynchronous/Clocked serial mode

CCTS	Function
0	The $\overline{\text{CTSA}}$ input level has not changed.
1	The $\overline{\text{CTSA}}$ input level has changed.

**Bit 2: CDCD ( $\overline{\text{DCDA}}$  line level change)**

This bit is set to 1 when the  $\overline{\text{DCDA}}$  line level changes. It is cleared when 1 is written to this bit position.

When this bit and the CDCDE bit (bit 2 in IE1) are equal to 1, an RXINT interrupt request is issued to the CPU.



- Asynchronous/Clocked serial mode

CDCD	Function
0	The $\overline{\text{DCDA}}$ line input level has not changed.
1	The $\overline{\text{DCDA}}$ line input level has changed.

#### Bit 1: BRKD (Break start detection)

This bit is set to 1 when a break start is detected. It is cleared when 1 is written to this bit position. When this bit and the BRKDE bit (bit 1 in IE1) are 1, a RXINT interrupt request is issued.

- Asynchronous mode

BRKD	Function
0	A break start has not been detected.
1	A break start has been detected.

- Clocked serial mode

This bit is never set to 1 in clocked serial mode.

#### Bit 0: BRKE (Break end detection)

This bit is set to 1 when a break end is detected. It is cleared when 1 is written to this bit position. When this bit and the BRKEE bit (bit 0) in IE1 are equal to 1, an RXINT internal interrupt request is issued to the CPU.

- Asynchronous mode

BRKE	Function
0	A break end has not been detected.
1	A break end has been detected.

- Clocked serial mode

This bit is never set to 1 in clocked serial mode.

## 5.2.11 ASCII Status Register 2 (ST2)

Since the ASCII/CSIO receive buffer has only 1 stage, the receive status is available in ST2 immediately after the character has been assembled.

This register indicates the parity/MP bit or the MSB value and whether or not parity errors, framing errors, or overrun errors have occurred.

This register is reset under the following conditions:

- Writing 1 to a bit in this register clears the bit.
- Issuing an RX reset or channel reset command resets the entire register.
- System stop mode resets the entire register.

The bits in this register can be used to generate an RXINT interrupt request.

	7	6	5	4	3	2	1	0
Async	—*	PMP	PE	FRME	OVRN	—*	—*	—*
Clocked Serial								
Read/Write	—	R/W	R/W	R/W	R/W	—	—	—
Initial Value	0	0	0	0	0	0	0	0

Parity/MP Bit  
Parity/MP bit value  
0: Parity/MP bit value 0  
1: Parity/MP bit value 1

Overrun Error  
0: No overrun error detected  
1: Overrun error detected

Framing Error  
0: No framing error detected  
1: Framing error detected

Parity Error  
0: No parity error detected  
1: Parity error detected

- Reserved. These bits always read 0 and can be set to 0 or 1.

**Bit 7:** Reserved. This bit always reads 0 and can be set to 0 or 1.

**Bit 6: PMP (Parity/MP bit)**

This bit indicates the value of the parity bit when parity checking is selected (i.e. the PMPM1-0 bits of MD1 are set to 10 for even parity or 11 for odd parity).

When the MP bit is selected (i.e. the PMPM1-0 bits are set to 01), the value of the MP bit is indicated. The MSB of the received character is indicated when the parity/MP bit is not used (i.e. the PMPM1-0 bits are set to 00).

- Asynchronous/Clocked serial mode

<b>PMP</b>	<b>Function</b>
0	The parity bit, MP bit, or the received character MSB is 0.
1	The parity bit, MP bit, or the received character MSB is 1.

The PMP bit is changed when the receive buffer becomes ready to receive the next character.

This bit can be cleared by writing 1 to this bit position.

When this bit and the PMPE bit in IE2 are equal to 1, an RXINT interrupt request is issued to the CPU.

**Bit 5: PE (Parity error)**

- Asynchronous/Clocked serial mode

<b>PE</b>	<b>Function</b>
0	No parity error has occurred.
1	A parity error has occurred.

Once the PE bit is set, it is not cleared until 1 is written to this bit position or the receiver is reset.

When this bit and the PEE bit (bit 5) in IE2 are equal to 1, an RXINT interrupt request is issued to the CPU.

#### Bit 4: FRME (Framing error)

- Asynchronous mode

FRME	Function
0	No framing error has occurred.
1	A framing error has occurred.

Once the FRME bit is set, it is not cleared until 1 is written to this bit position or the receiver is reset. When this bit and the FRMEE bit in IE2 are equal to 1, an RXINT interrupt request is issued to the CPU.

- Clocked serial mode

Framing errors cannot be detected in the clocked serial mode. When read, this bit is always 0.

#### Bit 3: OVRN (Overflow error)

- Asynchronous/Clocked serial mode

OVRN	Function
0	No overrun error has occurred.
1	An overrun error has occurred.

Once the OVRN bit is set, it is not cleared until 1 is written to this bit position or the receiver is reset. When this bit and the OVRNE bit in IE2 are equal to 1, an RXINT interrupt request is issued to the CPU.

**Bits 2-0:** Reserved. These bits always read 0 and can be set to 0 or 1.

### 5.2.12 ASCII Status Register 3 (ST3)

This register is used to indicate the levels of the  $\overline{\text{CTSA}}$  and  $\overline{\text{DCDA}}$  input lines and the enable/disable state of the transmitter and receiver.

This register is a read-only register.

The register is reset under the following conditions:

- Issuing a TX reset command clears bits 3 and 1.
- Issuing an RX reset command clears bits 2 and 0.
- Issuing a hardware reset command or a channel reset command resets the entire register.
- System stop mode resets the entire register.

	7	6	5	4	3	2	1	0
Async	–*1	–*1	–*1	–*1	$\overline{\text{CTS}}$	$\overline{\text{DCD}}$	TXENBL	RXENBL
Clocked Serial								
Read/Write	–	–	–	–	R	R	R	R
Initial Value	0	0	0	0	X*2	X*2	0	0

$\overline{\text{CTS}}$  Input Line Level

0:  $\overline{\text{CTS}}$  line low level

1:  $\overline{\text{CTS}}$  line high level

$\overline{\text{DCDA}}$  Input Line Level

0:  $\overline{\text{DCDA}}$  line low level

1:  $\overline{\text{DCDA}}$  line high level

TX Enable

0: Disable

1: Enable

RX Enable

0: Disable

1: Enable

\*1 Reserved. These bits always read 0.

\*2 Indicates that the value is undefined.

**Bits 7-4:** Reserved. These bits always read 0.

**Bit 3:  $\overline{\text{CTS}}$  ( $\overline{\text{CTSA}}$  input line level)**

This read-only bit indicates the level of the  $\overline{\text{CTSA}}$  input line. This bit is not used to generate an interrupt.

- Asynchronous/Clocked serial mode

$\overline{\text{CTS}}$	Function
0	The $\overline{\text{CTSA}}$ input is low.
1	The $\overline{\text{CTSA}}$ input is high.

**Bit 2:  $\overline{\text{DCD}}$  ( $\overline{\text{DCDA}}$  input line level)**

This read-only bit indicates the level of the  $\overline{\text{DCDA}}$  input line. This bit is not used to generate an interrupt.

- Asynchronous/Clocked serial mode

$\overline{\text{DCD}}$	Function
0	The $\overline{\text{DCDA}}$ input is low.
1	The $\overline{\text{DCDA}}$ input is high.

**Bit 1: TXENBL (TX enable)**

This read-only bit indicates the transmitter enable/disable state.  
Issue a command to change the enable/disable state.

- Asynchronous/Clocked serial mode

TXENBL	Function
0	The transmitter is in the disable state.
1	The transmitter is in the enable state.

## Bit 0: RXENBL (RX enable)

This read-only bit indicates the receiver enable/disable state.  
Issue a command to change the enable/disable state.

- Asynchronous/Clocked serial mode

### RXENBL    Function

0	The receiver is in the disable state.
1	The receiver is in the enable state.

### 5.2.13 ASCI Interrupt Enable Register 0 (IE0)

This register is used to specify whether to enable/disable the TXINT, RXINT, TXRDY, or RXRDY internal interrupts. An interrupt request can only be generated when the ST0 status bit is set and the corresponding bit of this register is equal to 1.

For details about interrupts, see section 5.6 "Internal Interrupts."

	7	6	5	4	3	2	1	0
Async	TXINTE	RXINTE	-*	-*	-*	-*	TXRDYE	RXRDYE
Clocked Serial								
Read/Write	R/W	R/W	-	-	-	-	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

<u>TXINT Interrupt Enable</u> 0: Disable 1: Enable	<u>RXINT Interrupt Enable</u> 0: Disable 1: Enable	<u>TXRDY Interrupt Enable</u> 0: Disable 1: Enable	<u>RXRDY Interrupt Enable</u> 0: Disable 1: Enable
--	--	--	--

\* Reserved. These bits always read 0 and should be set to 0.

**Bit 7: TXINTE (TXINT interrupt enable)**

- Asynchronous/Clocked serial mode

<b>TXINTE</b>	<b>Function</b>
0	Does not issue a TXINT internal interrupt request.
1	Issues a TXINT internal interrupt request when the TXINT bit of ST0 is 1.

**Bit 6: RXINTE (RXINT interrupt enable)**

- Asynchronous/Clocked serial mode

<b>RXINTE</b>	<b>Function</b>
0	Does not issue an RXINT internal interrupt.
1	Issues an RXINT internal interrupt request when the RXINT bit of ST0 is 1.

**Bit 5-2:** Reserved. These bits always read 0 and should be set to 0.

**Bit 1: TXRDYE (TXRDY interrupt enable)**

- Asynchronous/Clocked serial mode

<b>TXRDYE</b>	<b>Function</b>
0	Does not issue a TXRDY interrupt.
1	Issues a TXRDY interrupt request when the TXRDY bit of ST0 is 1.

**Bit 0: RXRDYE (RXRDY interrupt enable)**

- Asynchronous/Clocked serial mode

<b>RXRDYE</b>	<b>Function</b>
0	Does not issue an RXRDY interrupt.
1	Issues an RXRDY interrupt request when the RXRDY bit of ST0 is 1.



## 5.2.14 ASCI Interrupt Enable Register 1 (IE1)

This register is used to enable or disable a TXINT/RXINT internal interrupt originating from a status bit (IDL, CCTS, CDCD, BRKD, or BRKE) in ST1.

When a specific enable bit of this register and the corresponding status bit of ST1 are equal to 1, the TXINT bit or RXINT bit of ST0 is set. This causes a TXINT or RXINT interrupt request to be issued to the CPU.

For details about interrupts, see section 5.6 "Internal Interrupts."

	7	6	5	4	3	2	1	0
Async	—*	IDL	—*	—*	CCTSE	CDCDE	BRKDE	BRKEE
Clocked Serial								
Read/Write	—	R/W	—	—	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

<p><u>IDL Interrupt Enable</u> 0: Disable 1: Enable</p>	<p><u>CCTS Interrupt Enable</u> 0: Disable 1: Enable</p>	<p><u>CDCD Interrupt Enable</u> 0: Disable 1: Enable</p>	<p><u>BRKD Interrupt Enable</u> • Asynchronous mode 0: Disable 1: Enable</p>	<p><u>BRKE Interrupt Enable</u> • Asynchronous mode 0: Disable 1: Enable</p>
---	--	--	--	--

\* Reserved. These bits always read 0 and should be set to 0.

**Bit 7:** Reserved. This bit always reads 0 and should be set to 0.

**Bit 6: IDLE (IDL interrupt enable)**

This bit specifies whether or not to enable a TXINT internal interrupt originating in the IDL bit of ST1.

- Asynchronous/Clocked serial mode

<b>IDLE</b>	<b>Function</b>
0	Disables a TXINT internal interrupt originating in the IDL bit.
1	Enables a TXINT internal interrupt originating in the IDL bit. If the IDL bit in ST1 is 1, the TXINT bit in ST0 is set to 1 and a TXINT internal interrupt request is issued to the CPU.

**Bits 5-4:** Reserved. These bits always read 0 and should be set to 0.

**Bit 3: CCTSE (CCTS interrupt enable)**

This bit specifies whether or not to enable a TXINT internal interrupt originating in the CCTS bit of ST1.

- Asynchronous/Clocked serial mode

<b>CCTSE</b>	<b>Function</b>
0	Disables a TXINT internal interrupt originating in the CCTS bit.
1	Enables a TXINT internal interrupt originating in the CCTS bit. If the CCTS bit in ST1 is 1, the TXINT bit in ST0 is set to 1 and a TXINT internal interrupt request is issued to the CPU.

**Bit 2: CDCDE (CDCD interrupt enable)**

This bit specifies whether to enable an RXINT internal interrupt originating in the CDCD bit of ST1.

- Asynchronous/Clocked serial mode

<b>CDCDE</b>	<b>Function</b>
0	Disables an RXINT internal interrupt originating in the CDCD bit.
1	Enables an RXINT internal interrupt originating in the CDCD bit. If the CDCD bit in ST1 is 1, the RXINT bit in the ST0 is set to 1 and an RXINT internal interrupt request is issued to the CPU.

**Bit 1: BRKDE (BRKD interrupt enable)**

This bit specifies whether to enable an RXINT internal interrupt originating in the BRKD bit of ST1.

- Asynchronous mode

<b>BRKDE</b>	<b>Function</b>
0	Disables an RXINT internal interrupt originating in the BRKD bit.
1	Enables an RXINT internal interrupt originating in the BRKD bit. If the BRKD bit in ST1 is 1, the RXINT bit in ST0 is set to 1 and an RXINT internal interrupt request is issued to the CPU.

- Clocked serial mode

This bit is never set in the clocked serial mode.

**Bit 0: BRKEE (BRKE interrupt enable)**

This bit specifies whether to enable an RXINT internal interrupt originating in the BRKE bit of ST1.

- Asynchronous mode

<b>BRKEE</b>	<b>Function</b>
0	Disables an RXINT internal interrupt originating in the BRKE bit.
1	Enables an RXINT internal interrupt originating in the BRKE bit. If the BRKE bit in ST1 is 1, the RXINT bit in ST0 is set to 1 and an RXINT internal interrupt request is issued to the CPU.

- Clocked serial mode

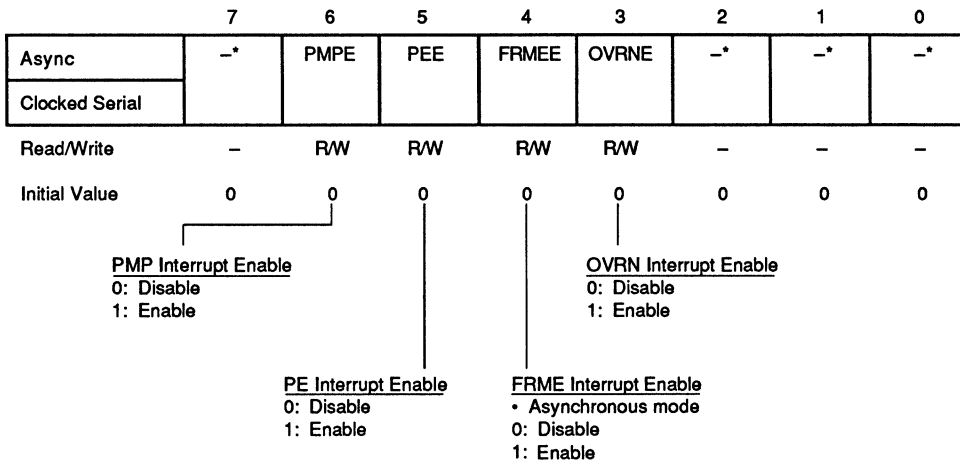
This bit is never set in the clocked serial mode.

### 5.2.15 ASCII Interrupt Enable Register 2 (IE2)

This register is used to enable or disable an RXINT internal interrupt originating in a status bit (PMP, PE, FRME, or OVRN) of ST2.

When a specific enable bit of this register and the corresponding status bit in ST2 are both 1, the RXINT bit of ST0 is set to 1. This causes an RXINT internal interrupt request to be issued to the CPU.

For details about interrupts, see section 5.6 "Internal Interrupts."



- Reserved. These bits always read 0 and should be set to 0.

**Bit 7:** Reserved. This bit always reads 0 and should be set to 0.

#### Bit 6: PMPE (PMP interrupt enable)

- Asynchronous/Clocked serial mode

PMPE	Function
0	Disables an RXINT internal interrupt originating in the PMP bit.
1	Enables an RXINT internal interrupt originating in the PMP bit. If the PMP bit in ST2 is 1, the RXINT bit in ST0 is set to 1 and an RXINT internal interrupt request is issued to the CPU.

#### Bit 5: PEE (PE interrupt enable)

- Asynchronous/Clocked serial mode

PEE	Function
0	Disables an RXINT internal interrupt originating in the PE bit.
1	Enables an RXINT internal interrupt originating in the PE bit. If the PE bit in ST2 is 1, the RXINT bit in ST0 is set to 1 and an RXINT internal interrupt request is issued to the CPU.

#### Bit 4: FRMEE (FRME interrupt enable)

- Asynchronous mode

FRMEE	Function
0	Disables an RXINT internal interrupt originating in the FRME bit
1	Enables an RXINT internal interrupt originating in the FRME bit If the FRME bit in ST2 is 1, the RXINT bit in ST0 is set to 1 and an RXINT internal interrupt request is issued to the CPU.

- Clocked serial mode

This bit is never set in clocked serial mode.

### Bit 3: OVRN (OVRN interrupt enable)

- Asynchronous/Clocked serial mode

OVRNE	Function
0	Disables an RXINT internal interrupt originating in the OVRN bit.
1	Enables an RXINT internal interrupt originating in the OVRN bit. If the OVRN bit in ST2 is 1, the RXINT bit in ST0 is set to 1 and an RXINT internal interrupt request is issued to the CPU.

**Bits 2-0:** Reserved. These bits always read 0 and should be set to 0.

### 5.2.16 ASCII TX/RX Buffer Register (TRB)

This is an 8-bit register used to write a transmit character or to read a receive character.

	7	6	5	4	3	2	1	0
Async	TRB7	TRB6	TRB5	TRB4	TRB3	TRB2	TRB1	TRB0
Clocked Serial								
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	X	X	X	X	X	X	X	X

Transmit/Receive buffer value

X: Undefined

### Bits 7-0: TRB7-0 (TX/RX buffer)

- Asynchronous/Clocked serial mode

A receive character in the receive buffer can be read via the TRB7-0 bits. If the RXRDY bit in ST0 is 0, the value of the TRB7-0 bits is undefined.

A transmit character can be loaded into the transmit buffer via the TRB7-0 bits. If the TXRDY bit in ST0 is 0, writing to the TRB7-0 bits causes current data and the buffer contents to be lost.

## 5.3 Operation

### 5.3.1 Asynchronous Mode

In the asynchronous mode, characters are synchronized by appending a start bit and stop bit(s) before transmission. The transmission line usually remains at high level (mark). The line level goes low (space) to indicate the beginning of a start bit. Data is transmitted and received as characters; figure 5-3 shows the character format.

To select the asynchronous mode, set the PRTCL2-0 bits (bits 7-5 in MD0) to 000. To select the receive character length (7 or 8 bits), use MD1.

The ASCII functions as a subset of the MSC1 when both are operated in the asynchronous mode.\*

The start bit signals the beginning of a data transfer and is followed by character data beginning with the LSB. A parity/MP bit can be appended or omitted. MD1 is used to select parity/MP bit.

Parity/MP bit can be selected as: MP bit, even/odd parity, or no parity/MP bit.

The end of the data transfer is signalled by 1 or 2 stop bits. MD0 is used to select the number of stop bits.

\* When using the MSC1 and ASCII/CSIO together in the asynchronous mode, take note of the following points regarding the ASCII:

- The transmit / receive character length is only 7 or 8 bits (5, 6, 7 or 8 bits for the MSC1).
- The stop bit length is only 1 or 2 bits (1, 1.5, or 2 bits for the MSC1).
- The ASCII baud rate generator setting is common to the transmitter and receiver.
- The ASCII transmit and receive buffers have only one stage (double buffer).

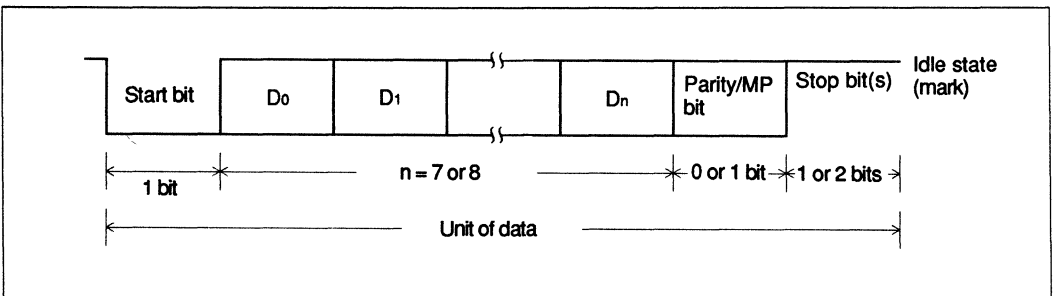
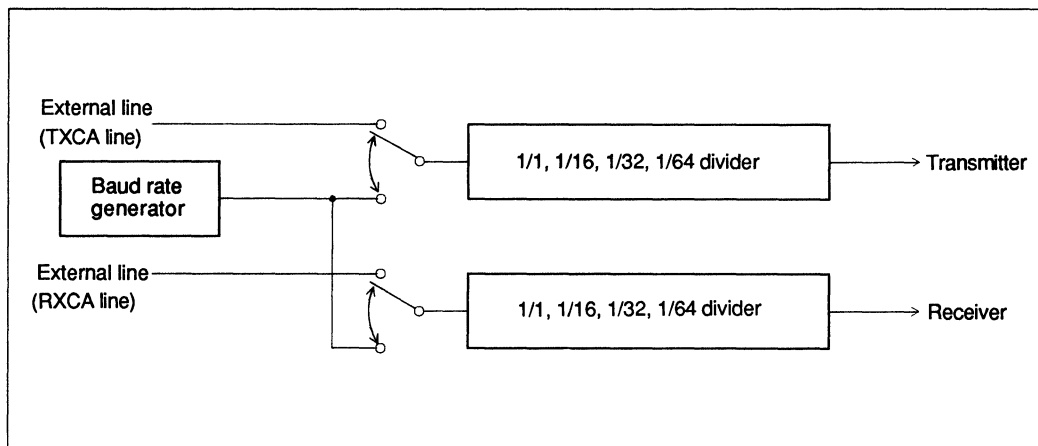


Figure 5-3. Character Format in Asynchronous Mode

The bit rate can be selected as either 1/1, 1/16, 1/32, or 1/64 of the input clock. Asynchronous communication can be performed in 1/16, 1/32 or 1/64 of the input clock (see figure 5-4). The MD1 BRATE1-0 bits are used to specify the bit rate. Either an external clock or an internal baud rate generator can be selected as the I/O clock.



**Figure 5-4. Bit Rate Selection**

For more information about the baud rate generator, see section 5.5 "Baud Rate Generator."

**Transmission operation:** Figure 5-5 shows the state transition diagram for asynchronous mode transmission.

- **TX disable state**  
The transmitter is placed in the TX disable state by a hardware reset, a channel reset command, a TX reset command, or by a TX disable command.  
In this state, the TXDA line remains high and the TXRDY bit in ST0 is cleared.
- **Idle state**  
A TX enable command causes the transmitter to leave the TX disable state and enter the idle state.  
In the idle state, the TXDA line stays high until data is loaded into the transmit buffer. Once data is loaded, the transmitter enters the start bit transmit state.
- **Start bit transmit state**  
The TXDA line goes low for one bit cycle, then enters the character transmit state.
- **Character transmit state**  
The character in the transmit buffer is transmitted beginning with the LSB.



- **Parity/MP bit transmit state**  
A parity or MP bit is transmitted as specified by the PMPM1-0 bits of MD1.
- **Stop bit transmit state**  
A stop bit(s) is transmitted as specified by the STOP1-0 bits of MD0, then the transmitter returns to the idle state.
- **Break send state**  
Setting the BRK bit in CTL causes the TXDA line to go low. Clearing the BRK bit causes the transmitter to leave this state.
- **One cycle mark transmit state**  
The TXDA line goes high for one bit cycle after leaving the break send state.

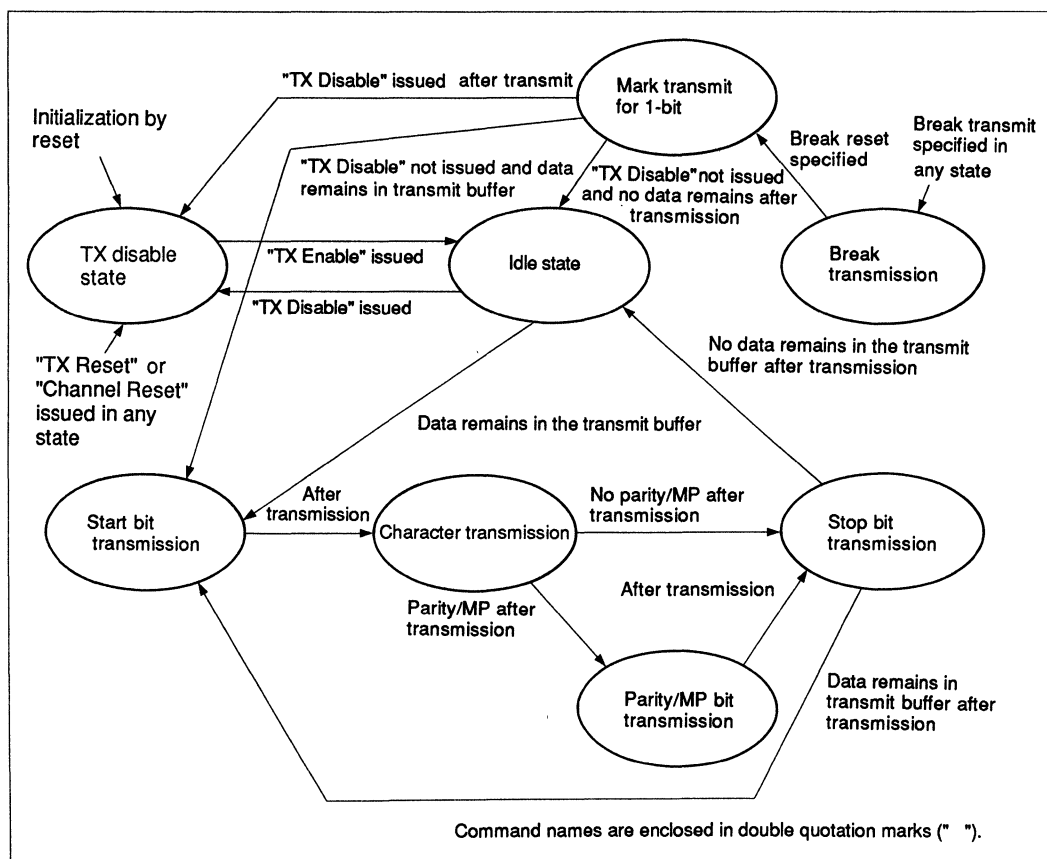


Figure 5-5. State Transition Diagram for Asynchronous Mode Transmission

The transmission cycle begins when the transmitter is in the idle state and when a character is loaded into the transmit buffer. The level of the transmit signal is changed at the falling edge of the transmit clock (see figure 5-6). For figure 5-6 (a) and (b), the character length is 8 bits, parity is used, and the stop bit length is 1 bit.

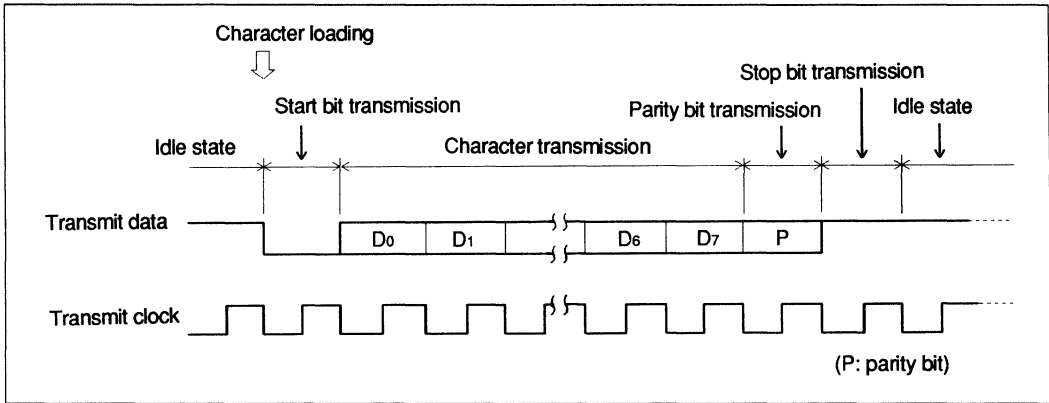


Figure 5-6. (a) Transmit Operation Using 1/1 Clock

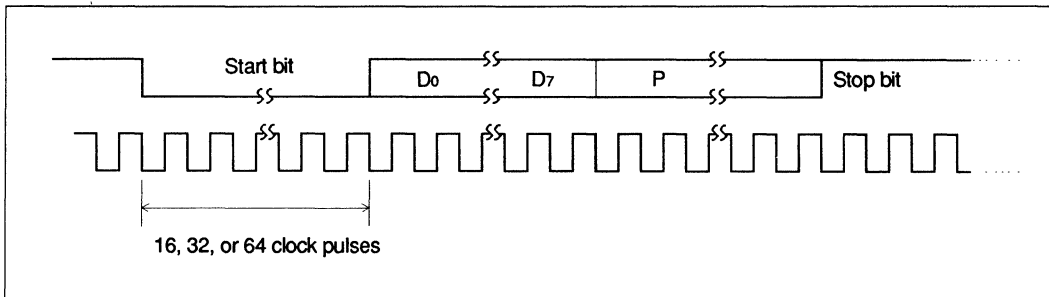


Figure 5-6. (b) Transmit Operation Using 1/16, 1/32, or 1/64 Clock

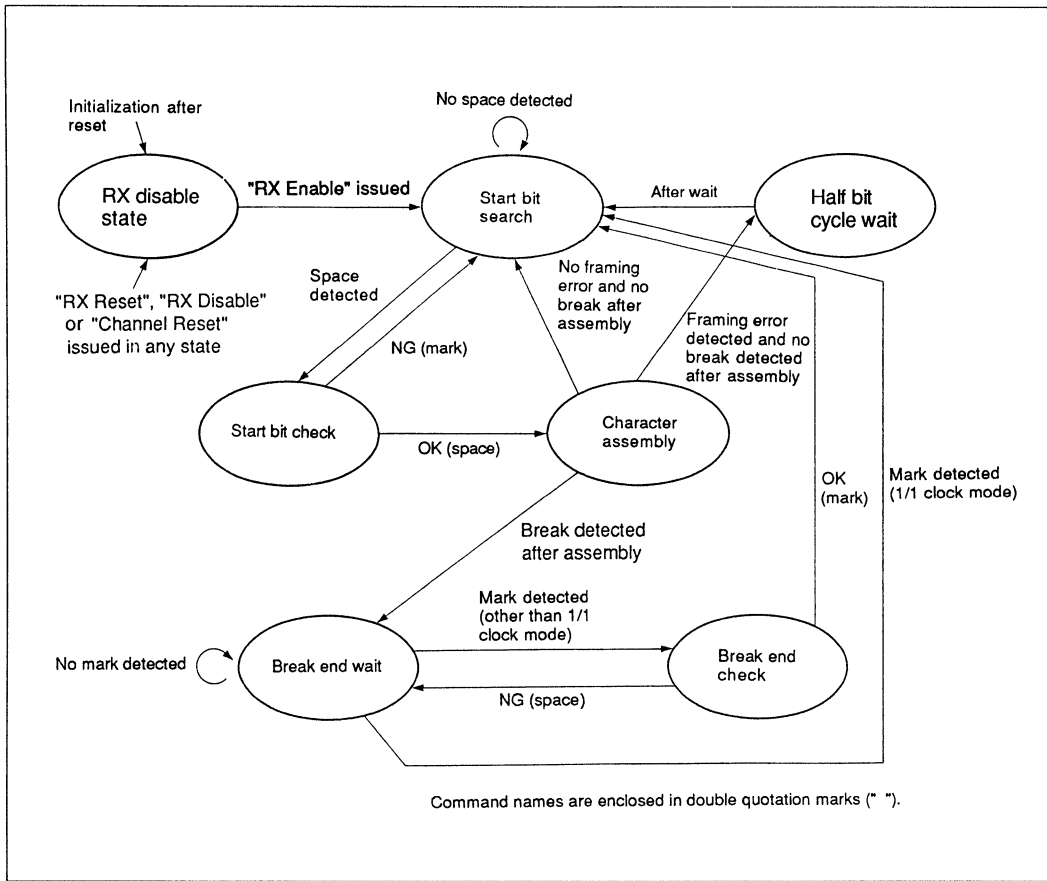
**Receive operation:** Figure 5-7 shows the state transition diagram for asynchronous reception.

- **RX disable state**

The receiver is placed in the RX disable state by a hardware reset, a channel reset command, an RX reset, or by an RX disable command. In this state, the RXDA line level is ignored and no reception operations occur. The contents of the receive shift register are lost, but the RX buffer is not affected.

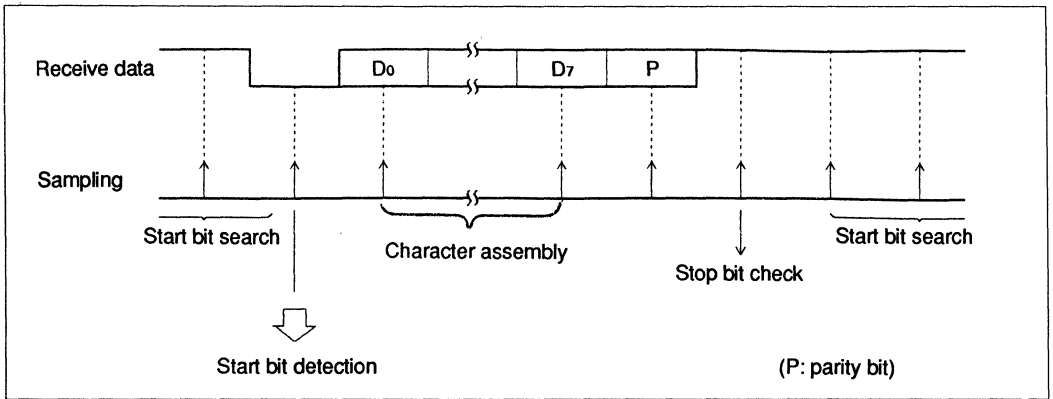
- **Start bit search state**  
An RX enable command causes the receiver to leave the RX disable state and enter the start bit search state. In this state, the level of the RXDA line is sampled at the leading edge of each receive clock cycle until a low sample (space) is detected.
- **Start bit check state**  
When a space is detected in the start bit search state, the receiver enters the start bit check state. After a delay of a half bit cycle, the RXDA line is sampled again. If a high level (mark) is detected, the receiver returns to the start bit search state. If a low level (space) is detected, the receiver enters the character assembly state.  
In the 1/1 clock mode, this state is skipped and the receiver enters the character assembly state directly.
- **Character assembly state**  
The receive data is sampled every bit cycle and the character is assembled. Character assembly ends when the first stop bit is sampled.
- **Half bit cycle wait state**  
If a framing error occurs after character assembly has been completed, the receiver enters a wait state for half a bit cycle to skip the stop bit associated with the framing error. It then returns to the start bit search state. For details on framing errors, see "Error check."
- **Break end wait state**  
If a break is detected after character assembly, the receiver enters this state. The RXDA line level is checked every clock cycle until a mark is detected. For details on break, see "Break send/detection."
- **Break end check state**  
When a mark is detected in the break end wait state, the receiver enters the break end check state. After a half-bit cycle delay, the RXDA line is checked again.\* If the line level has changed to space, the receiver returns to the break end wait state. If the RXDA line has remained at mark, the receiver enters the start bit search state.

\* In the 1/1 clock mode, this state is skipped. The receiver enters the start bit search state directly.

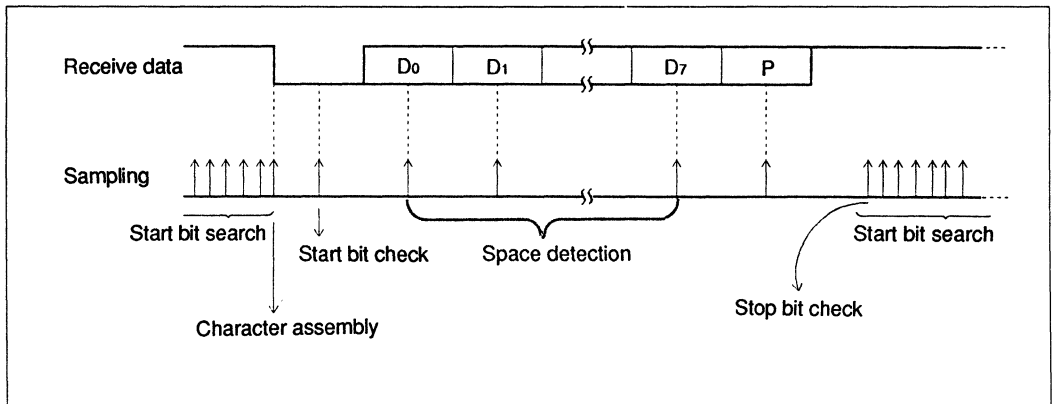


**Figure 5-7. State Transition Diagram for Asynchronous Mode Reception**

Figures 5-8 (a) and (b) show examples of receive data sampling. In these examples, the character length is 8 bits, parity is used, and the stop bit length is 1.



**Figure 5-8. (a) Receive Data Sampling Using 1/1 Clock**



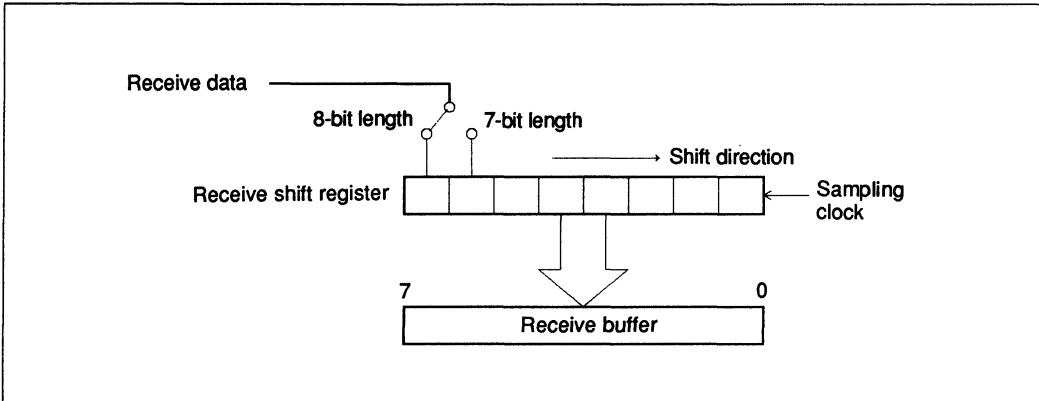
**Figure 5-8. (b) Receive Data Sampling Using 1/16, 1/32, or 1/64 Clock**

A reception operation starts when an RX enable command is issued.

When the 1/1 clock is used (figure 5-8 (a)), the receiver searches for a start bit at the leading edge of each clock. If a space is detected, character assembly starts at the next leading clock edge. Character assembling involves transferring each character bit (sampled at each clock cycle) to a receive shift register (see figure 5-9).

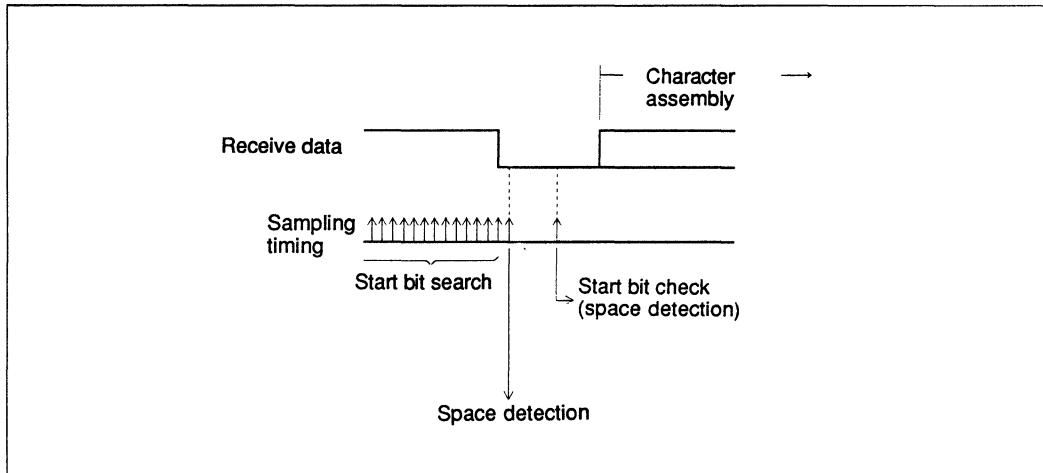
Data having the character length specified by the RXCHR1-0 bits in MD1 is transferred to the receive shift register. If it exists, the parity/MP bit is then sampled. During the next clock, the stop bit is sampled to complete the assembly of the character. At this time, the contents of the receive shift register are loaded into the receive buffer.

A search for the start bit begins one clock cycle after a character is assembled and sampling is done at the leading edge of each clock cycle.

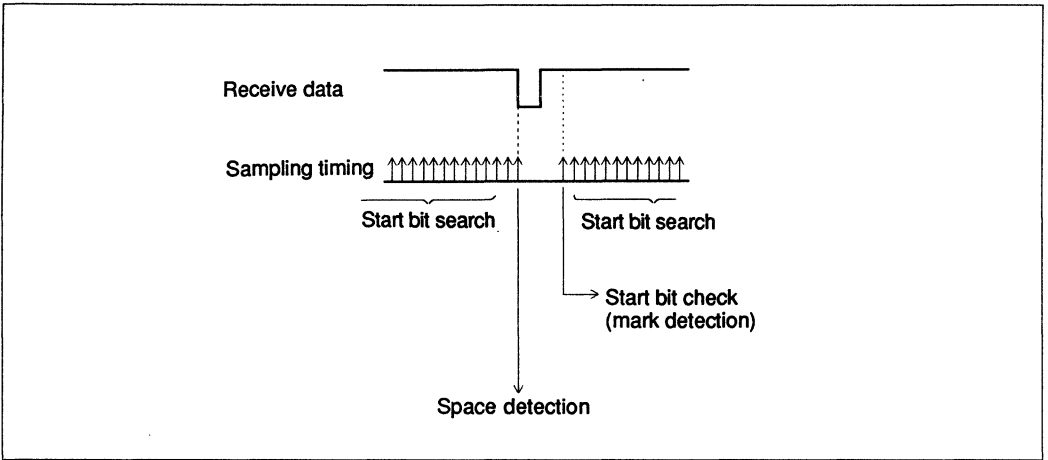


**Figure 5-9. Receive Character Assembly by Shift Register**

When using the 1/16, 1/32, or 1/64 clock mode (figure 5-8 (b)), the receiver searches for a start bit at the rising edge of each clock. When a space is detected, the receive line is sampled again after a half bit cycle delay. If another space is detected, the character assembly sequence begins after one bit cycle. If a mark is detected, start bit search resumes (See figures 5-10 (a) and (b)). This helps to prevent transmission line noise from triggering the character assembly sequence.



**Figure 5-10. (a) Start Bit Sampling (normal start bit)**

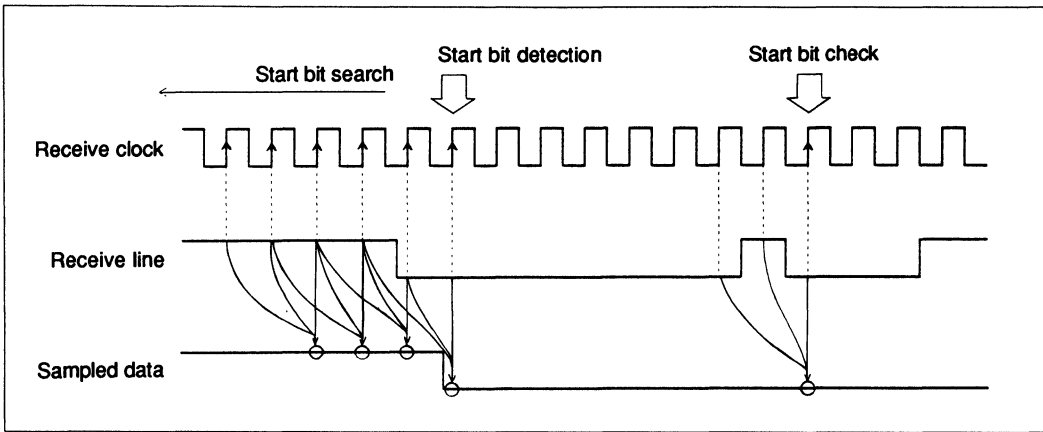


**Figure 5-10. (b) Start Bit Sampling (false start bit)**

In the character assembly state, data is sampled during every bit cycle. When the MSB or the parity bit (if it exists) is sampled, the stop bit is checked during the next bit cycle. If a mark is detected (normal), the search for the start bit begins immediately. If a space is detected (framing error), the search for the start bit resumes after a delay of a half bit cycle.

In the 1/16, 1/32, or 1/64 clock mode, the noise suppressor function operates during sampling of the start, character, parity, and stop bits.

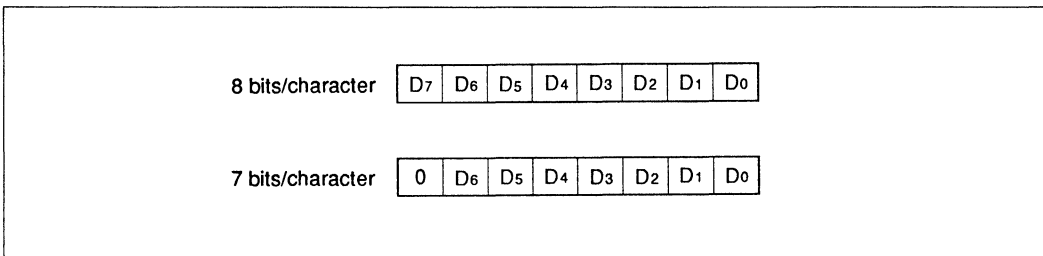
The noise suppressor function analyzes the current and two preceding values (sampled at the full clock rate) and selects the receive value using majority rule (see figure 5-11).



**Figure 5-11. Noise Suppressor Function**

In the asynchronous mode, receive character length is 7 or 8 bits. The RXCHRI-0 bits of MD1 is used to select the receive character length.

Figure 5-12 shows the receive data format. When the receive character is 7 bits, the MSB is 0.



**Figure 5-12. Receive Character**

**Parity/MP bit:** An even/odd parity bit or MP bit can be selected by setting the PMPM1-0 bits in MD1 accordingly.

When even parity is selected, the transmitter selects the parity bit so that the total number of 1s (in the character plus parity bit) is even. The receiver checks that the number of 1s received is even.

Similarly, if odd parity is selected, the parity bit is set so that the total number of 1s transmitted is odd.

When the MP bit is selected, an MP bit is appended to support multiprocessor communications. For details, see "Multiprocessor Mode."



## Error checks:

### ① Parity check

The receiver verifies that received data has the proper parity bit.

If even parity is specified and an odd number of 1s are detected in the received characters and parity bits, the PE (parity error) bit in ASCII status register 2 (ST2) is set to 1 when the receive data containing the parity error becomes ready to be read. The situation for odd parity is the same except that an even number of 1s triggers the error.

Once a parity error has occurred, even if subsequent data is normally received, the PE bit cannot be cleared until 1 is written to the PE bit by the CPU or the ASCII/CSIO is reset.

When the PE bit is set, an internal interrupt is generated (if enabled).

\* For further details about the PE bit, see section 5.2.11 "ASCII Status Register 2 (ST2)."

### ② Framing errors

A framing error occurs when a space is detected during a stop bit check.

Even if the stop bit length is 2 bits, only the first bit is checked.

When the data containing a framing error becomes ready to read, the FRME bit\*1 in ST2 is set.

A framing error does not stop reception operation. When the 1/1 clock is used, start bit scanning resumes immediately following the framing error. When the 1/16, 1/32, or 1/64 clock is used, scanning resumes half a bit cycle after the framing error. This delay allows invalid stop bit(s) to clear.

Once the FRME bit is set, it is not cleared until 1 is written to this bit position or until a reset occurs.

When the FRME bit is set, an internal interrupt is generated (if enabled).

\* For the FRME bit, see section 5.2.11 "ASCII Status Register 2 (ST2)."

### ③ Overrun errors

If the receive buffer is full when new data arrives, an overrun error occurs and the new data overwrites the data currently in the receive buffer (TRB).

The OVRN bit\* in ST2 is set when the data that caused the overrun becomes ready to read.

The OVRN bit is cleared only by writing a 1 to this bit position or by a reset.

When the OVRN bit is set, an internal interrupt is generated (if enabled).

\* For the OVRN bit, see section 5.2.11 "ASCII Status Register 2 (ST2)."

**Break send and detection:** When the transmitter wants to suspend data transmission, it sends a break signal (space).

Normally, a break send request is issued after completing a character transmission. A break must be sent for one or more character cycles.

To send a break, set the BRK bit in CTL. When this bit is set, the TXDA line goes low at the falling edge of the next transmit clock.

To cancel a break send, clear the BRK bit. When this bit is cleared, the TXDA line goes high (mark) at the falling edge of the next transmit clock. The transmitter guarantees that the line stays high (mark) for one or more bit cycles before transmitting next start bit.

When a break send is requested, data in the transmit shift register is lost, while data in the transmit buffer remains unaffected.

The receiver detects a break in the following way: if the data and parity bits are all equal to 0 and a framing error is detected, the receiver assumes that a break request has been sent. It then sets the BRKD bit in ST1 and truncates the null character (it is not transferred to the receive buffer).

The ASCI/CSIO receiver detects a break as shown in figure 5-13. If break sending starts while a character is being transmitted, the break must be sent for at least two consecutive character cycles.

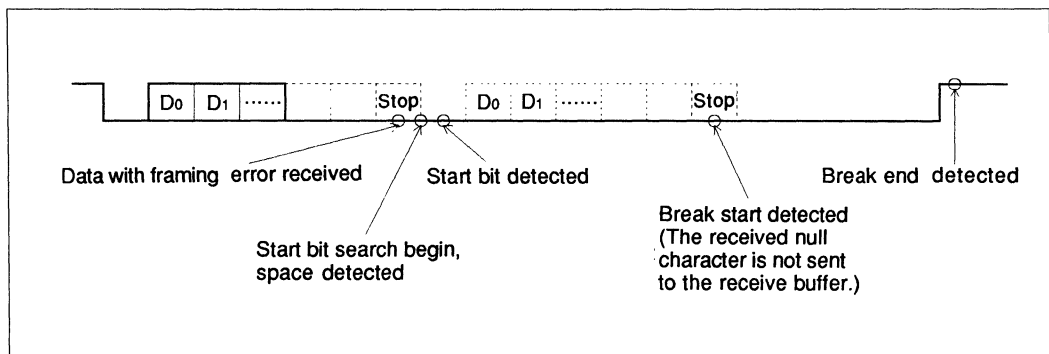


Figure 5-13. Break Detection

If marks are detected for half a bit cycle or longer,\* the receiver assumes that the break has ended and sets the BRKE bit in ST1. Both break start detection (setting the BRKD bit) and break end detection (setting the BRKE bit) generate interrupts.

\* When the 1/1 clock is used, the first mark detected signals the end of a break.

## Supplementary explanation

A break is usually sent in the following way:

- ① Wait for the end of a transmission (idle state)
- ② Set the BRK bit
- ③ Wait one or more character cycles
- ④ Clear the BRK bit

**Multiprocessor Mode:** The ASCI/CSIO supports a facility for specifying whether or not a specific terminal should receive data.

When the MP (multiprocessor) bit mode is selected, an MP bit is appended instead of a parity bit. Use the PMPM1-0 bits in MD1 to select the MP bit mode.

When the MP bit mode is selected, data is normally transmitted with the MP bit set to 0. The MP bit can be set to 1 by issuing an MP bit on command immediately before transferring the transmit data to the transmit buffer. This command only affects the first character that is loaded.

On the receive side, the MP bit is transferred to the receive buffer together with other status information. When the receive data becomes ready to read, the MP bit value is made available in ST2. Data with MP bit = 0 can be ignored (not transferred to the receive buffer) by issuing a search MP bit command. The effects of this command is invalidated when data with MP bit = 1 is received. Subsequent data is received in the normal manner. For MP bit on command and search MP bit command, see section 5.2.8 "ASCI Command Registers (CMD)."

Figure 5-14 shows how communications are arranged between multiprocessors using the MP bit.

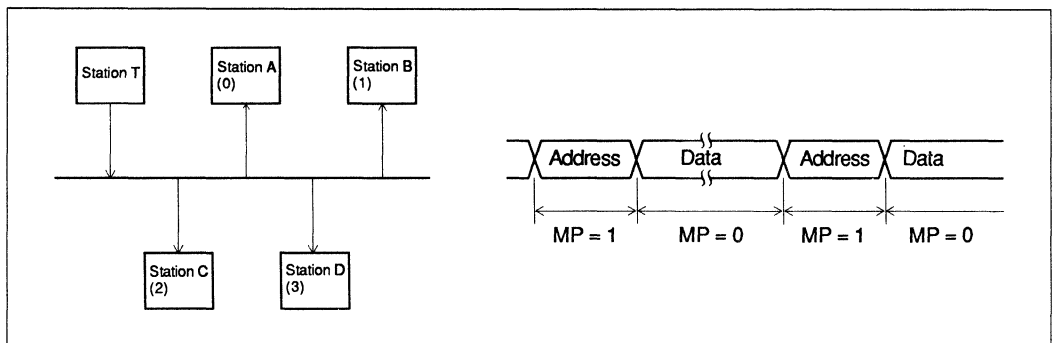


Figure 5-14. Sample MP Bit Operation

In figure 5-14, T is a transmit station and A, B, C, and D are receive stations. Stations A, B, C, and D are assigned addresses 0, 1, 2, and 3, respectively.

To transmit data from T to B, T executes an MP bit on command, then loads data representing the address of station B into the transmit buffer.

The receive stations are continually scanning the communications path. When they receive data with the MP bit = 1, they assume that the data is a station address and compare it with their own address. In this example, the received data matches the address of station B. All subsequent data with MP bit = 0 is assumed to be destined for B. The other receive stations issue a search MP bit command and ignore the data. Thus, the transmit station can transmit data to a specific receive station by transmitting the destination address with MP bit = 1 followed by data with MP bit = 0.

To communicate with a different receive station, the transmitter sends a new station address with the MP bit = 1. The transmit station can now communicate with a different station.

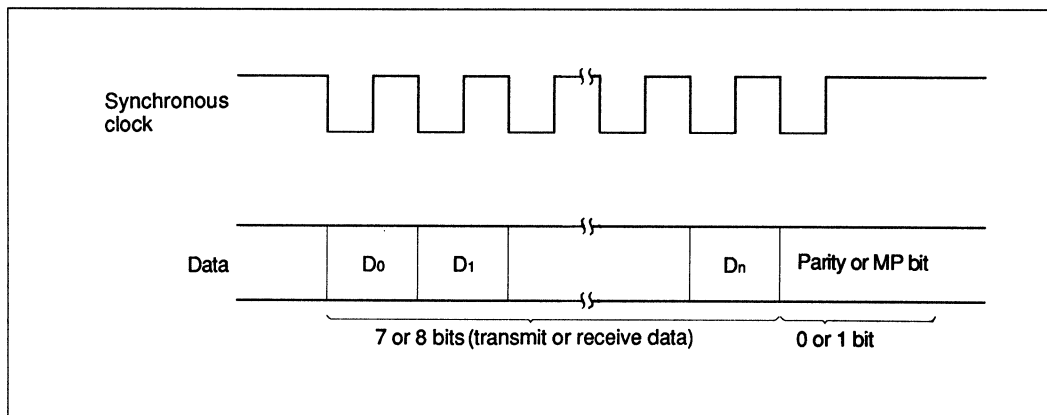
### 5.3.2 Clocked Serial Mode

In the clocked serial mode, data and clock signals are used for communications (see figure 5-15). Data is transmitted and received synchronously using the clock signals. The ASCI/CSIO can be used as the master or slave in clocked serial mode.

The clock divider is not available in this mode. When the ASCI/CSIO is the slave, the clock source is external; when the ASCI/CSIO is the master, the internal baud rate generator is used.

To select the clocked serial mode, set the PRTCL2-0 bits (bits 7-5) in MD0 to 110.

**Character format:** Figure 5-15 shows the communications character format for the clocked serial mode.



**Figure 5-15. Communications Character Format for Clocked Serial Mode**

The character length is 7 or 8 bits and a parity or MP bit is sometimes appended. Specification of the parity/MP bit must be the same for the transmitter and receiver.

The transmit data level changes at the falling edge of the synchronous clock and receive data is read at the leading edge.

**Transmit/Receive operation:** The TXCS2-0 bits (bits 6-4) in TXS specify whether the transmitter is to be used as the master or slave. For the receiver, the RXCS2-0 bits in F<sup>TX</sup> are used.

The transmitter and receiver selections can be made independently. Table 5-5 lists the possible combinations of master and slave settings. Normal reception operation is not guaranteed when the transmitter is the slave and the receiver is the master.

**Table 5-5. Availability of Master/Slave Combinations**

Transmitter	Receiver	Available/Unavailable
Master	Master	○
Master	Slave	○
Slave	Master	×
Slave	Slave	○

The master outputs the synchronous clock. The clock line is normally high except that a negative pulse is generated for each bit transmitted or received.

The slave transmits or receives data depending on the synchronous clock input from the TXCA and RXCA lines. In both the master and slave modes, the transmit data level changes at the trailing clock edge and receive data is read at the leading clock edge.

① Master mode operation

Select the internal baud rate generator as the clock source. The transmitter outputs the data and clock at the bit rate determined by the baud rate generator. The transmit clock is output on the TXCA line. This line remains high when nothing is being transmitted.

The receiver samples data according to the transmit clock. Even when the ASCI/CSIO is used only for reception, enable the transmitter and transmit dummy data as shown in figure 5-16 (2). Figure 5-16 (1) shows a sample transmission from the master to the slave. Transmission starts when data is loaded into the transmit buffer. The transmit data changes at the falling edge of the TXCA clock. The slave samples the data (input from the RXCA line) at the rising edge of the TXCA clock.

② Slave mode operation

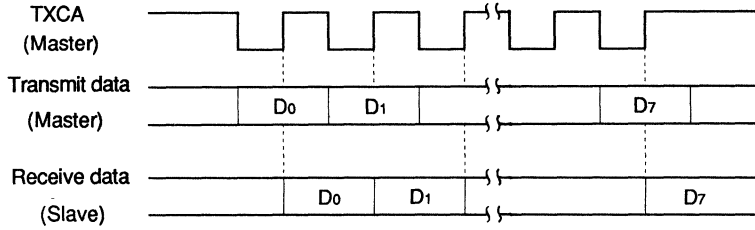
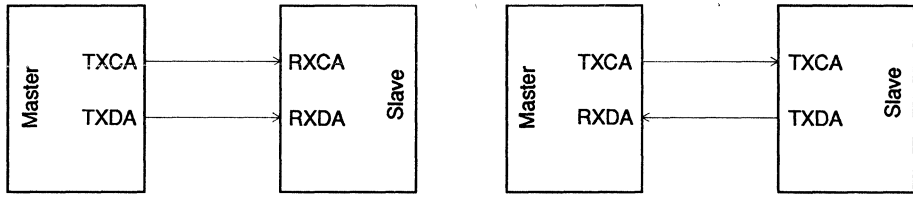
Select the external clock as the clock source. The transmitter outputs data via the TXDA line in sync with the clock input via the TXCA line. When there is no output data, the clock is ignored and the TXDA line remains at the level of the last bit transmitted.

The receiver reads data from the RXDA line in sync with the clock input via the RXCA line.

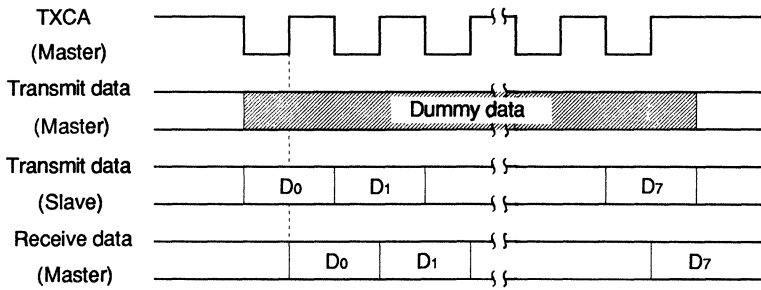
If the transmit and receive characters are not synchronized, the slave remains suspended after the end of transmission/reception.

In this case, reset the slave with a TX reset or an RX reset command.

Figure 5-16 (2) shows a sample transmission from the slave to the master. The slave loads data into the transmit buffer. Since transmission from the slave is synchronized with the clock input via the TXCA line, the master must supply the TXCA clock by transmitting dummy data. The slave transmits data using this clock. The master samples data from the slave at the leading edge of the TXCA clock. Treatment of the parity/MP bit in the clocked serial mode is the same as in the asynchronous mode. Table 5-6 shows transmission/reception operation in the master and slave modes.



(1) Transmission from Master to Slave



(2) Transmission from Slave to Master

**Figure 5-16. Sample Transmissions in Clocked Serial Mode**

**Table 5-6. Transmission/Reception Operation in Clocked Serial Mode**

<b>Item</b>			<b>Contents</b>
Slave mode	Mode setting		Select the external clock as the clock source
	TXCA line		Input
	RXCA line		Input
	Operation	Transmit	Outputs data on the TXDA line in synch with the clock input via the TXCA line. When no output data exists, the clock is ignored and the TXDA line remains at the level of the last bit transmitted.
		Receive	Reads data from the RXDA line in synch with the clock input via the RXCA line.
Master mode	Mode setting		Select the internal baud rate generator as the clock source
	TXCA line		The transmit clock is output on these lines. They remain high when nothing is being transmitted.
	RXCA line		
	Operation	Transmit	Outputs data and clock at the bit rate determined by the baud rate generator.
		Receive	Samples data in synch with the transmit clock. When performing reception only, enable the transmitter and transmit dummy data.

Note: If the transmit and receive characters are not synchronized, issue a TX or an RX reset command.

## 5.4 Transmit/Receive Clock Selection

### 5.4.1 Overview

ASCII/CSIO transmit and receive clock sources are selected from among the following sources:

- Transmit clock sources
  - TXCA line input
  - Baud rate generator output

Note: The TXCS 2-0 bits of the ASCII TX clock source register (TXS) are used to select a particular transmit clock source.



- Receive clock sources
  - RXCA line input clock
  - Baud rate generator output

Note: The RXCS 2-0 bits of the ASCII RX clock source register (RXS) are used to select a particular receive clock source.

The internal baud rate generator (BRG) outputs a clock signal produced by dividing the system clock signal both to the transmitter and receiver.

Figure 5-17 shows how the ASCII/CSIO clock is supplied for transmission and reception.

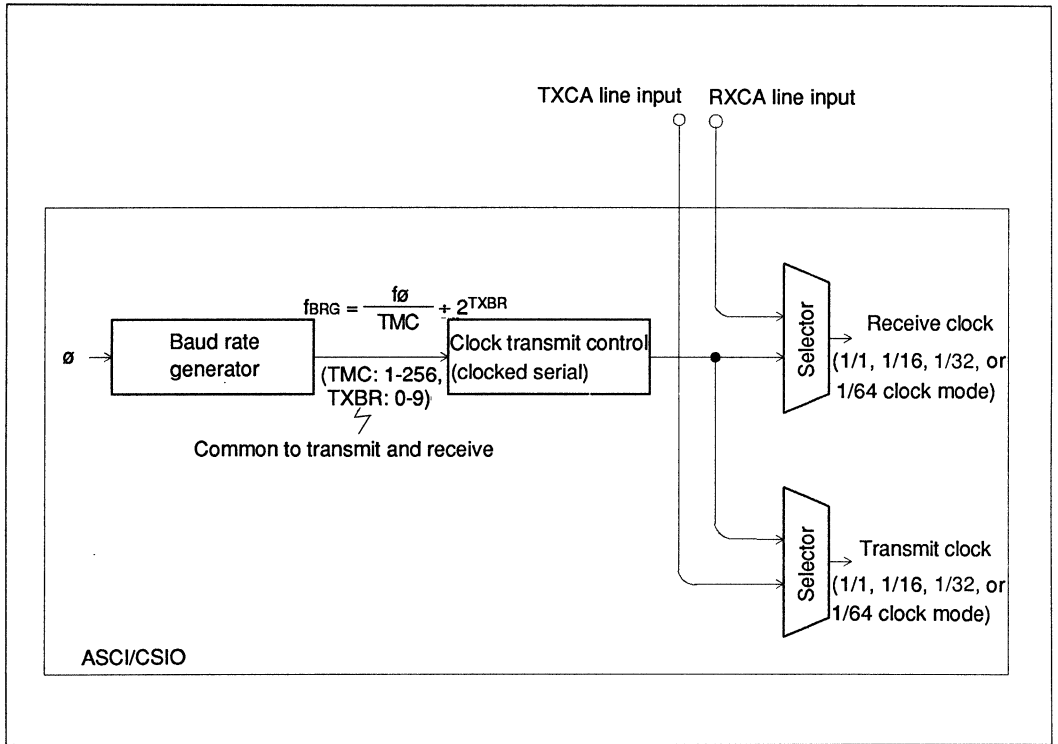


Figure 5-17. Selecting Transmit and Receive Clocks

### 5.4.2 Supplying the Transmit Clock

The baud rate generator output or TXCA line input is used as the transmit clock. When the transmitter baud rate generator output is used as the transmit clock, the TXCA line functions as a transmit clock output.

In the asynchronous mode, the actual bit rate is determined by the clock mode (1/1, 1/16, 1/32 and 1/64). In the clocked serial mode, it is determined by the 1/1 clock mode.

### 5.4.3 Supplying the Receive Clock

The baud rate generator output or RXCA line input is used as the receive clock. When the baud rate generator output is used as the receive clock, the RXCA line functions as a receive clock output.

In the asynchronous mode, the actual bit rate is determined by the clock mode (1/1, 1/16, 1/32, or 1/64), and in the clocked serial mode, it is determined by the 1/1 clock mode.

### 5.4.4 Baud Rate Generator

The output frequency of the baud rate generator for transmission and reception is obtained by the following equation (For details, see section 5.5 "Baud Rate Generator"):

$$f_{BRG} = \frac{f_{\phi}}{TMC} + 2^{TXBR}$$

where,

f <sub>BRG</sub> :	BRG output frequency
f <sub>φ</sub> :	System clock frequency
TMC:	ASCI time constant register value = 1 to 256
TXBR:	ASCI TX clock source register TXBR 3-0 bit values = 0 to 9

## 5.5 Baud Rate Generator

### 5.5.1 Overview

The ASCI/CSIO has a internal baud rate generator (BRG). Its output can be used as the transmit or receive clock. The output frequency ranges from  $f\phi/2$  to  $f\phi/2^{17}$  where,  $f\phi$  is the CPU clock frequency.

#### (1) BRG features

- Output clock frequency range:  $f\phi/2$  to  $f\phi/2^{17}$  ( $2^{17} = 131072$ ) ( $f\phi$ : CPU clock frequency)
- Frequency accuracy within  $\pm 0.5\%$  for frequencies ranging from  $f\phi/100$  to  $f\phi/2^{17}$  \*.

\* 
$$|f - f_{BRG}| \leq \frac{50}{\text{Time constant register value}}$$

Where,  $f$  is the target frequency and  $f_{BRG}$  is the BRG output frequency closest to  $f$  ( $f\phi/2 \geq f \geq f\phi/2^{17}$ ).

#### (2) BRG block diagram

The block diagram of the baud rate generator in the ASCI/CSIO is shown in figure 5-18.

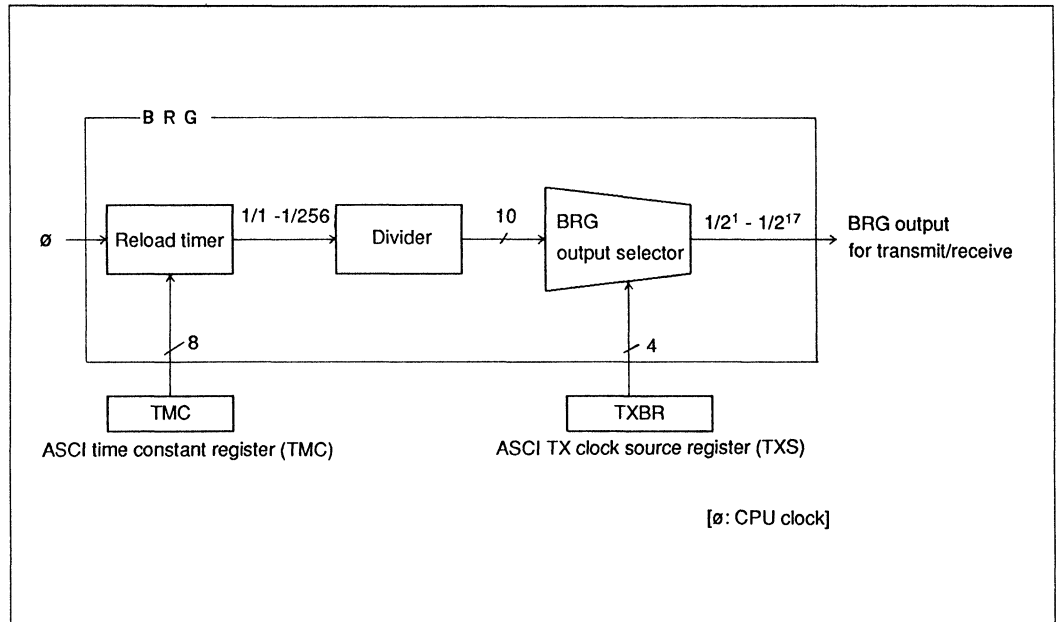
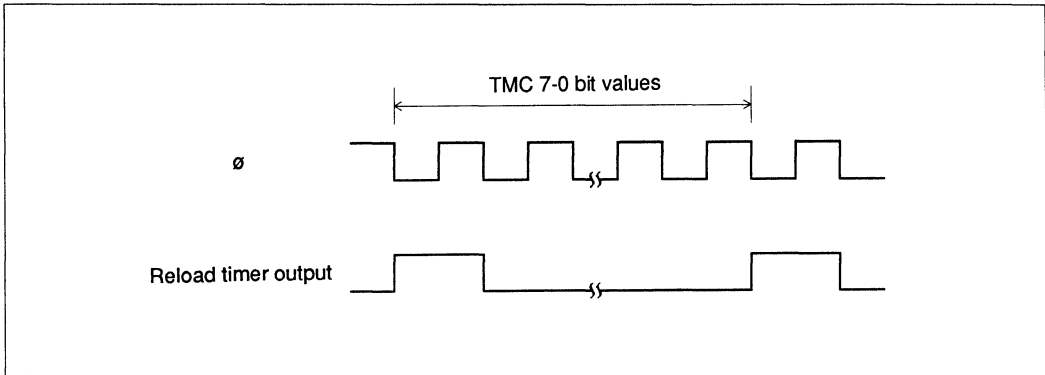


Figure 5-18. BRG Block Diagram

### 5.5.2 Functions

The clock frequency output by the BRG is determined by the contents of the TMC and TXS.

TMC is an 8-bit register for specifying the value to be loaded into the baud rate generator reload timer. The reload timer is decremented based on the CPU clock  $\phi$  and the timer outputs a high level signal during one clock cycle, each time the reload timer equals to 1. Thus, high level signals are output in units equal to the number of clock cycles specified in the 7-0 bits of TMC, as shown in the figure 5-19. When the TMC value is set to 0, it equals 256, and when the TMC value is set to 1, the line is fixed at high level.



**Figure 5-19. Reload Timer Output**

The reload timer output is input to the frequency divider. The division ratio for transmit data is specified by TXBR 3-0 bits of TXS and the division ratio for receive data is specified by RXBR 3-0 bits of MRSX.

The relationship between the register set values and generated clock frequency is given below.

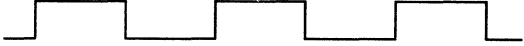

$$f_{BRG} = \frac{f_{\phi}}{TMC} + 2^{TXBR}$$

where,

- |   |   |   |
|---|---|---|
| { | <p><math>f_{BRG}</math>: BRG output frequency for transmit/receive (<math>f_{BRG} = f_{\phi}</math> cannot be used.)</p> <p><math>f_{\phi}</math>: CPU clock frequency</p> <p>TMC: Value from the ASCI time constant register (1 to 256)</p> <p>TXBR: Value from the ASCI TX clock source register (0 to 9)</p> | } |
|---|---|---|

Table 5-7 lists register set values and output clock waveforms.

**Table 5-7. BRG Output Waveform and Register Set Values**

TXBR Set Value	Waveform
1 – 9	Duty cycle = 50% 
TMC ≠ 1	High level for 1 CPU clock cycle 
0	Duty cycle = 50% when TMC = 2 Duty cycle ≠ 50% when TMC > 2
TMC = 1	Always high (cannot be used as clock)

TXBR: Value of bits 3-0 of the ASCII TX clock source register

TMC: Value of bits 7-0 of the ASCII time constant register

### 5.5.3 Register Set Values and Bit Rates

Table 5-8 gives sample settings for frequencies and bit rates in the asynchronous mode. Table 5-9 gives settings for the clocked serial mode.

(1) Asynchronous mode

**Table 5-8. Register Values and Corresponding Bit Rates in Asynchronous Mode**

Bit Rate (bps)	f $\phi$ (MHz)							
	1.7898				2.4576			
	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	–	–	–	–	1	1	1/32	0.00
19200	–	–	–	–	1	1	1/64	0.00
9600	–	–	–	–	1	2	1/64	0.00
4800	–	–	–	–	1	3	1/64	0.00
2400	47	0	1/16	–0.83	1	4	1/64	0.00
1200	93	0	1/16	–0.25	1	5	1/64	0.00
600	93	0	1/32	–0.25	1	6	1/64	0.00
300	93	0	1/64	–0.25	1	7	1/64	0.00
150	93	1	1/64	–0.25	1	8	1/64	0.00
110	127	1	1/64	0.10	175	1	1/64	–0.25

Bit Rate (bps)	f $\phi$ (MHz)							
	3.072				4			
	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	5	0	1/16	0.00	–	–	–	–
19200	5	0	1/32	0.00	13	0	1/16	0.16
9600	5	0	1/64	0.00	13	0	1/32	0.16
4800	5	1	1/64	0.00	13	0	1/64	0.16
2400	5	2	1/64	0.00	13	1	1/64	0.16
1200	5	3	1/64	0.00	13	2	1/64	0.16
600	5	4	1/64	0.00	13	3	1/64	0.16
300	5	5	1/64	0.00	13	4	1/64	0.16
150	5	6	1/64	0.00	13	5	1/64	0.16
110	109	2	1/64	0.08	71	3	1/64	0.03

- TMC: Value of the TMC7-0 bits in the ASCII time constant register  
 BR: Value of the TXBR3-0 bits in the ASCII TX clock source register  
 CM: Clock mode in the asynchronous mode (bit rate/clock rate)

**Table 5-8. Register Values and Corresponding Bit Rates in Asynchronous Mode (cont.)**

Bit Rate (bps)	$f_{\theta}$ (MHz)							
	4.608				4.9152			
	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	–	–	–	–	1	1	1/64	0.00
19200	15	0	1/16	0.00	1	2	1/64	0.00
9600	15	0	1/32	0.00	1	3	1/64	0.00
4800	15	0	1/64	0.00	1	4	1/64	0.00
2400	15	1	1/64	0.00	1	5	1/64	0.00
1200	15	2	1/64	0.00	1	6	1/64	0.00
600	15	3	1/64	0.00	1	7	1/64	0.00
300	15	4	1/64	0.00	1	8	1/64	0.00
150	15	5	1/64	0.00	1	9	1/64	0.00
110	41	4	1/64	–0.22	175	2	1/64	–0.25

Bit Rate (bps)	$f_{\theta}$ (MHz)							
	6				6.144			
	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	–	–	–	–	5	0	1/32	0.00
19200	–	–	–	–	5	0	1/64	0.00
9600	39	0	1/16	0.16	5	1	1/64	0.00
4800	39	0	1/32	0.16	5	2	1/64	0.00
2400	39	0	1/64	0.16	5	3	1/64	0.00
1200	39	1	1/64	0.16	5	4	1/64	0.00
600	39	2	1/64	0.16	5	5	1/64	0.00
300	39	3	1/64	0.16	5	6	1/64	0.00
150	39	4	1/64	0.16	5	7	1/64	0.00
110	213	2	1/64	0.03	109	3	1/64	0.08

- TMC: Value of the TMC7-0 bits in the ASCII time constant register
- BR: Value of the TXBR3-0 bits in the ASCII TX clock source register
- CM: Clock mode in the asynchronous mode (bit rate/clock rate)

**Table 5-8. Register Values and Corresponding Bit Rates in Asynchronous Mode (cont.)**

Bit Rate (bps)	f $\phi$ (MHz)							
	8				9.216			
	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	13	0	1/16	0.16	15	0	1/16	0.00
19200	13	0	1/32	0.16	15	0	1/32	0.00
9600	13	0	1/64	0.16	15	0	1/64	0.00
4800	13	1	1/64	0.16	15	1	1/64	0.00
2400	13	2	1/64	0.16	15	2	1/64	0.00
1200	13	3	1/64	0.16	15	3	1/64	0.00
600	13	4	1/64	0.16	15	4	1/64	0.00
300	13	5	1/64	0.16	15	5	1/64	0.00
150	13	6	1/64	0.16	15	6	1/64	0.00
110	71	4	1/64	0.03	41	5	1/64	-0.22

Bit Rate (bps)	f $\phi$ (MHz)							
	9.8304				10			
	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	2	1	1/64	0.00	-	-	-	-
19200	2	2	1/64	0.00	-	-	-	-
9600	2	3	1/64	0.00	65	0	1/16	0.16
4800	2	4	1/64	0.00	65	0	1/32	0.16
2400	2	5	1/64	0.00	65	0	1/64	0.16
1200	2	6	1/64	0.00	65	1	1/64	0.16
600	2	7	1/64	0.00	65	2	1/64	0.16
300	2	8	1/64	0.00	65	3	1/64	0.16
150	2	9	1/64	0.00	65	4	1/64	0.16
110	175	3	1/64	-0.25	89	4	1/64	-0.25

- TMC: Value of the TMC7-0 bits in the ASCII time constant register
- BR: Value of the TXBR3-0 bits in the ASCII TX clock source register
- CM: Clock mode in the asynchronous mode (bit rate/clock rate)



**Table 5-8. Register Values and Corresponding Bit Rates in Asynchronous Mode (cont.)**

Bit Rate (bps)	f $\phi$ (MHz)			
	12*			
	TMC	BR	CM	Deviation (%)
38400	–	–	–	–
19200	39	0	1/16	0.16
9600	39	0	1/32	0.16
4800	39	0	1/64	0.16
2400	39	1	1/64	0.16
1200	39	2	1/64	0.16
600	39	3	1/64	0.16
300	39	4	1/64	0.16
150	39	5	1/64	0.16
110	213	3	1/64	0.03

TMC: Value of the TMC7-0 bits in the ASCII time constant register

BR: Value of the TXBR3-0 bits in the ASCII TX clock source register

CM: Clock mode in the asynchronous mode (bit rate/clock rate)

\* f $\phi$  = 12 MHz is used as an example.

## (2) Clocked serial mode

**Table 5-9. Register Set Values and Bit Rates (clocked serial mode)**

Bit Rate (bps)	$f\phi$ (MHz)								
	2.4576			3.072			4		
	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)
38400	32	1	0.00	40	1	0.00	52	1	0.16
19200	32	2	0.00	40	2	0.00	52	2	0.16
9600	32	3	0.00	40	3	0.00	52	3	0.16
4800	32	4	0.00	40	4	0.00	52	4	0.16
2400	32	5	0.00	40	5	0.00	52	5	0.16
1200	32	6	0.00	40	6	0.00	52	6	0.16
600	32	7	0.00	40	7	0.00	52	7	0.16
300	32	8	0.00	40	8	0.00	52	8	0.16

Bit Rate (bps)	$f\phi$ (MHz)								
	4.608			4.9152			6		
	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)
38400	60	1	0.00	64	1	0.00	78	1	0.16
19200	60	2	0.00	64	2	0.00	78	2	0.16
9600	60	3	0.00	64	3	0.00	78	3	0.16
4800	60	4	0.00	64	4	0.00	78	4	0.16
2400	60	5	0.00	64	5	0.00	78	5	0.16
1200	60	6	0.00	64	6	0.00	78	6	0.16
600	60	7	0.00	64	7	0.00	78	7	0.16
300	60	8	0.00	64	8	0.00	78	8	0.16

Bit Rate (bps)	$f\phi$ (MHz)								
	6.144			8			9.216		
	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)
38400	80	1	0.00	104	1	0.16	120	1	0.00
19200	80	2	0.00	104	2	0.16	120	2	0.00
9600	80	3	0.00	104	3	0.16	120	3	0.00
4800	80	4	0.00	104	4	0.16	120	4	0.00
2400	80	5	0.00	104	5	0.16	120	5	0.00
1200	80	6	0.00	104	6	0.16	120	6	0.00
600	80	7	0.00	104	7	0.16	120	7	0.00
300	80	8	0.00	104	8	0.16	120	8	0.00

TMC: Value of the TMC7-0 bits in the ASCII time constant register

BR: Value of the TXBR3-0 bits in the ASCII TX clock source register

**Table 5-9. Register Set Values and Bit Rates (clocked serial mode) (cont.)**

Bit Rate (bps)	$f\phi$ (MHz)								
	9.8304			10			12*		
	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)	TMC	BR	Deviation(%)
38400	128	1	0.00	130	1	0.16	156	1	0.16
19200	128	2	0.00	130	2	0.16	156	2	0.16
9600	128	3	0.00	130	3	0.16	156	3	0.16
4800	128	4	0.00	130	4	0.16	156	4	0.16
2400	128	5	0.00	130	5	0.16	156	5	0.16
1200	128	6	0.00	130	6	0.16	156	6	0.16
600	128	7	0.00	130	7	0.16	156	7	0.16
300	128	8	0.00	130	8	0.16	156	8	0.16

TMC: Value of the TMC7-0 bits in the ASCII time constant register

BR: Value of the TXBR3-0 bits in the ASCII TX clock source register.

\*  $f\phi = 12$  MHz is used as an example.

## 5.6 Internal Interrupts

### 5.6.1 Interrupt Types and Sources

The ASCII/CSIO can issue four types of interrupt requests: TXRDY, RXRDY, TXINT, and RXINT.

These interrupts originate in the status bits (bits 7, 6, 1, and 0) of ST0 and are enabled/disabled by the enable bits (bits 7, 6, 1, and 0) of IE0.

The TXINT and RXINT internal interrupts are also assigned status bits for individual sources and corresponding enable bits. A status bit and its enable bit are ANDed for each interrupt source. The interrupt sources are indicated by the TXINT bit (bit 7) or RXINT bit (bit 6) in ST0 regardless of the TXINTE bit (bit 7) or RXINTE bit (bit 6) in the IE0.

## 5.6.2 Interrupt Clear

An interrupt can be cleared in the following way:

### (1) TXRDY interrupt

Transfer data into the transmit buffer or disable the transmitter. This interrupt can also be cleared by a channel reset or a TX reset command.

### (2) RXRDY interrupt

Read data from the receive buffer.

This interrupt can also be cleared by a channel reset or an RX reset command.

### (3) TXINT interrupt

When the interrupt source is TX idle, change the state by loading transmit data, etc.

When the interrupt source is a  $\overline{\text{CTS}}$ A line level change, write 1 in the CCTS bit position (bit 3) in ST1.

### (4) RXINT interrupt

When the interrupt source is a  $\overline{\text{DCDA}}$  line level change, break start or break end detection, parity error, framing error, or an overrun error, write 1 to the corresponding status bit position.

When the interrupt source is a parity/MP error, write 1 to the PMP bit (bit 6) position in ST2 or read the receive data. When the data is read, a character with the parity/MP bit = 0 is received, and the RXRDY bit goes to 1 (enabling the next data read), the interrupt is cleared.

Table 5-10 lists ASCII internal interrupts.

**Table 5-10. Internal Interrupts, Sources, and Clear Procedures**

Interrupt Type	Interrupt Status Bit	Enable Bit	Source	Source		Clear Procedure*1
				Status Bit	Enable Bit	
TXRDY interrupt	TXRDY	TXRDYE	TX ready	–	–	Load transmit data to fill the transmit buffer or disable the transmitter
RXRDY interrupt	RXRDY	RXRDE	RX ready	–	–	Read receive data to empty the receive buffer
TXINT interrupt	TXINT	TXINTE	1 Transmitter idle state	IDL	IDLE	Change the state
			2 CTS $\bar{A}$ line level change	CCTS	CCTSE	Write 1 to the status bit position
RXINT interrupt	RXINT	RXINTE	1 DCDA line level change	CDCD	CDCDE	1-7: Write 1 in the status bit
			2 Break start detected	BRKD	BRKDE	4: Read the receive data *2
			3 Break end detected	BRKE	BRKEE	
			4 Receive data with parity/MP bit = 1	PMP	PMPE	
			5 Parity error	PE	PEE	
			6 Framing error	FRME	FRMEE	
			7 Overrun error	OVRN	OVRNE	

\*1 TXRDY and TXINT interrupts can also be cleared by a channel or TX reset command. RXRDY and RXINT interrupts can also be cleared by a channel or RX reset command.

\*2 This source is cleared if the parity/MP bit of the next character is 0. The interrupt is cleared when the RXRDY bit goes to 1 (enabling the next data read) after reading the current data.

### 5.6.3 Interrupt Request Conditions

The conditions for the TXRDY interrupt (TXRDY), RXRDY interrupt (RXRDY), TXINT interrupt (TXINT), and RXINT interrupt (RXINT) are listed below.

(1) TXRDY interrupt request condition

TXRDY: TXRDY•TXRDYE

(2) RXRDY interrupt request condition

RXRDY: RXRDY•RXRDYE

(3) TXINT interrupt request condition

TXINT: TXINT•TXINTE

TXINT = IDL•IDLE + CCTS•CCTSE

(4) RXINT interrupt request condition

RXINT: RXINT•RXINTE

RXINT = CDCD•CDCDE

+ BRKD•BRKDE

+ BRKE•BRKEE

+ PMP•PMPE

+ PE•PEE

+ FRME•FRMEE

+ OVRN•OVRNE

Figure 5-20 shows the relationship between the interrupts, status, and enable bits.

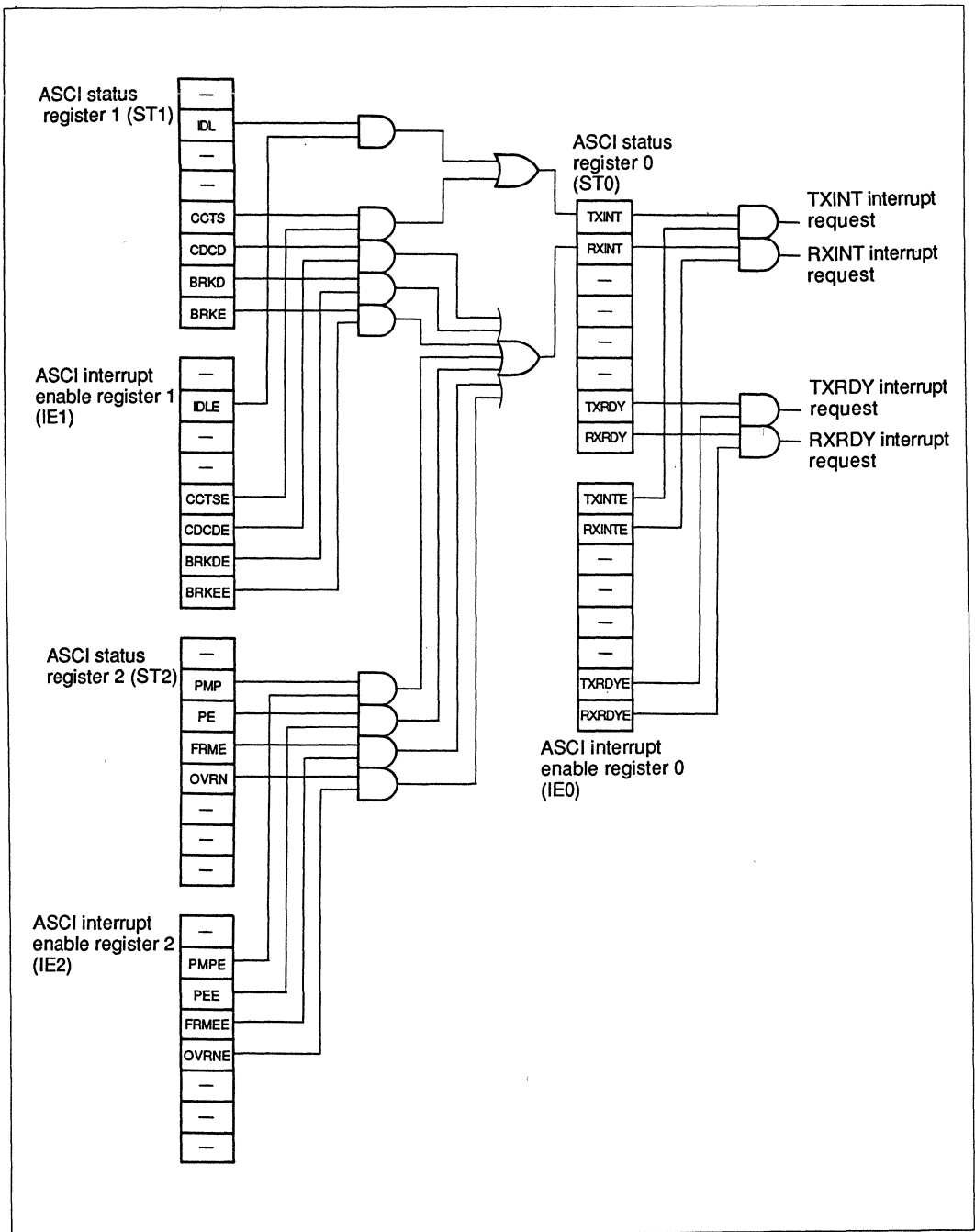


Figure 5-20. Relationship Between Interrupts, Status, and Enable Bits

## 5.7 Application Examples

### 5.7.1 Serial Data Transfer by the CPU

#### (1) Transferring Transmit Data

- Polling

The CPU determines the transfer timing by monitoring the TXRDY bit of ST0. The TXRDY interrupt must be disabled.

- Interrupt

The CPU transfers data when a TXRDY interrupt arrives. A TXRDY interrupt request (if enabled) is issued repeatedly while the TXRDY bit (bit 1) in ST0 is equal to 1. Interrupts stop when the TXRDY bit goes to 0.

#### (2) Transferring Receive Data

- Polling

The CPU determines the data transfer timing by monitoring the RXRDY bit (bit 0) in ST0. The RXRDY interrupt must be disabled.

- Interrupt

The CPU transfers data when an RXRDY interrupt arrives. An RXRDY interrupt request (if enabled) is issued repeatedly while the RXRDY bit (bit 0) of ST0 is equal to 1. Interrupts stop when the RXRDY bit goes to 0.

### 5.7.2 Maximum Bit Rates

Table 5-11 lists the expressions used to calculate the maximum bit rates for the ASCI/CSIO. When these bit rates are exceeded, correct operation can not be guaranteed.



**Table 5-11. Expressions to Calculate Maximum Bit Rates**

Protocol Mode	Clock Mode	Expression to Calculate Maximum Bit Rate	
		External Clock	Internal BRG
Asynchronous	1/64	$f \phi + 160$	$f \phi + 128$
	1/32	$f \phi + 80$	$f \phi + 64$
	1/16	$f \phi + 40$	$f \phi + 32$
	1/1	$f \phi + 2.5^*$	$f \phi + 2$
Clocked serial	1/1	$f \phi + 2.5^*$	$f \phi + 2$

( $f\phi$ : System clock frequency)

For example, in the asynchronous mode if an external clock input is used, the clock mode is 1/32, and the system clock frequency is 10 MHz, the maximum bit rate is:

$$10 \text{ MHz} + 80 = 125 \text{ kbps}$$

- \* Values obtained by the expressions listed in table 5-11 are the maximum frequencies applicable to the transmitter and receiver. The maximum receive bit rates are exactly the same as these values, but the maximum transmit bit rates actually used are less than the calculated values depending on transmission data timing.

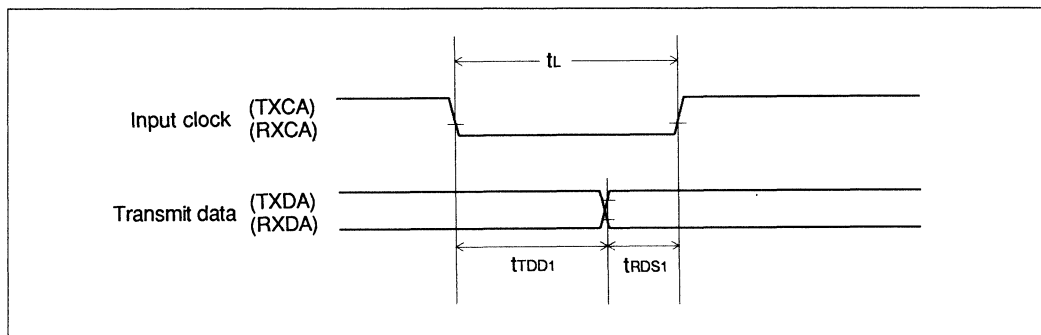
In the asynchronous 1/1 clock mode or in the clocked serial mode, transmit data is valid after a delay of  $t_{\text{TDD1}}$  after the falling edge of the input clock as shown in figure 5-21. The receiver samples data at the rising edge of the input clock.

The minimum low level period of the input clock is:

$$t_L = t_{\text{TDD1}} + t_{\text{RDS1}}$$

where,  $t_{\text{RDS1}}$  is the receive set-up time.

The maximum bit rate is the frequency that satisfies this low level condition.



**Figure 5-21. Input Clock and Transmit Data**

For example, if  $t_{rDD1} = 310$  ns and  $t_{rDS1} = 90$  ns, the clock low level time is:

$$t_L = 310 \text{ ns} + 90 \text{ ns} = 400 \text{ ns}$$

Assuming that the duty of the clock having this low level time is 50%, the clock period is:

$$400 \text{ ns} + 400 \text{ ns} = 800 \text{ ns}$$

Therefore, the maximum bit rate is:

$$\frac{1}{800 \text{ ns}} = 1.25 \text{ Mbps}$$

## 5.8 Generating MSCI-Compatible Programs

### 5.8.1 MSCI Compatibility

The ASCII/CSIO functions as a subset of the MSCI when both are operated in the asynchronous mode. Limitations on compatibility are outlined below.

- (1) The baud rate generator setting is common to transmit and receive.
- (2) The transmit/receive character length is 7 or 8 bits.
- (3) The stop bit length is 1 or 2 bits.
- (4) The transmit and receive buffers have only one stage.

### 5.8.2 Precautions for Generating MSCI-Compatible Programs

To write programs that are compatible with the ASCII/CSIO and MSCI, observe the following precautions:

- (1) The four low-order bits of the ASCII RX clock source register (RXS) must be the same as the four low-order bits of the ASCII TX clock source register (TXS). Write 0s to the unused bits.
- (2) The transmit/receive character length must be 7 or 8 bits.
- (3) The stop bit length must be 1 or 2 bits.
- (4) Write transmit data in the transmit buffer when the TXRDY bits (bit 1) in ASCII ST0 and MSCI MST0 are equal to 1. Read receive data from the receive buffer when the RXRDY bits (bit 0) in ASCII ST0 and MSCI MST0 are both 1.

## 5.9 Reset Operation

The ASCI/CSIO reset conditions are as follows:

- (1) The receiver and transmitter are disabled, and the transmit/receive buffers are cleared.
- (2) The input/output lines (RXCA and TXCA) function as inputs, and the output lines (TXDA and  $\overline{\text{RSTA}}$ ) are placed in the inactive state.
- (3) All the internal registers are reset, and the following operations are selected.
  - Asynchronous mode (stop bit length of 1, character length of 8 bits, 1/1 clock rate, no parity) is entered.
  - Full duplex communication is selected.
  - The transmit/receive status bits and interrupt enable bits are all cleared to 0.
  - The TXCA line input is used as the transmit clock and the RXCA line input is used as the receive clock.
  - The baud rate generator is initialized.

## Section 6. Direct Memory Access Controller (DMAC)

### 6.1 Overview

The HD64180S has a two-channel on-chip direct memory access controller (DMAC) that supports a proprietary chained-block transfer mode. Channel 0 is connected to the MSCI receiver and channel 1 to the MSCI transmitter. Other than this, the specifications for the two channels are identical.

#### 6.1.1 Functions

The on-chip DMAC supports the following DMA transfer modes: single-block transfer (dual address), single-block transfer (single address), and chained-block transfer (single address). The features and functions of each operation mode are given below:

##### (1) Single-block transfer mode (dual address)

- Up to 64 kbytes of data can be transferred between memory and memory, or between memory and I/O (memory-mapped I/O) in byte units.
- Up to 1 Mbyte of memory (memory-mapped I/O) can be directly addressed without having to use the MMU; up to 64 kbytes of I/O address space can be specified.
- Memory-to-memory transfers are executed by an auto request function, with burst and cycle steal modes selectable.
- For transfers between memory and I/O (memory-mapped I/O), the method for detecting the external DMA request signal  $\overline{\text{DREQ}}$  can be selected as edge or level sensitive.
- Internal interrupt generation on completion of DMA transfer can be specified.
- Maximum data transfer rate is 1.67 Mbytes/sec (at 10 MHz with no wait states inserted).

(2) Single-block transfer mode (single address)

- Up to 64 kbytes of data can be sent in byte units either from the MSCI to memory (DMAC channel 0) or from memory to the MSCI (DMAC channel 1).
- Up to 1 Mbyte of memory can be directly addressed without having to use the MMU.
- Internal interrupt generation on completion of DMA transfer can be specified.
- Maximum data transfer rate is 3.33 Mbytes/sec (at 10 MHz with no wait states inserted).

(3) Chained-block transfer mode (single address)

- This data transfer mode is available between the MSCI and memory when the MSCI is specified for the bit synchronous mode (channel 0: MSCI to memory, channel 1: memory to MSCI). By writing and reading data to/from buffers in memory, successive single and multi-frame transfers can be performed.
- Internal interrupt generation on DMA transfer completion or on completion of each frame transfer can be specified.
- Maximum data transfer rate is 3.33 Mbytes/sec (at 10 MHz with no wait states inserted).

(4) Channel 0/channel 1 priority is software-selectable in each of these transfer modes.

## 6.1.2 Configuration and Operation

The configuration of one of the DMAC channels is shown in figure 6-1.

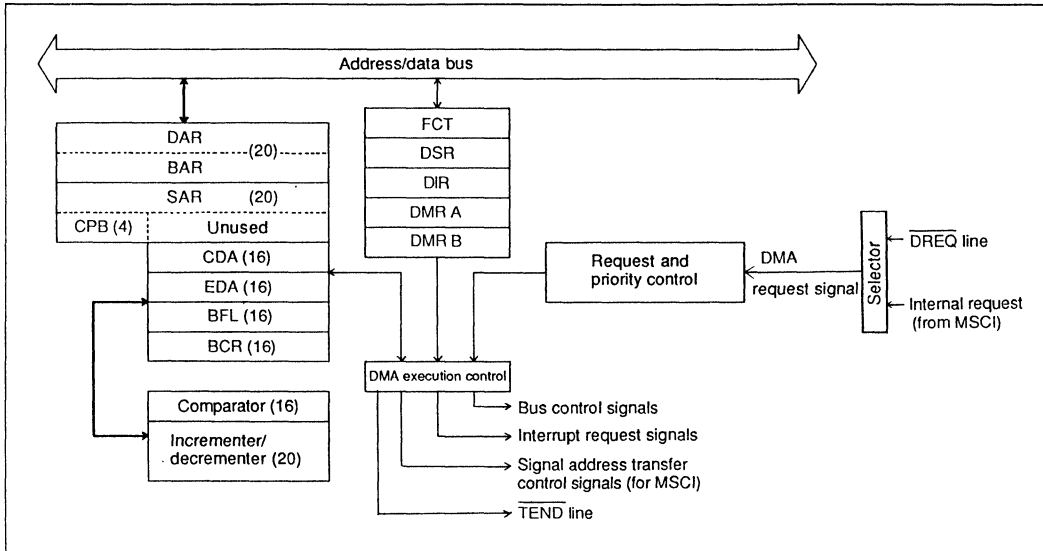


Figure 6-1. DMAC Block Diagram (one channel)

The HD64180S's on-chip DMAC supports three different operation modes; single-block transfer mode (dual address), single-block transfer mode (single address), and chained-block transfer mode. Single-block transfer (single address) and chained-block transfer are used for DMA operations between the on-chip MSCI and memory.

In each of these modes, DMA transfer is initiated by a transfer request received when DMA has been enabled after the DMAC's internal registers have been loaded with the required values (in the DMA initial state).

### (1) Single-block transfer mode (dual address)

In this mode, data is transferred between memory and memory, or between memory and I/O (memory-mapped I/O), by successive pairs of DMA read/write cycles, with each pair transferring 1 byte of data. After the specified number of bytes (up to 64 kbytes) have been transferred, the DMAC returns to the DMA initial state.

For memory-to-memory transfers, the transfer request is made by an auto-request issued by the DMAC. When the registers have been initialized, enabling the DMAC causes it to immediately request bus control and begin transfer operation. For transfers between memory and I/O (memory-mapped I/O), the  $\overline{\text{DREQ}}$  signal from the external I/O supplies the transfer request. In this case, the DMAC

requests bus control and starts the transfer operation when a low (active)  $\overline{\text{DREQ}}$  is detected while DMAC is enabled.

### (2) Single-block transfer mode (single address)

In this mode, for transfers between memory and the MSCI, 1 byte of data is transferred by a single memory read or memory write cycle. After the specified number of bytes (up to 64 kbytes) have been transferred, the DMAC returns to the DMA initial state.

Because DMAC channel 0 is hardwired to the MSCI receiver and channel 1 is hardwired to the MSCI transmitter, the transfer direction for each channel is fixed (channel 0: MSCI → memory; channel 1: memory → MSCI).

The transfer request used is an internal request signal specified in accordance with the states of the MSCI receive/transmit buffers.

### (3) Chained-block transfer mode

In this mode, frame boundary-data are transferred between memory and the MSCI by a single address type. When the MSCI is set to the bit synchronous mode, 1 byte of data is transferred by each memory read or memory write cycle. After a frame or multi-frames have been transferred, the DMAC returns to the DMA initial state.

In this mode, it is always necessary to establish the required buffers and descriptors in memory before performing transfer operations, regardless of the transfer direction.

The user may establish as many buffers as an application requires. These buffers are linked in a chain formation by the descriptors. Thus, the starting address of the buffer and of the next descriptor is specified in each descriptor.

When transferring data from the MSCI to memory, loading the necessary information into the DMAC registers and then enabling DMA causes the DMAC to write data sequentially into the receive buffers in memory. For memory-to-MSCI transfers, the data in these buffers is read sequentially. Even while DMA is enabled, buffers whose contents have already been read/written can be released and reused, enabling the transfer of successive data frames.

In this mode, transfer requests are generated by an internal request signal specified in accordance with the states of the MSCI receive/transmit buffers. The transfer direction is fixed as MSCI → memory for DMAC channel 0 and memory → MSCI for channel 1.

### 6.1.3 Registers

Each DMAC channel contains the registers listed in table 6-1. These registers can be accessed using the I/O instruction set for the on-chip CPU.

**Table 6-1. Registers**

Register Name	Symbol	I/O Address		Initial Value	Read/Write
		Channel 0	Channel 1	MSB ↔ LSB	
Destination address register (buffer address register) L*1	DARL (BARL)	0058H	0070H	XXXXXXXXXX	R/W
Destination address register (buffer address register) H*1	DARH (BARH)	0059H	0071H	XXXXXXXXXX	R/W
Destination address register (buffer address register) B*1 *2	DARB (BARB)	005AH	0072H	1 1 1 1 XXXX	R/W
Source address register L*3	SARL	005BH	0073H	XXXXXXXXXX	R/W
Source address register H*3	SARH	005CH	0074H	XXXXXXXXXX	R/W
Source address register B (chain pointer base) *1*2	SARB (CPB)	005DH	0075H	1 1 1 1 XXXX	R/W
Current descriptor address register L	CDAL	005EH	0076H	XXXXXXXXXX	R/W
Current descriptor address register H	CDAH	005FH	0077H	XXXXXXXXXX	R/W
Error descriptor address register L	EDAL	0060H	0078H	XXXXXXXXXX	R/W
Error descriptor address register H	EDAH	0061H	0079H	XXXXXXXXXX	R/W
Receive buffer length L	BFL L	0062H	007AH	XXXXXXXXXX	R/W
Receive buffer length H	BFL H	0063H	007BH	XXXXXXXXXX	R/W
Byte count register L	BCRL	0064H	007CH	XXXXXXXXXX	R/W
Byte count register H	BCRH	0065H	007DH	XXXXXXXXXX	R/W
DMA status register *4	DSR	0068H	0080H	0 0 0 0 0 0 1	R/W
DMA mode register A	DMRA	0069H	0081H	0 0 1 0 0 0 0	R/W
DMA mode register B	DMRB	006AH	0082H	0 0 0 1 1 0 0	R/W
Frame-end interrupt-counter	FCT	006BH	0083H	0 0 0 0 0 0 0	R
DMA interrupt enable register	DIR	006CH	0084H	0 0 0 0 0 0 0	R/W
DMA command register	DCR	006DH	0085H	—————	W

X : Undefined



- \*1 The register names in parentheses are used in the chained-block transfer mode. For details, refer to the descriptions of each register.
- \*2 Only the four low-order bits of this register are used. Values written to the four high-order bits are ignored; when read these bits always return a value of 1.
- \*3 These registers are not used in the chained-block transfer mode. Therefore, the user should not load any values into these registers while in the chained-block transfer mode.
- \*4 Some of the bits in this register are cleared when 1 is written to them; some of these bits are write-only. For details, refer to section 6.2.7 "DMA Status Register."

In addition to the registers above, the two registers described in table 6-2 are shared by channels 0 and 1.

**Table 6-2. Registers Shared by Channels 0 and 1**

<b>Register Name</b>	<b>Symbol</b>	<b>I/O Address</b>	<b>Initial Value MSB↔LSB</b>	<b>Read/Write</b>
DMA priority control register	PCR	001CH	0 0 0 0 0 0 0 0	R/W
DMA master enable register	DMER	001DH	1 0 0 0 0 0 0 0	R/W

## 6.2 Registers

### 6.2.1 Destination Address Register (DAR) L, H, B (Buffer Address Register (BAR) L, H, B)

One set of these three 8-bit registers is provided for each of channels 0 and 1.

- Single-block transfer mode (single address/dual address)

In this mode, these registers are used as the destination address register (DAR) which specifies the destination address to which data is to be transferred. The 20-bit destination address is specified by DARB (bits 20 – 16), DARH (bits 15 – 8), and DARL (bits 7 – 0). These registers can be used to directly access up to 1 Mbyte of memory space without having to use the MMU, or up to 64 kbytes of I/O addresses.\*1

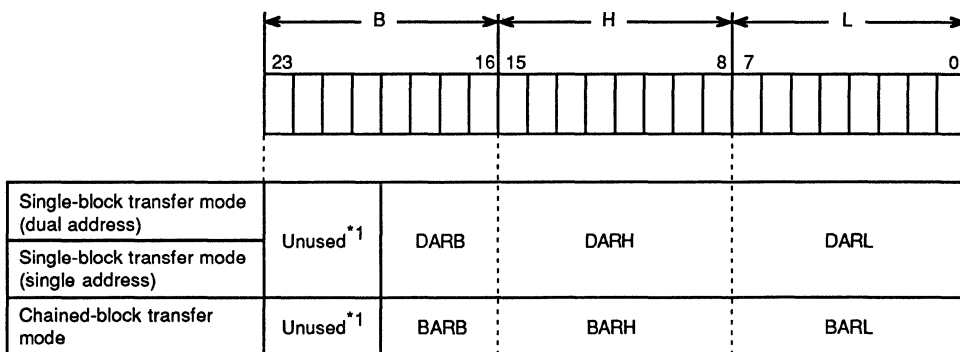
The registers should be set in DMA initial state.\*2 After a reset, the value of these registers is undefined.

\*1 When accessing an I/O, the four low-order bits of DARB are output on address lines A16-19.

\*2 The DMAC has the following operation states: initial, enable, and HALT states. For details, refer to section 6.2.12 "DMA Command Register."

- Chained-block transfer mode

In this mode, these registers are used as the buffer address register (BAR) which contains the address of the data within the buffer currently being accessed. The DMAC writes this 20-bit memory address in BARB (bits 20-16), BARH (bits 15-8), and BARL (bits 7-0). In the chained-block mode, write access by the CPU to these registers is prohibited.



\*1 The unused bits in DARB (BARB) always return a value of 1 when read.

## 6.2.2 Source Address Register (SAR) L, H, B (Chain Pointer Base (CPB))

One set of these three 8-bit registers is provided for each of channels 0 and 1.

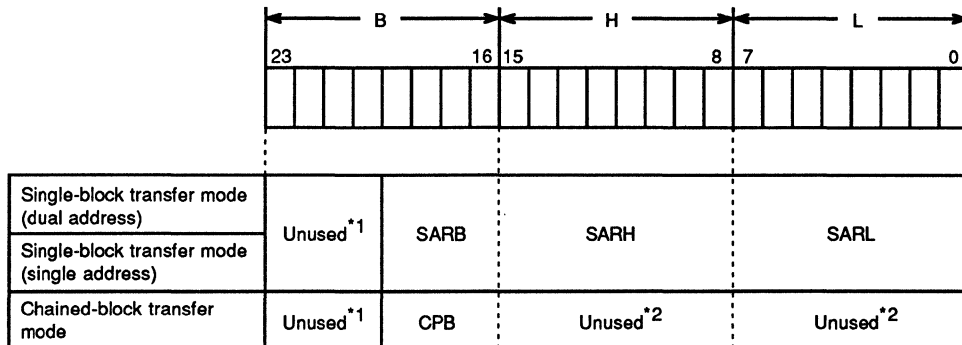
- Single-block transfer mode (single address/dual address)

In this mode, these registers are used as the source address register (SAR) which specifies the source address of the data to be transferred. The 20-bit source address is specified by SARB (bits 20 – 16), SARH (bits 15 – 8), and SARL (bits 7 – 0). These registers can be used to directly access up to 1 Mbyte of memory space without having to use the MMU, or up to 64 kbytes of I/O addresses. (During I/O access, the four low-order bits of SARB are output on A16–19).

The registers should be set in DMA initial state. After a reset, the value of these registers is undefined.

- Chained-block transfer mode

In this mode, SARB is used as the chain pointer base (CPB) to specify the four high-order bits of the 20-bit descriptor address. The 64 kbyte memory space specified by this setting is then used as the descriptor area. The registers should be set in DMA initial state. After a reset, the value of these registers is undefined.



\*1 The unused bits in SARB (CPB) always return a value of 1 when read.

\*2 In the chained-block transfer mode, these registers are used for internal operations. Nothing, therefore, should be written to them.

### 6.2.3 Current Descriptor Address Register (CDA) L, H

One set of these two 8-bit registers is provided for each of channels 0 and 1.

- Single-block transfer mode (single address/dual address)

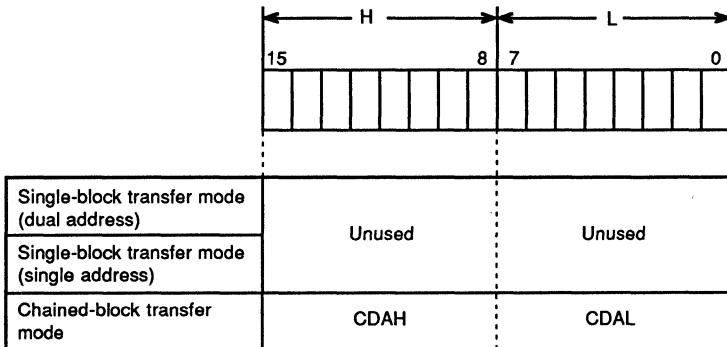
In this mode, these registers are not used. The contents of these registers are irrelevant.

- Chained-block transfer mode

These two registers should be initialized to the low-order 16 bits of the 20-bit starting address of the descriptor that points to the first buffer to be written or read in the chained-block transfer mode. Later, when the buffers are switched, the DMAC will update this to the starting address of the next descriptor. The high-order 4 bits of the descriptor are specified by the chain pointer base (CPB) and are not updated by the DMAC.

These registers can be read even when DMA is enabled. When the CPU reads the current descriptor address registers (CDA), CDAL should be read first followed by CDAH. Values read from CDAL and CDAH are the values at the moment of reading CDAL.

The registers should be set in DMA initial state. After a reset, the value of these registers is undefined.



### 6.2.4 Error Descriptor Address Register (EDA) L, H

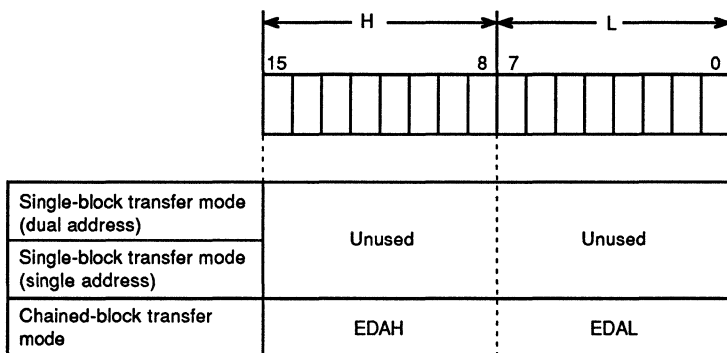
One set of these two 8-bit registers is provided for each of channels 0 and 1.

- Single-block transfer mode (single address/dual address)

In this mode, these registers are not used. The contents of these registers are irrelevant.

- Chained-block transfer mode

These two registers should be initialized to the 16 low-order bits of the 20-bit starting address of the descriptor that points to the buffer following the last buffer to be written or read in the chained-block transfer mode. The 4 high-order bits of the descriptor are specified by the chain pointer base (CPB). These registers can be written to by the CPU even when DMA is enabled. When the CPU updates the EDA, the EDAL should be written first followed by the EDAH. Both EDAL and EDAH are updated at the same time when EDAH is written. After a reset, the value of these registers is undefined.



### 6.2.5 Receive Buffer Length (BFL) L, H

One set of these two 8-bit registers is provided for each of channels 0 and 1.

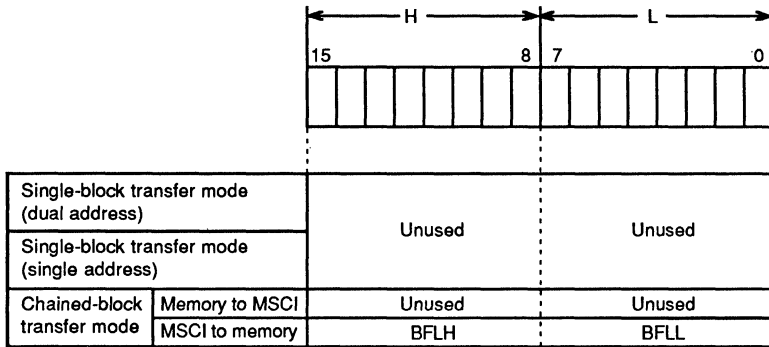
- Single-block transfer mode (single address/dual address)

In this mode, these registers are not used. The contents of these registers are irrelevant.

- Chained-block transfer mode

These registers are used only during chained-block transfer from the MSCSI to memory to specify, the buffer length in memory in byte units.

The registers should be set in DMA initial state. After a reset, the value of these registers is undefined.



### 6.2.6 Byte Count Register (BCR) L, H

One set of these two 8-bit registers is provided for each of channels 0 and 1.

- Single-block transfer mode (single address/dual address)

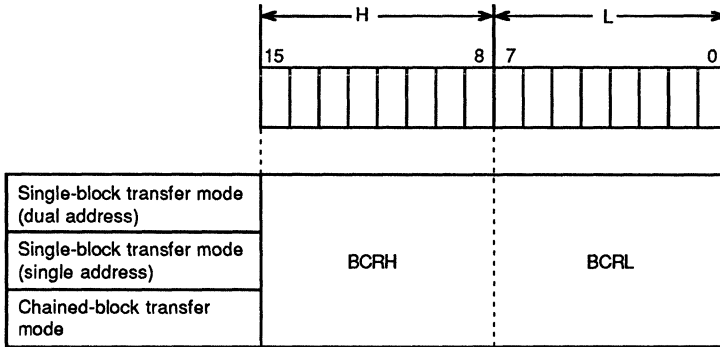
The number of bytes to be transferred (up to 64 kbytes) is set as a 16-bit value in these registers. The value in these registers is decremented (- 1) each time one byte of data is transferred by the DMAC; the transfer operation terminates when this value becomes 0000H. If 0000H is set as the initial value, 64 kbytes will be transferred.

The registers should be set in DMA initial state. After a reset, the value of these registers is undefined.

- Chained-block transfer mode

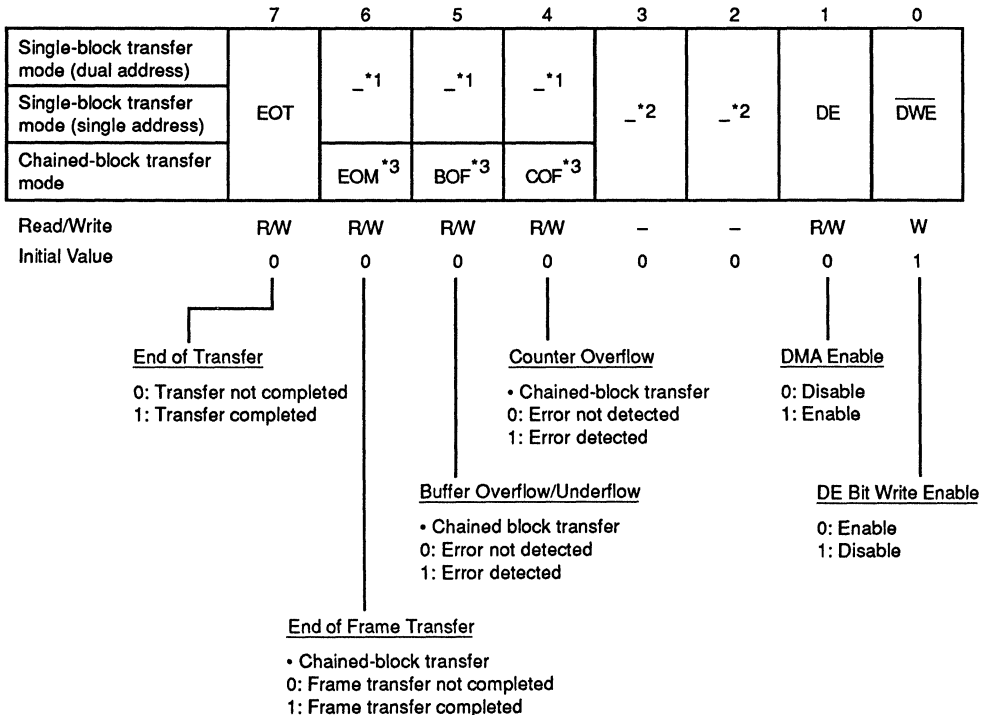
In this mode, the DMAC writes the number of bytes remaining in the buffer currently being accessed. When this value becomes 0000H, read/write access to the current buffer terminates and the next buffer is selected. At this time, the value of the byte count registers (BCR) will be updated, either to the byte length stored in the descriptor data length in the case of memory-to-MSCI (send: buffer read) transfers, or to the value of the receive buffer length (BFL) in the case of MSCI-to-memory (receive: buffer write) transfers.

The CPU is prohibited from writing data to the BCR during chained-block transfers. After a reset, the value of these registers is undefined.



### 6.2.7 DMA Status Register (DSR)

Identical registers are used for channels 0 and 1 to indicate the status of a DMA transfer. This register is also used to enable or disable each DMAC channel.



\*1 These bits are reserved. When read, their value is undefined. They can be set to 0 or 1.

\*2 Reserved. These bits always read 0 and should be set to 0.

\*3 These bits can be cleared by writing 1 to their bit positions.

### **Bit 7: EOT (End of Transfer)**

- Single-block transfer (dual/single address)/Chained-block transfer mode

This bit is set to 1 to indicate that the transfer operation by the DMAC has been completed normally. See the table "DMAC Operating Modes" in section 6.4.1 for the conditions governing DMA normal completion.

This bit is cleared to 0 when 1 is written to it.

When this bit and the EOTE bit in the DMAC interrupt enable register (DIR) are both 1, the DMAC generates an internal interrupt (DMIB).

### **Bit 6: EOM (End of Frame Transfer)**

- Single-block transfer mode (dual/single address)

This bit is reserved. When read, the value returned is undefined. Either 0 or 1 can be written to this bit.

- Chained-block transfer mode

The EOM bit goes to 1 to indicate that a transfer of one frame has been completed normally.

Provided that the frame interrupt counter (FCT) is disabled, this bit is cleared when 1 is written to this bit position.

While the FCT is enabled\*<sup>1</sup> and its value is other than 0000, the EOM bit remains 1. When 1 is written to this bit, the counter is decremented. When the counter value becomes 0000, the EOM bit goes to 0. When the FCT is enabled and the EOM bit is 0, 1 must not be written to this bit.

The EOM bit is also cleared when a frame-end interrupt-counter-clear command is issued.

When an FCT overflow occurs, FCT is reset to 0000 and the EOM bit goes to 1. At this time, the EOM bit can be cleared by a frame-end interrupt-counter-clear command specified by the DMA command register.

If this bit is 1 and the EOME bit in the DMA interrupt enable register (DIR) is 1, the DMAC generates an internal interrupt (DMIB).

\*<sup>1</sup> See section 6.2.8 "DMA Mode Register A" for details of the CNTE bit.



#### **Bit 5: BOF (Buffer Overflow/Underflow)**

- Single-block transfer mode (dual /single address)

This bit is reserved. When read, the value returned is undefined. Either 0 or 1 can be written to this bit.

- Chained-block transfer mode

The BOF bit goes to 1 when a buffer overflow or underflow occurs in the DMAC. In this mode, buffer overflow is defined as the condition that occurs during MSCI-to-memory (receive) transfer when a transfer request is issued by the MSCI while the value of the current descriptor address register (CDA) and that of the error descriptor address register (EDA) are the same. A buffer underflow is the condition during memory-to-MSCI transfer (transmit) when a transfer request is issued by the MSCI while the value of the CDA and the EDA are the same.

When 1 is written to this bit, it is cleared.

If both this bit and the BOFE bit in DIR are 1, the DMAC generates an internal interrupt (DMIA).

#### **Bit 4: COF (Counter Overflow)**

- Single-block transfer mode (dual/single address)

This bit is reserved. When read, the value returned is undefined. Either 0 or 1 can be written to this bit.

- Chained-block transfer mode

The COF bit indicates an overflow of the frame-end interrupt-counter (FCT). It goes to 1 when a frame transfer is completed while the FCT value is 1111. At this time, the FCT goes to 0000.

When 1 is written to this bit, it is cleared.

If both this bit and the COFE bit in DIR are 1, the DMAC generates an internal interrupt (DMIA).

**Bits 3 and 2:** Reserved. Bits 3 and 2 are always read 0 and should be set to 0.

#### **Bit 1: DE (DMA Enable)**

- Single-block transfer (dual/single address)/Chained-block transfer modes

This bit enables or disables the corresponding DMA channel.

DE	Function
0	Disables DMA channel 0 or 1
1	Enables DMA channel 0 or 1

When writing a value to the DE bit, 0 must be written to  $\overline{\text{DWE}}$  bit at the same time. For auto-requests (memory-to-memory transfers), transfer starts when this bit is set to 1. If the transfer request source is an external line or the MSCI, transfer starts when the request is issued while this bit is 1.

When the DMA transfer end condition is satisfied, the DE bit of the corresponding channel is automatically cleared. For the DMA transfer end condition, see the table "DMAC Operating Modes" in section 6.4.1.

The DMAC enters the halt state when 0 is written to the DE bit during a transfer.

**Bit 0:  $\overline{\text{DWE}}$  (DE Bit Write Enable)**

- Single-block transfer (dual/single address)/Chained-block transfer mode

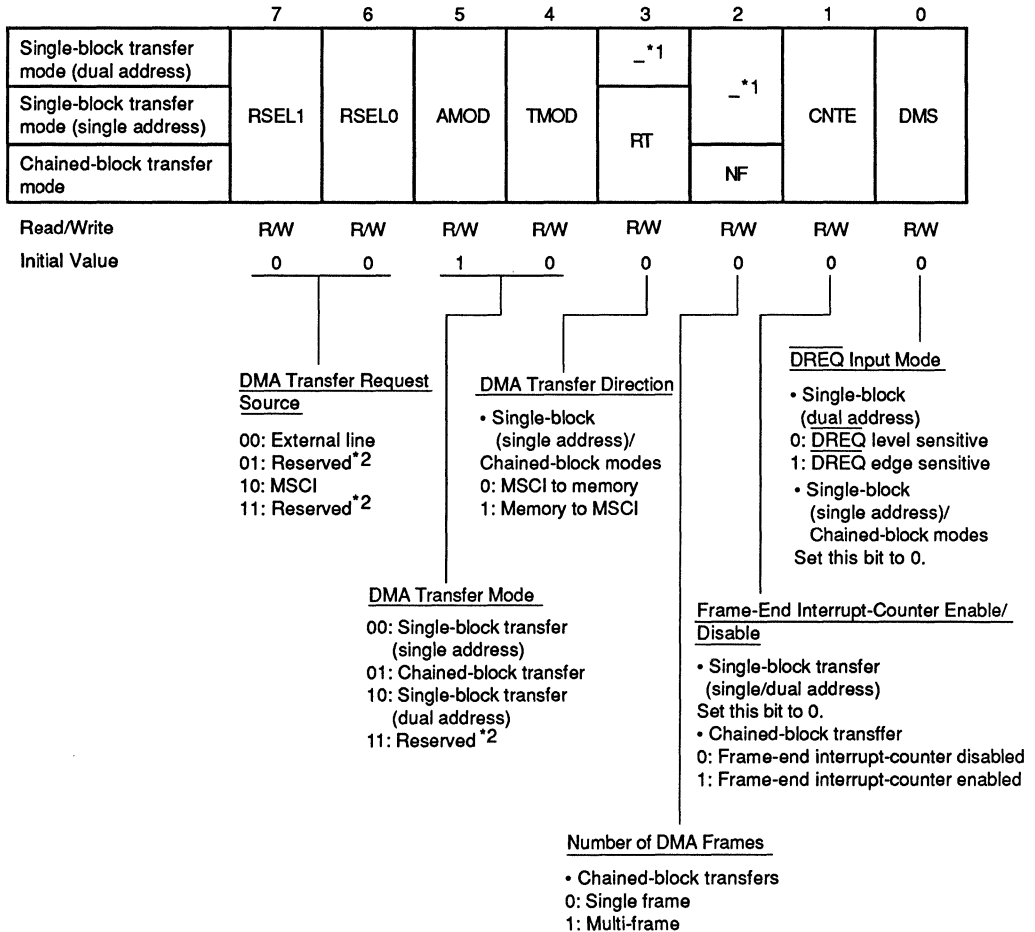
$\overline{\text{DWE}}$  enables write operations to the DMA enable (DE) bit. When writing to the DE bit, 0 must be written to  $\overline{\text{DWE}}$ . However, since the value of this bit is not retained, 0 must be written to  $\overline{\text{DWE}}$  bit when writing any values to the DE bit.

When read, this bit always returns a value of 1.

## 6.2.8 DMA Mode Register A (DMRA)

Identical registers are used for channels 0 and 1. This register specifies the DMA transfer source, transfer mode, transfer direction, number of DMA frames, enable/disable of the frame-end interrupt-counter, and the method for receiving a DMA request from an external line (level or edge sensitive).

This register should be set in DMA initial state.



\*1 Reserved bit. When this bit is read, the value returned is undefined. Either 0 or 1 can be written to this bit.

\*2 This setting is reserved. If selected, correct operation cannot be guaranteed.

### Bits 7-6: RSEL1-0 (DMA Transfer Request Source)

- Single-block (dual/single address)/Chained-block modes

The RSEL bits specify the DMA transfer request source. These bits are cleared to 0 after reset.

RSEL1	RSEL0	Function
0	0	DMA transfer request by external line
0	1	Reserved
1	0	DMA transfer request by MSCI*
1	1	Reserved

\* The DMS bit in this register must be set to 0 before the DMAC can receive a request from the MSCI.

#### Bit 5: AMOD (DMA Transfer Address Mode)

#### Bit 4: TMOD (DMA Transfer Mode)

- Single-block (dual/single address)/Chained-block modes

These bits specify the DMAC operation mode. After reset, the value of AMOD is 1 and that of TMOD is 0.

AMOD	TMOD	DMAC Operation Mode
0	0	Single-block transfer (single address)
0	1	Chained-block transfer
1	0	Single-block transfer (dual address)
1	1	Reserved*

\* This setting is reserved. If selected, correct operation cannot be guaranteed.

#### Bit 3: RT (DMA Transfer Direction)

- Single-block transfer (dual address)

This bit is reserved. When read, the value is undefined. Either 0 or 1 can be written to this bit.

- Single-block transfer (single address)/Chained-block transfer mode

In these modes, this bit specifies the direction of the DMA transfer (either from the MSCI to memory or from memory to the MSCI).

This bit is cleared to 0 after a reset.

RT	DMA Transfer Direction
0	From MSCI to memory
1	From memory to MSCI

### Bit 2: NF (Number of DMA Frames)

- Single-block transfer (dual/single address)

This bit is reserved. When read, the value is undefined. Either 0 or 1 can be written to this bit.

- Chained-block transfer

This bit specifies either single or multi-frame chained-block transfer between memory and the MSCl.

This bit is cleared to 0 after a reset.

NF	Transfer mode
0	Single frame mode
1	Multi-frame mode

### Bit 1: CNTE (Frame-End Interrupt-Counter Enable/Disable)

- Single-block transfer (dual/single address)

In these modes, this bit should be set to 0.

- Chained-block transfer

This bit specifies enable/disable of the frame-end interrupt-counter (FCT). See section 6.2.7 "DMA Status Register" and 6.2.10 "Frame-End Interrupt-Counter."

This bit is cleared to 0 after a reset.

CNTE	Function
0	Disables the frame-end interrupt-counter
1	Enables the frame-end interrupt-counter

### Bit 0: DMS ( $\overline{\text{DREQ}}$ Input Mode)

- Single-block (dual/single address)/Chained-block modes

The DMS bit specifies the mode (level or edge sensitive) in which DMA requests input from an external line or from the MSCl.

DMS	Function
0	<p>Level sensitive</p> <p>When the DMS bit is set to level sensitive, the DMAC samples the <math>\overline{\text{DREQ}}</math> line at the rising edge of the <math>\phi</math> clock, two cycles prior to the end of the current machine cycle. When the DMAC acquires the bus control, the DMAC starts transfer operation at the next bus cycle.</p>
1	<p>Edge sensitive</p> <p>When the DMS bit is set to edge sensitive, if the DMAC detects a <math>\overline{\text{DREQ}}</math> falling edge prior to the rising edge of the <math>\phi</math> clock two cycles before the end of a machine cycle, the DMAC acquires the bus control and starts transfer operation at the next bus cycle.</p>

Note: When receiving a DMA transfer request from the MSCI (the RSEL1–0 bits in DMA mode register A are set to 1,0) the DMS bit must be 0. In other words, the DMA request from the MSCI is always level sensitive.

### 6.2.9 DMA Mode Register B (DMRB)

Identical registers are used for channels 0 and 1. This register is used to specify the transfer direction when single-block (dual address) mode has been selected. It is also used to specify the transfer mode in the single-block transfer between memory and memory. The single-block (dual address) mode is specified by the AMOD and TMOD bits in DMA mode register A.

This register should be set in DMA initial state.

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)				DM1	DM0	SM1	SM0	MMOD
Single-block transfer mode (single address)	_*1	_*1	_*1	_*2	_*2	_*2	_*2	_*2
Chained-block transfer mode								

Read/Write	–	–	–	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	1	1	0	0	0

Destination (address increment or decrement)

- Single-block (dual address)
- 00: Memory (+1)
- 01: Memory (–1)
- 10: Memory (fixed)
- 11: I/O (fixed)

Source (address increment or decrement)

- Single-block (dual address)
- 00: Memory (+1)
- 01: Memory (–1)
- 10: Memory (fixed)
- 11: I/O (fixed)

Mode for Memory-to-Memory Transfers

- Single-block (dual address)
- 0: Cycle steal mode
- 1: Burst mode

\*1 Reserved. These bits always read 0 and should be set to 0.

\*2 Reserved. When read, these bits are undefined. They can be set to 0 or 1.

**Bits 7-5:** Reserved. These bits always read 0 and should be set to 0.

**Bits 4-3: DM1-0 (Destination)**

- Single-block (dual address)

The DM bits specify the DMA transfer destination (memory or I/O), as well as address increment/decrement. These bits are both set to 1 after reset.

DM1	DM0	Function
0	0	Memory is specified as the destination; the destination address register (DAR) is incremented (+1) after each byte transfer.
0	1	Memory is specified as the destination; DAR is decremented (-1) after each byte transfer.
1	0	Memory is specified as the destination; the address is not changed.
1	1	I/O is specified as the destination; the address is not changed.

- Single-block (single address)/Chained-block transfer modes

In these modes, these bits are reserved. When read, the value returned is undefined. When writing to these bit positions, either 0 or 1 can be written.

**Bits 2-1: SM1-0 (Source)**

- Single-block (dual address)

The SM bits specify the DMA transfer source (memory or I/O), as well as address increment/decrement. These bits are cleared to 0 after reset.

SM1	SM0	Function
0	0	Memory is specified as the source; source address register (SAR) is incremented (+1) after each byte transfer.
0	1	Memory is specified as the source; SAR is decremented (-1) after each byte transfer.
1	0	Memory is specified as the source; the address is not changed.
1	1	I/O is specified as the source; the address is not changed.

- Single-block (single address)/Chained-block transfer modes

In these modes, these bits are reserved. When read, the value returned is undefined. When writing to these bit positions, either 0 or 1 can be written.

The DM1–0 and SM1–0 bits are used in combination to specify the following transfer modes.

DM1	DM0	SM1	SM0	Transfer Mode	Address Increment/Decrement*1
0	0	0	0	Memory-to-Memory	SAR + 1, DAR + 1
0	0	0	1	Memory-to-Memory	SAR – 1, DAR + 1
0	0	1	0	Memory*2 -to-Memory	SAR fixed, DAR + 1
0	0	1	1	I/O-to-Memory	SAR fixed, DAR + 1
0	1	0	0	Memory-to-Memory	SAR + 1, DAR – 1
0	1	0	1	Memory-to-Memory	SAR – 1, DAR – 1
0	1	1	0	Memory*2 -to-Memory	SAR fixed, DAR – 1
0	1	1	1	I/O-to-Memory	SAR fixed, DAR – 1
1	0	0	0	Memory-to-Memory*2	SAR + 1, DAR fixed
1	0	0	1	Memory-to-Memory*2	SAR – 1, DAR fixed
1	0	1	0	Reserved*3	
1	0	1	1	Reserved*3	
1	1	0	0	Memory-to-I/O	SAR + 1, DAR fixed
1	1	0	1	Memory-to-I/O	SAR – 1, DAR fixed
1	1	1	0	Reserved*3	
1	1	1	1	Reserved*3	

\*1 SAR: Source address register

DAR: Destination address register

\*2 Because this address is fixed, the memory in this mode could be used as memory-mapped I/O.

\*3 This setting is reserved. If selected, correct operation is not guaranteed.

There are 12 modes available. Transfers between I/O (memory-mapped I/O) and I/O (memory-mapped I/O) are not allowed.

Irrespective of the address increment/decrement setting, the  $\overline{\text{ME}}$  line is always asserted (low) when memory (including memory-mapped I/O) is specified; likewise, the  $\overline{\text{IOE}}$  line is asserted (low) when I/O is specified.

When I/O or memory-mapped I/O is specified as either the source or destination, DMA operation is controlled by the  $\overline{\text{DREQ}}$  line. For transfers between memory and memory, DMA operation is not affected by the  $\overline{\text{DREQ}}$  line (auto-request).



## Bit 0: MMOD (Mode for Memory-to-Memory Transfers)

- Single-block mode (dual address)

MMOD specifies the DMA mode (cycle steal or burst) for transfers between memory and memory.

MMOD	Function
0	Cycle steal mode. The bus is released at the end of each byte transfer. During the bus cycle of another master, the DMAC continues to request the bus.
1	Burst mode. In the burst transfer mode, once a DMA transfer starts, the DMAC retains control of the bus until the transfer is completed. Bus control is never returned to the CPU. The DMAC continues to request bus control until the byte count register goes to 0000H.

- Single-block mode (single address)/Chained-block mode

This bit is reserved. When read, the value returned is undefined. Either 0 or 1 can be written to this bit position.

## 6.2.10 Frame-End Interrupt-Counter (FCT)

Identical registers are used for channels 0 and 1. This read-only 4-bit counter counts the unprocessed interrupts which occurred during a chained-block transfer between the MSC1 and memory in the multi-frame transfer mode.

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)					_ *2	_ *2	_ *2	_ *2
Single-block transfer mode (single address)	_ *1	_ *1	_ *1	_ *1				
Chained-block transfer mode					FCT3	FCT2	FCT1	FCT0
Read/Write	-	-	-	-	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

Frame-End Interrupt-Counter Value

- \*1 Reserved. These bits always read 0.
- \*2 Reserved. When read, these bits are undefined.

**Bits 7-4:** These bits are reserved. They always read 0.

### **Bits 3-0: FCT3 – 0 (Frame-End Interrupt-Counter Value)**

- Single-block mode (dual/single address)

These bits are reserved. When read, the value is undefined.

- Chained-block mode

In the chained-block multi-frame transfer mode, the DMAC can request a DMIB internal interrupt (frame end interrupt) at the end of each frame. (The DMAC remains enabled and successive interrupts can occur.) If the transfer of successive requested frames is completed before the CPU executes the interrupt processing routine, some interrupt requests might remain unprocessed. This register counts such interrupts.

The frame-end interrupt-counter (FCT) is enabled or disabled by the CNTE bit in DMA mode register A (DMRA). For details, see section 6.2.8 "DMA Mode Register A."

The EOM bit in the DSR remains at 1 until this counter goes to 0000. When 1 is written to the EOM bit, the FCT is decremented. (While the FCT is enabled and its value is 0000, the EOM bit in the DSR must not be set to 1.)

If frame transfer continues after the counter value reaches 1111, the DMAC stops the transfer operation after the next 1-frame transfer has been completed. At this time, the COF bit in the DSR goes to 1. If the COFE bit in the DMA interrupt enable register is 1, the DMAC generates a counter overflow interrupt (DMIA).

At this time, the FCT goes to 0000 and the EOM bit in DSR goes to 1. The EOM bit can be cleared by a frame-end interrupt-counter-clear command specified by the DMA command register.

## 6.2.11 DMA Interrupt Enable Register (DIR)

Identical registers are used for channels 0 and 1.

The DIR register enables interrupts caused by the EOT, EOM, BOF, and COF bits of the DSR.

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)	EOTE	_*1	_*1	_*1	_*2	_*2	_*2	_*2
Single-block transfer mode (single address)								
Chained-block transfer mode		EOME	BOFE	COFE				
Read/Write	R/W	R/W	R/W	R/W	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0

Transfer End Interrupt Enable

0: Disable  
1: Enable

Counter Overflow Interrupt Enable

• Chained-block transfer mode  
0: Disable  
1: Enable

Frame Transfer End Interrupt Enable

• Chained-block transfer mode  
0: Disable  
1: Enable

Buffer Overflow/Underflow Interrupt Enable

• Chained-block transfer mode  
0: Disable  
1: Enable

\*1 Reserved. When read, these bits are undefined. They can be set to 0 or 1.

\*2 Reserved. These bits always read 0 and should be set to 0.

### Bit 7: EOTE (Transfer End Interrupt Enable)

- Single-block (dual/single address)/Chained-block transfer

EOTE specifies whether to enable or disable a DMA normal end interrupt (DMIB) originating from the EOT bit.

#### EOTE Function

0	Disables an internal interrupt (DMIB) caused by the EOT bit
1	Enables an internal interrupt (DMIB) caused by the EOT bit

#### Bit 6: EOME (Frame Transfer End Interrupt Enable)

- Single-block transfer (dual/single address)

This bit is reserved. When read, the value is undefined. Either 0 or 1 can be written to this bit position.

- Chained-block transfer mode

EOME specifies whether to enable or disable a DMA frame end interrupt (DMIB).

EOME	Function
0	Disables an internal interrupt (DMIB) caused by the EOM bit
1	Enables an internal interrupt (DMIB) caused by the EOM bit

#### Bit 5: BOFE (Buffer Overflow/Underflow Interrupt Enable)

- Single-block transfer (dual/single address)

This bit is reserved. When read, the value is undefined. Either 0 or 1 can be written to this bit position.

- Chained-block transfer mode

BOFE specifies whether to enable or disable a buffer overflow/underflow interrupt (DMIA).

BOFE	Function
0	Disables an internal interrupt (DMIA) caused by the BOF bit
1	Enables an internal interrupt (DMIA) caused by the BOF bit

#### Bit 4: COFE (Counter Overflow Interrupt Enable)

- Single-block transfer (dual/single address)

This bit is reserved. When read, the value is undefined. Either 0 or 1 can be written to this bit position.

- Chained-block transfer mode

COFE specifies whether to enable or disable a counter overflow interrupt (DMIA).

COFE	Function
0	Disables an internal interrupt (DMIA) caused by the COF bit
1	Enables an internal interrupt (DMIA) caused by the COF bit

**Bits 3-0:** Reserved. These bits always read 0 and should be set to 0.

## 6.2.12 DMA Command Register (DCR)

Identical registers are used for channels 0 and 1. The DCR register is used to issue a software abort or frame-end interrupt-counter-clear command for the DMAC. This register always reads 00H.

	7	6	5	4	3	2	1	0
Single-block transfer mode(dual address)								
Single-block transfer mode(single address)	_*1	_*1	_*1	_*1	_*1	_*1	CMD1	CMD0
Chained-block transfer mode								
Read/Write	-	-	-	-	-	-	W	W
Initial Value	-	-	-	-	-	-	-	-

Command Specification \*2  
 01: Software abort  
 10: Frame-end interrupt-counter-clear  
 Others: Reserved

\*1 Reserved. These bits always read 0 and should be set to 0.

\*2 These commands should not be issued while the corresponding DMAC channel is enabled (DE = 1).

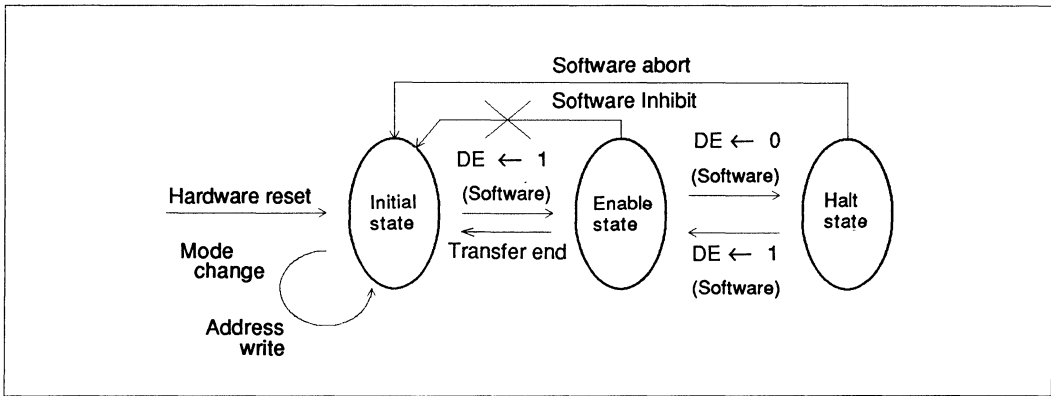
No values other than those shown here (01H and 02H) should be written to this register.

Command Name	Function
Software abort (01H)	Initializes the corresponding DMAC channel (see figure 6-2). All DMAC registers maintain their previous value.
Frame-end interrupt - counter-clear (02H)	Clears the frame-end interrupt-counter (FCT) of the corresponding DMAC channel to 0H and the EOM bit in the DSR to 0.

If the DMAC has been disabled by software (DE bit cleared), the DMAC must be initialized by a software abort command before the new DMAC operation. This is necessary because the DMAC retains its internal state after being disabled.

If the DMAC has been disabled because the transfer end conditions were satisfied, no initialization is necessary.

The state transition diagram for the three operating modes of the DMAC (initial state, enable state, and halt state) is shown in figure 6-2.



**Figure 6-2. Software Abort and DMAC Operation**

If the DE bit in the DSR is cleared by the CPU while the DMAC is enabled, the DMAC enters the halt state.

Issuing a software abort command while the DMAC is halted initializes it. The DMRA, DMRB, DSR, FCT, and DIR remain unchanged.

**Note:** The mode, address, and data length should not be changed during the DMAC halt or enable states. If it is necessary to change these values, the DMAC should be initialized by a software abort in advance.

After a DMA operation is completed (see the explanation of transfer end conditions in table 6-5 "DMAC Operating Modes"), the DMAC is initialized and no software abort commands are necessary.

### 6.2.13 DMA Priority Control Register (PCR)

PCR is shared by DMAC channels 0 and 1. It specifies channel priority.

When both channels issue a DMA request, the higher priority channel is given control of the bus.\*

\* DMA transfer may be requested by the following sources: an external signal ( $\overline{\text{DREQ}}$  line), an internal request (MSCI), or auto-request. See section 6.4 "Operating Modes."

	7	6	5	4	3	2	1	0
Single-block Transfer Mode (dual address)								
Single-block Transfer Mode (single address)	-*	-*	-*	-*	-*	-*	-*	PR0
Chained-block Transfer Mode								

Read/Write	-	-	-	-	-	-	-	R/W
Initial Value	0	0	0	0	0	0	0	0

Channel Priority

0: Channel 0 has priority over channel 1  
 1: Channel 1 has priority over channel 0

\* These bits are reserved. They always read 0 and must be set to 0.

**Bits 7–1:** Reserved. Bits 7–1 always read 0 and should be set to 0.

**Bit 0: PR0 (Channel Priority)**

Single-block (dual/single address)/Chained-block transfer

PR0	Function
0	DMAC channel 0 has priority over channel 1.
1	DMAC channel 1 has priority over channel 0.

## 6.2.14 DMA Master Enable Register (DMER)

DMER is shared by DMAC channels 0 and 1. It specifies whether to enable or disable the DMAC.

	7	6	5	4	3	2	1	0
Single-block Transfer Mode (dual address)	DME	-*	-*	-*	-*	-*	-*	-*
Single-block Transfer Mode (single address)								
Chained-block Transfer Mode								
Read/Write	R/W	-	-	-	-	-	-	-
Initial Value	1	0	0	0	0	0	0	0

|  
DMA Master-Enable  
 0: Disable  
 1: Enable

\* Reserved. These bits always read 0 and should be set to 0.

### Bit 7: DME (DMA Master-Enable)

- Single-block transfer (single/dual address)/Chained-block transfer modes  
Enables channels 0 and 1.

DME	Function
0	Disables both DMAC channels
1	Enables both channels (depending on each DE bit)

The DME bit is set to 1 by a reset. When the  $\overline{\text{NMI}}$  line is asserted, the DME bit goes to 0. This disables the DMAC and allows control to be passed to the CPU. The DE bit and DMAC internal states are not affected by the  $\overline{\text{NMI}}$  signal. After the  $\overline{\text{NMI}}$  has been processed, the DMAC can be restarted by writing a 1 to the DME bit.

Figure 6-3 shows DME bit state transitions during  $\overline{\text{NMI}}$  processing. Figure 6-4 shows the DMA enable logic.



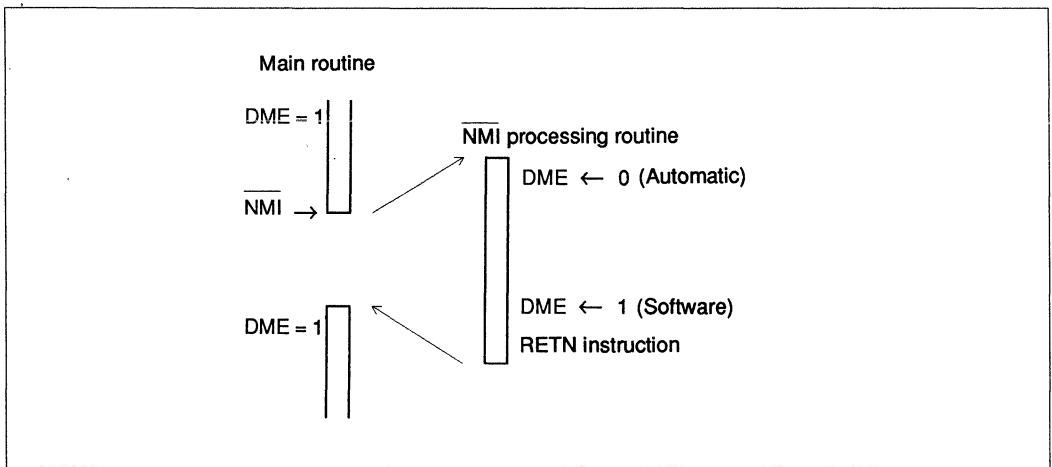


Figure 6-3. DME Bit State Transitions

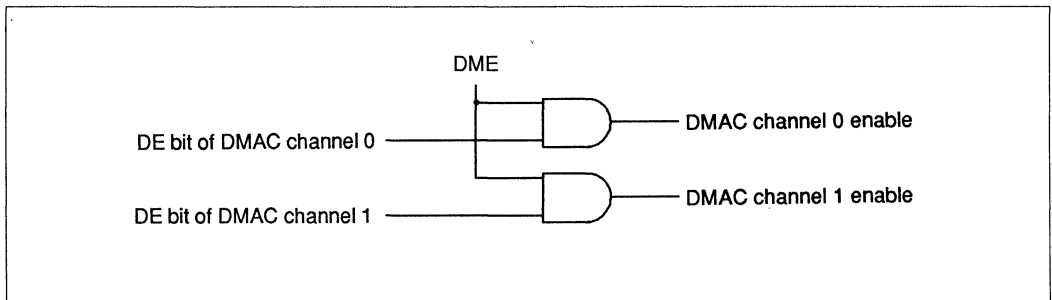


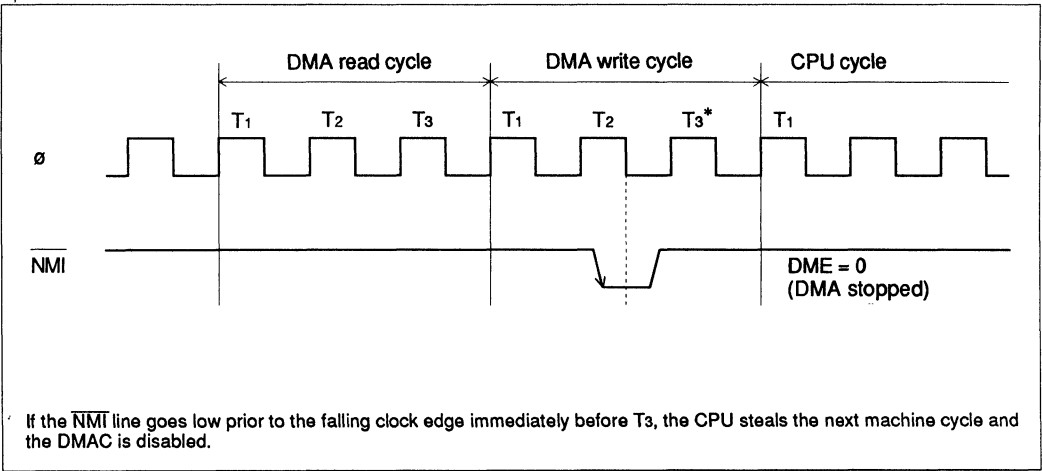
Figure 6-4. DMA Enable Logic

The DME bit can be set to 1 by software (it cannot be cleared).

**Bits 6-0:** Reserved. Bits 6 - 0 always read 0 and should be set to 0.

**NMI and DMA Operation:** If an NMI occurs during DMAC operation, the DMAC completes the current byte transfer, then passes control to the CPU. The DME bit in the DMA Master Enable Register is cleared and the DMA transfer is suspended. Since the DE bits and internal register values remain unchanged, the DMAC can be restarted by setting the DME bit.

Figure 6-5 shows the effect of an NMI on DMAC operation. An NMI could be used to suspend DMAC operation.



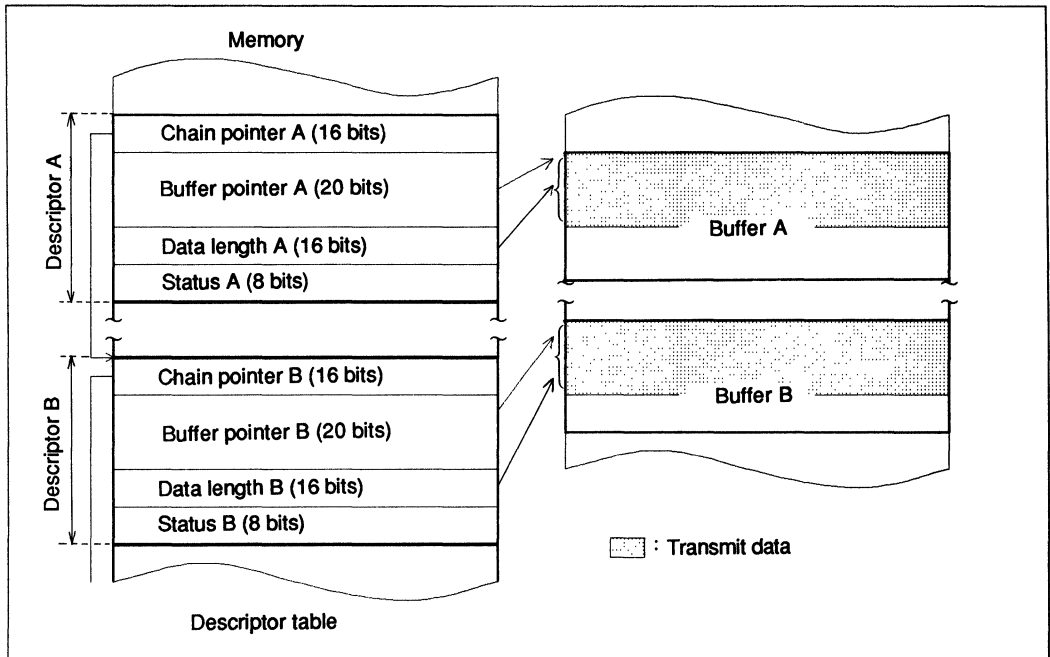
**Figure 6-5. Effect of an  $\overline{\text{NMI}}$  on DMAC Operation**

## 6.3 Descriptor

In the chained-block transfer mode, transmit/receive data is stored in buffers in system memory. Each buffer has a descriptor indicating its attributes. The buffers are linked by these descriptors.

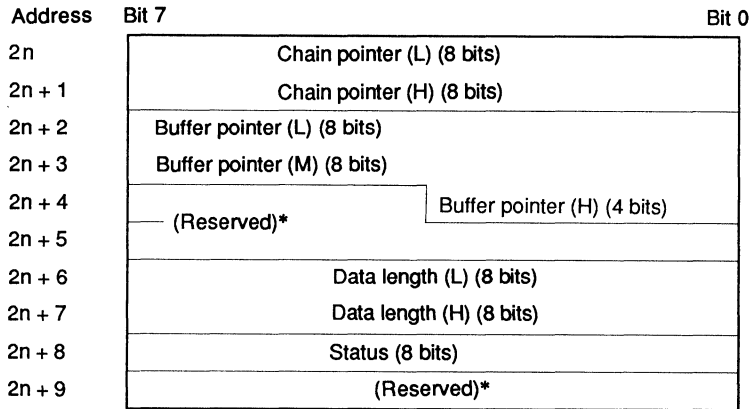
### 6.3.1 Chained-block Transfers from Memory to the MSCI (Transmit)

Figure 6-6 shows descriptors and buffers in system memory for chained-block transfers from memory to the MSCI.



**Figure 6-6. Descriptors and Buffers for Chained-block Transfers from Memory to the MSCI**

The descriptor consists of a 16-bit chain pointer, 20-bit buffer pointer, 16-bit data length field, and an 8-bit status field. These fields are allocated to system memory in bytes. Figure 6-7 shows the format of a descriptor. A detailed explanation of the descriptor follows.



\* The reserved areas in the descriptor are not overwritten by the DMAC and therefore retain their previous value.

**Figure 6-7. Descriptor Format**

**Chain Pointer (16 Bits):** The chain pointer specifies the 16 low-order bits of the 20-bit start address of the next descriptor. The four high-order bits are specified by the chain pointer base (CPB). The chain pointer value is loaded into the Current Descriptor Address Register (CDA) during buffer switching.

**Buffer Pointer (20 Bits):** The buffer pointer specifies the start address of the buffer corresponding to this descriptor. The buffer pointer value is loaded into the Buffer Address Register (BAR) at the start of transfer or during buffer switching.

**Data Length (16 Bits):** Data length specifies the length (in bytes) of the data in the buffer corresponding to this descriptor. The data length value is loaded into the Byte Count Register (BCR) at the start of transfer or during buffer switching.

For chained-block transfers from memory to the MSCI (transmit), this is controlled by the CPU (initialized by the CPU).

**Status (8 Bits):** Status specifies a frame transfer end or DMA transfer end after the buffer data corresponding to this descriptor has been transferred.

For chained-block transfers from memory to the MSCI (transmit), this is controlled by the CPU (initialized by the CPU).

Table 6-3 shows the status format for chained-block transfers from memory to the MSCI (transmit).

**Table 6-3. Status Configuration (transmit)**

<b>Bit</b>	<b>Function</b>
7	EOM
6	(Unused)
5	(Unused)
4	(Unused)
3	(Unused)
2	(Unused)
1	(Unused)
0	EOT

The functions of these bits are described below.

**Bit 7: EOM (End of Frame Transfer)**

Specifies whether or not the frame ends with the corresponding buffer.

<b>EOM</b>	<b>Function</b>
0	The frame does not end with the corresponding buffer.
1	The frame ends with the corresponding buffer.

**Bit 0: EOT (End of Transfer)**

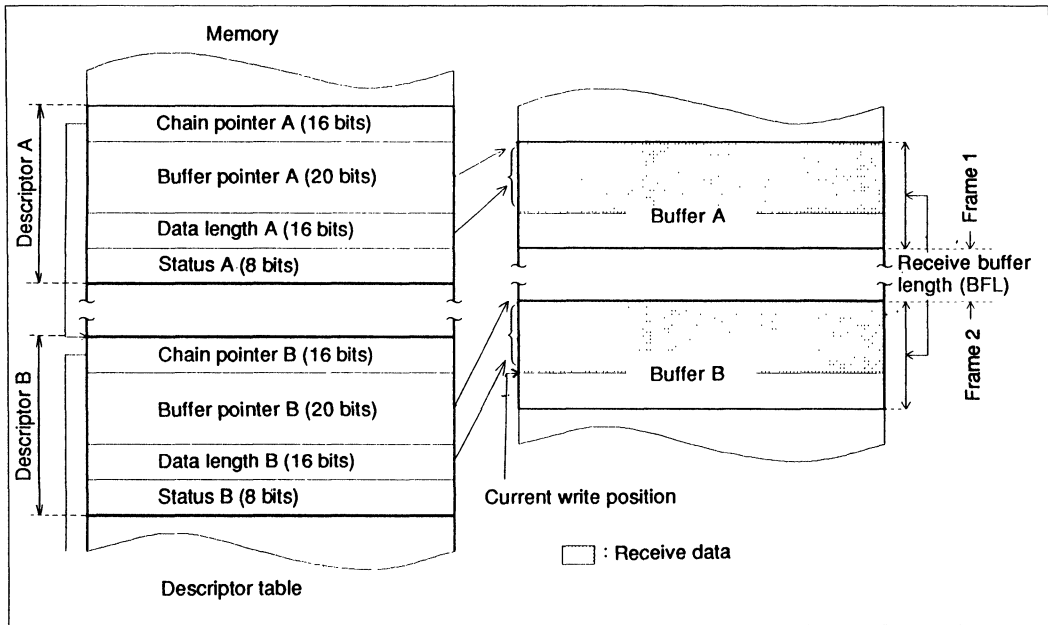
In multi-frame mode, this specifies whether or not to end the DMA transfer after transferring the current frame.

<b>EOT</b>	<b>Function</b>
0	End of transfer is not specified.
1	End of transfer is specified after transferring the current frame.

Note: Status bits 6 – 1 are not used for chained-block transfers from memory to the MSCI.

### 6.3.2 Chained-block Transfers from the MSCI to Memory (Receive)

Figure 6-8 shows descriptors and buffers in system memory for chained-block transfers from the MSCI to memory.



**Figure 6-8. Descriptors and Buffers for Chained-block Transfers from the MSCI to Memory**

The descriptor format for chained-block transfers from the MSCI to memory is the same as that shown in figure 6-7. The functions of the associated fields are listed below.

**Chain Pointer (16 Bits):** The chain pointer specifies the 16 low-order bits of the 20-bit start address of the next descriptor. The four high-order bits are specified by the chain pointer base (CPB). The chain pointer value is loaded into the current descriptor address register (CDA) during buffer switching.

**Buffer Pointer (20 Bits):** This specifies the start address of the buffer corresponding to this descriptor. The buffer pointer value is loaded into the buffer address register (BAR) at the start of transfer or during buffer switching.

**Data Length (16 Bits):** Data length indicates the length (in bytes) of data in the buffer corresponding to this descriptor.

For chained-block transfers from the MSCI to memory (receive), the length is specified by the DMAC. After the DMAC loads the receive data into the buffer, it loads the byte count of the data which is written to the buffer into this field.

**Status (8 Bits):** This indicates the status of the data in the buffer corresponding to this descriptor.

For chained-block transfers from the MSCI to memory (receive), the status is specified by the DMAC. After the DMAC loads the data into the buffer, it loads the status of the data which is written to the buffer into this field.

**Table 6-4. Status Configuration (receive)**

Bit	Function
7	EOM
6	Short frame
5	Abort
4	Residue bit
3	Overrun
2	CRC
1	(Unused)
0	(Unused)

Table 6-4 shows the status configuration for chained-block transfers from the MSCI to memory (receive).

If the frame ends with the buffer corresponding to the descriptor, the MSCI Frame Status Register (MFST) value, which is set immediately after the MSCI transmits the end-of-frame from the receive buffer to the internal data bus, is written to status bits 7-0.\* If the frame does not end with the buffer corresponding to the descriptor and if the buffer is switched inside a frame, status bits 7-0 are cleared.

\* For an explanation of each bit, see sections 4.2.11 "MSCI Status Register 2" and 4.2.13 "MSCI Frame Status Register."

## 6.4 Operating Modes

### 6.4.1 Overview

The on-chip DMAC supports single-block transfer modes (dual/single address) and chained-block transfer mode (single address). Each transfer mode is summarized in table 6-5.

**Table 6-5. DMAC Operating Modes**

Operating Mode*1	Single-block transfer mode (dual address)		Single-block transfer mode (single address)		Chained-block transfer mode (single address)				
	Between Memory and Memory	Between Memory and I/O*2 (memory mapped I/O)	Memory to MSCI	MSCI to Memory	Memory to MSCI		MSCI to Memory		
Requesting Source	Auto request*3	External line or MSCI	MSCI		Single frame transfer	Multi-frame transfer	Single frame transfer	Multi-frame transfer	
Data transfer unit	Single block		Single block		Single frame	Multi-frame	Single frame	Multi-frame	
Bus mode	Before transfer start either the burst or cycle steal mode must be selected.*4	Control is based on one of two methods. The request line is either edge or level sensitive.*5	Started by a request from the MSCI. A request from the MSCI is level sensitive.						
Minimum transfer cycle/byte	6 cycles		3 cycles						
Operation	Source address	Specified by source address register (SAR)			MSCI receiver	Specified by buffer address register (BAR)		MSCI receiver	
	Destination address	Specified by destination address register (DAR)		MSCI transmitter	Specified by DAR	MSCI transmitter		Specified by BAR	
	Transfer end condition	Normal end	The number of bytes of data specified in the byte count register (BCR) has been transferred.			One frame has been transferred.	The frame specified by the descriptor status field has been transferred.	One frame has been transferred.	—
	Error end	—			(1) A DMA transfer request is issued when the error descriptor address register (EDA) and current descriptor address register (CDA) match. (2) Frame-end interrupt-counter (FCT) overflows when it is enabled.				
Available MSCI Modes	—		Asynchronous, byte synchronous, or bit synchronous			Bit synchronous			

\*1 The operating mode is specified using the AMOD and TMOD bits of the DMA mode register A. For details, see section 6.2.8 "DMA Mode Register A."  
 \*2 I/O includes the MSCI. When the MSCI is specified as the source or destination, the request line is level sensitive. In this case, the DMS bit in the DMA mode register A (DMRA) should have been cleared in advance.  
 \*3 The DMAC issues transfer requests. Transfer starts immediately after the DMAC is enabled.  
 \*4 For details on burst and cycle steal transfers, see section 6.4.2 "Single-Block Transfers between Memory and Memory (Dual Address)."  
 \*5 For details on edge- or level-sensitive  $\overline{DREQ}$  input for transfer requests, see section 6.4.3 "Single-Block Transfers Between Memory and I/O (Memory-Mapped I/O) (Dual Address)."

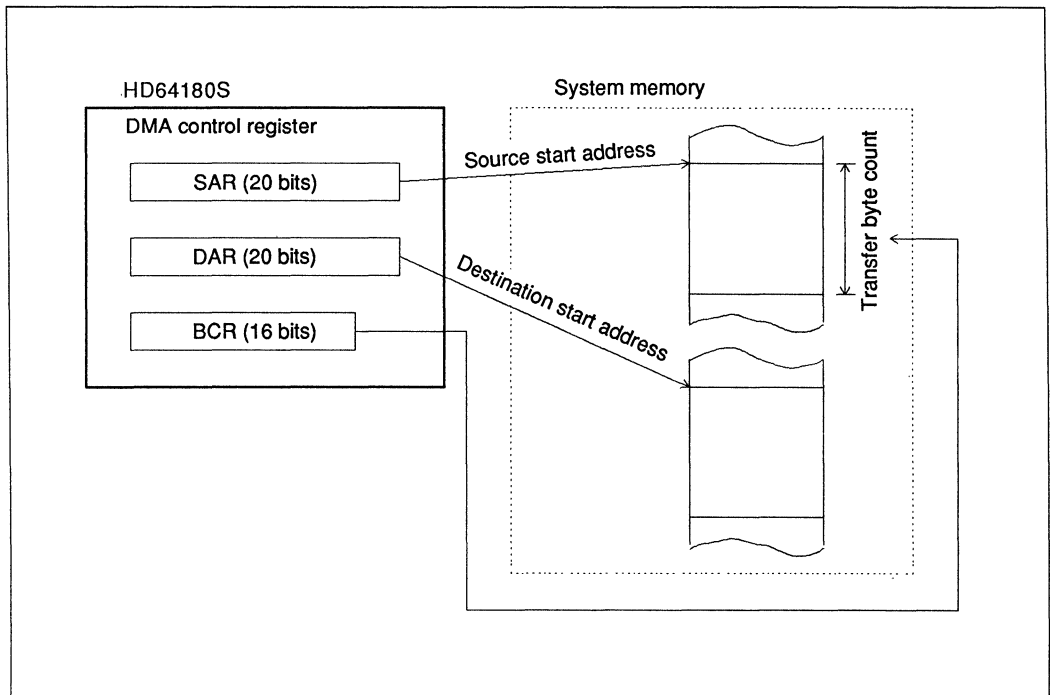


## 6.4.2 Single-block Transfers Between Memory and Memory (Dual Address)

### (1) Operation

For single-block transfers between memory and memory (dual address type), the source start address, destination start address, and transfer byte count must be loaded into the source address register (SAR), destination address register (DAR), and byte count register (BCR), respectively.

Figure 6-9 shows an example of a single-block transfer between memory and memory (dual address).



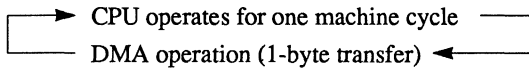
**Figure 6-9. Single-block Transfers Between Memory and Memory (Dual Address)**

The memory address data pointed to by SAR is DMA-transferred in byte units to the memory address pointed to by DAR. The number of bytes to be transferred is specified in BCR. BCR is decremented each time one byte of data is transferred. When BCR reaches 0000H, the DMAC stops transfer and enters the DMA initial state. At this time, an internal interrupt is generated (if enabled).

In the memory-to-memory single-block transfer mode (dual address), either burst transfer or cycle steal transfer can be selected.

In the burst mode, the DMA cycle lasts until the DMA transfer is complete (the byte count register reaches 0000H). The DMAC then passes control to the CPU.

In the cycle steal mode, the DMAC passes control to the CPU or another bus master\* after transferring each byte of data. The CPU executes one machine cycle, then passes control back to the DMAC as follows:



This is repeated until the transfer end conditions are satisfied.

\* See section 3.5 "Bus Arbiter."

## (2) Register setting

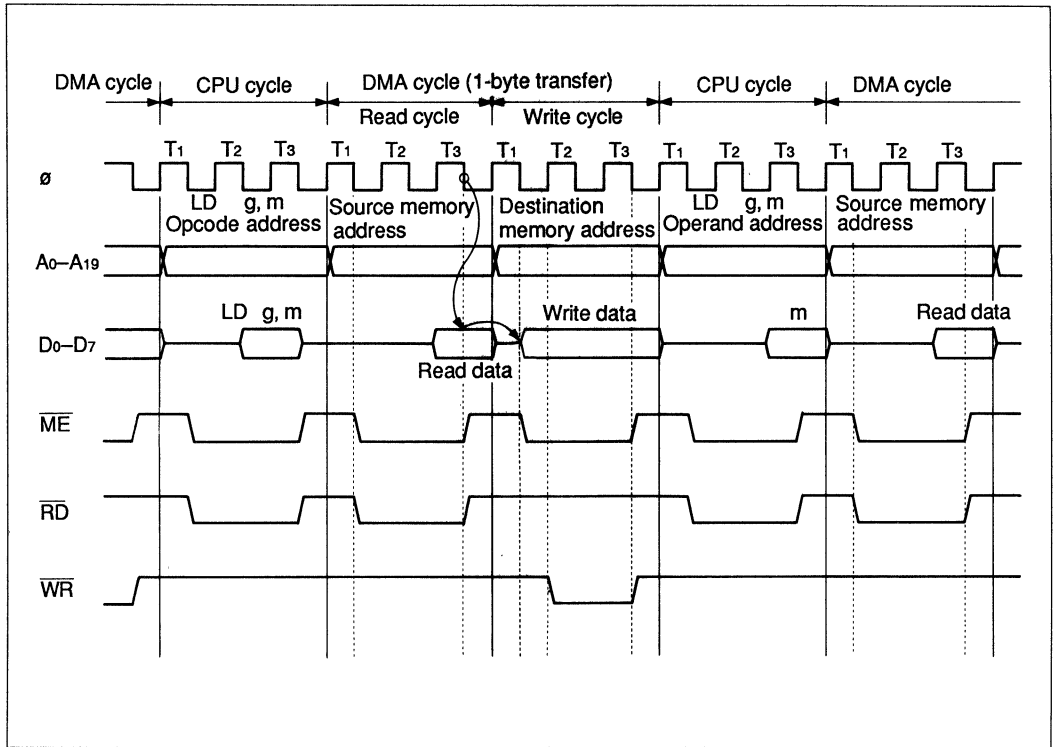
Follow the six steps listed below to begin a single-block transfer between memory and memory (steps ① to ⑥ do not have to be completed in sequence).

- ① Load the source start address into the SAR and the destination start address into the DAR.
- ② Load the transfer byte count into the BCR.
- ③ Select the single-block transfer mode (dual address) by setting the AMOD bit and TMOD bit in DMRA to 1 and 0, respectively, and clearing CNTE bit in DMRA.
- ④ Select the memory-to-memory transfer mode and the address increment/decrement by using the SM1-0 bits and the DM1-0 bits in DMRB for the source and destination, respectively.
- ⑤ Select either cycle steal or burst mode for the DMA using the MMOD bit in DMRB.
- ⑥ After completing steps ① to ⑤, set the DE bit to 1 (in DSR). After this write cycle is completed, 1 machine cycle of the next instruction is executed before DMA transfer starts.

If a bus cycle for a bus master other than the CPU is inserted immediately after the DSR write cycle, DMA transfer starts immediately after the bus cycle. Consequently, the next instruction is executed after the DMA transfer.

### (3) External bus timing

Figure 6-10 shows external bus timing for the memory-to-memory single-block transfer mode (dual address type) using cycle steal transfer.



**Figure 6-10. Memory-to-Memory Single-block Transfer (Dual Address)(Cycle Steal Mode)**

In this transfer mode, memory read and write cycles are successively executed to transfer one byte of data.

Notes on the timing for memory-to-memory single-block transfer mode:

- ① Transfer requests are issued by auto-request; the  $\overline{DREQ}$  line input has no effect on DMA transfers.
- ② Wait states can be inserted between the T2 and T3 states in each bus cycle (memory read and write cycles) using the  $\overline{WAIT}$  line or the wait control register.

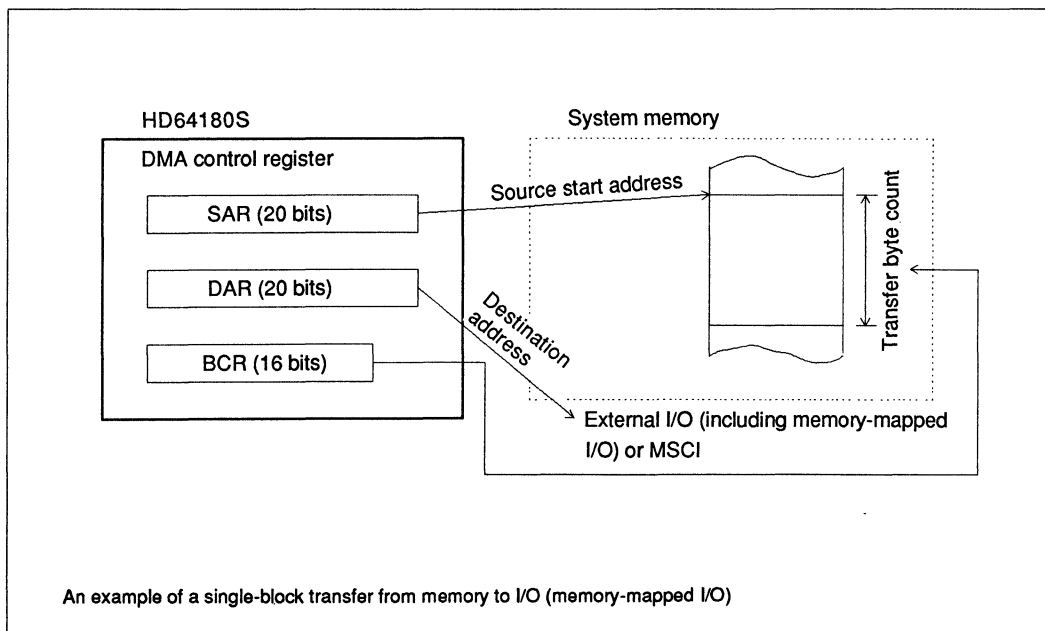
- ③ One T<sub>1</sub> clock cycle is inserted before the first byte of transfer.
- ④ In the last byte of the DMA transfer cycle (BCR = 0000H), the  $\overline{\text{TEND}}$  line is asserted from the rising edge of T<sub>2</sub> in a DMA read cycle to the end of the DMA write cycle. (The  $\overline{\text{TEND}}$  signal is output in the single-block transfer mode (dual address), independent of transfers between memory and memory and between memory and I/O (memory-mapped I/O)).

### 6.4.3 Single-block Transfers Between Memory and I/O (Memory-mapped I/O) (Dual Address)

#### (1) Operation

For single-block transfers between memory and I/O (memory-mapped I/O) (dual address type), the source start address, destination start address, and transfer byte count must be loaded into the source address register (SAR), destination address register (DAR), and byte count register (BCR), respectively.

Figure 6-11 shows an example of a single-block transfer from memory to I/O (memory-mapped I/O) (dual address type).



**Figure 6-11. Single-block Transfers between Memory and I/O (Memory-mapped I/O) Using Dual Address Type**

The memory address data set in SAR is DMA-transferred in byte units to the I/O (memory-mapped I/O) address set in DAR. The number of bytes to be transferred is specified in BCR. BCR is decremented each time one byte of data is transferred. The change in the address value to be set in SAR must be plus or minus one, and the DAR value must be fixed. When BCR reaches 0000H, the DMAC stops transfer and enters the DMA initial state. At this time, an internal interrupt is generated (if enabled).

For single-block transfers between memory and I/O (memory-mapped I/O), a transfer request from an external I/O device is received via the  $\overline{\text{DREQ}}$  line. The  $\overline{\text{DREQ}}$  line input can be level or edge sensitive (level sensitive for requests from the MSC1). The level- and edge-sensitive modes of the  $\overline{\text{DREQ}}$  line are explained below assuming that no bus requests are issued from bus masters (external I/O devices, refresh controller, and the other channel of the on-chip DMAC) other than the DMAC.

If the  $\overline{\text{DREQ}}$  line is level sensitive, DMA operation continues as long as the  $\overline{\text{DREQ}}$  line is low. When it goes high, control is passed to the CPU\*1 after the current data transfer is completed. The CPU then executes one machine cycle. If the  $\overline{\text{DREQ}}$  line is still high, the CPU executes another machine cycle.

While the  $\overline{\text{DREQ}}$  line is high, the CPU has control of the bus. When the  $\overline{\text{DREQ}}$  line goes low again, the CPU completes the current machine cycle, then passes control to the DMAC. The  $\overline{\text{DREQ}}$  signal is sampled, at the rising edge of the  $\phi$  clock one state before the T3 state in the DMA write cycle. Figure 6-12 shows how the  $\overline{\text{DREQ}}$  line level affects the CPU and the DMAC.

\*1 Bus masters other than CPU can have control of the bus. For details, see section 3.5 "Bus Arbiter."

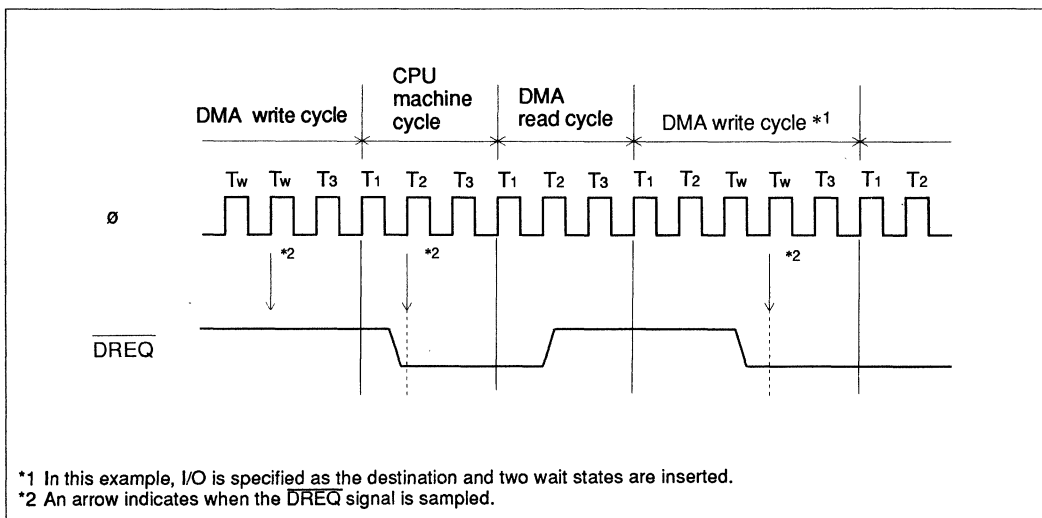
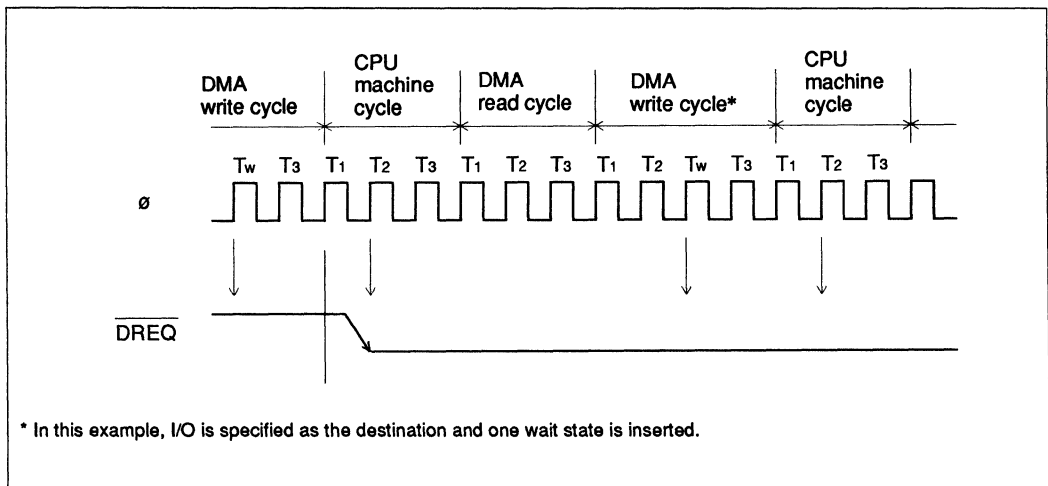


Figure 6-12. CPU and DMA Operation When the  $\overline{\text{DREQ}}$  Line is Level Sensitive

If the  $\overline{\text{DREQ}}$  input line is edge sensitive, DMA operation starts when a falling edge is detected on the  $\overline{\text{DREQ}}$  line. If another falling edge is detected between after the falling edge of the  $\phi$  clock T1 state in the DMA read cycle and before the rising edge of the  $\phi$  clock (one state before T3) in the write cycle, the DMAC continues operation. If another falling edge is not detected, control is passed to the CPU after the current transfer is completed. Unless a falling edge on the  $\overline{\text{DREQ}}$  line is detected prior to the rising edge of the  $\phi$  clock (one state prior to T3 or T1) during the CPU's machine cycle, the CPU continues operation. If an edge is detected, control is passed to the DMAC at the end of the current machine cycle. If two or more falling edges are inserted before the DMA transfer, only one transfer is performed.

Figure 6-13 shows how the  $\overline{\text{DREQ}}$  line level affects the CPU and the DMAC.



**Figure 6-13. CPU and DMA Operation When the  $\overline{\text{DREQ}}$  Line is Edge Sensitive**

## (2) Register setting

To start a single-block transfer (dual address) between memory and I/O (memory-mapped I/O) follow the 6 steps listed below (steps ① to ⑥ do not have to be completed in sequence).

- ① Load the memory start address into the SAR (or the DAR), and the I/O address in the DAR (or the SAR).
- ② Load the transfer byte count into the BCR.
- ③ Select the single-block transfer mode (dual address type) by setting the AMOD bit in DMRA and clearing the TMOD bit and the CNTE bit in the DMRA.

- ④ Select either the  $\overline{\text{DREQ}}$  line or MSCI as a DMA transfer requesting source using the RSEL 1 – 0 bits in DMRA. For the  $\overline{\text{DREQ}}$  input, specify whether edge or level sensitive using the DMS bit in DMRA and for MSCI, clear the DMS bit (level sensitive).
- ⑤ Specify the transfer direction whether memory-to-I/O (memory-mapped I/O) or I/O (memory-mapped I/O)-to-memory and also specify memory address increment/decrement using DM1-0 bits and SM1-0 bits in the DMA mode register B (DMRB).
- ⑥ After completing steps ① to ⑤, set the DE bit in DSR to 1. The DMAC will start DMA operation according to the request of the  $\overline{\text{DREQ}}$  or MSCI.

(3) External bus timing

Figure 6-14 (a) and (b) shows the external bus timing for single-block transfer between memory and I/O (memory-mapped I/O) (dual address).

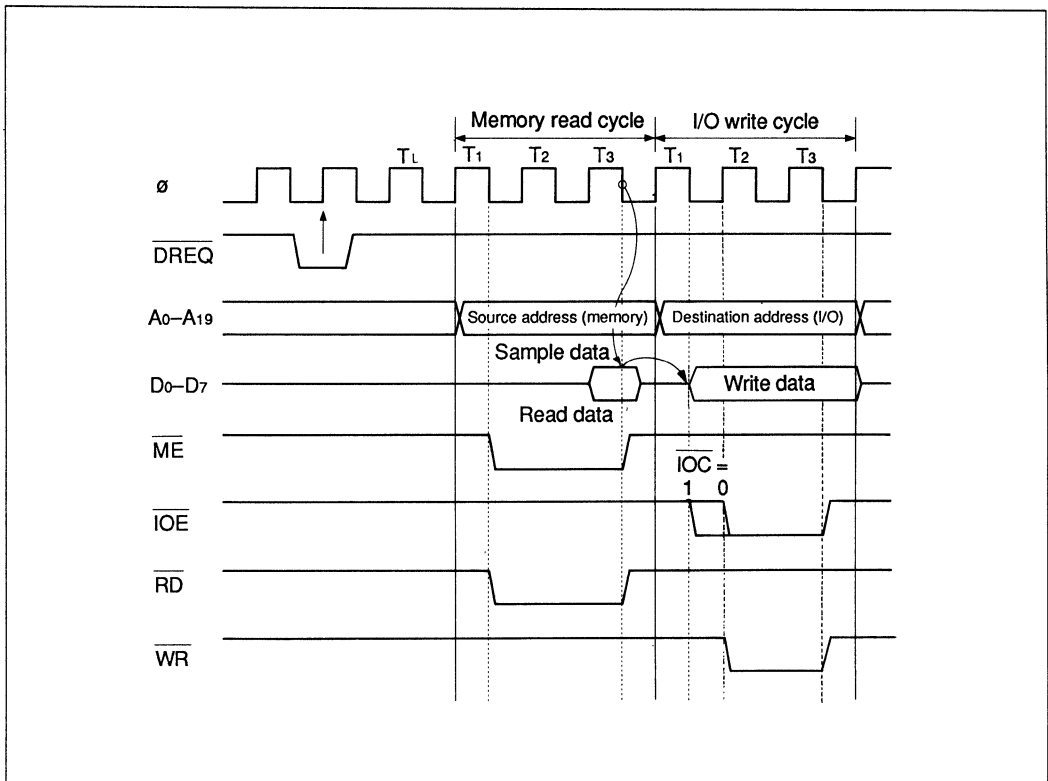
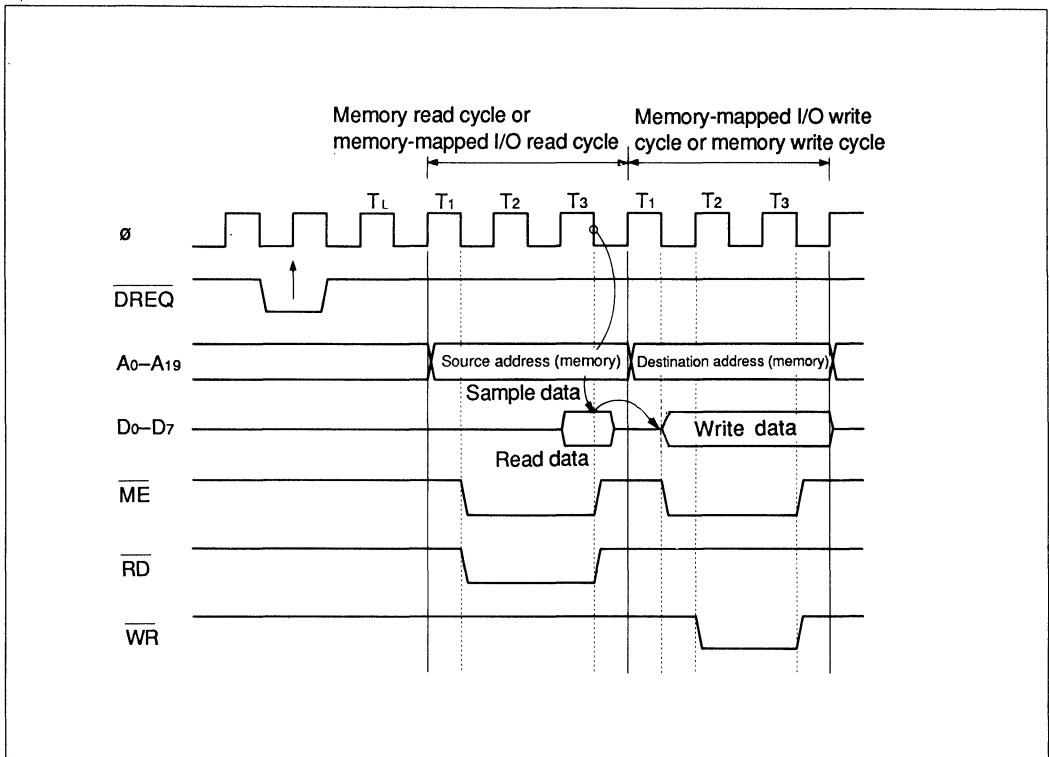


Figure 6-14. (a) Memory-to-I/O Single-block Transfer Mode (Dual Address)



**Figure 6-14. (b) Single-block Transfer between Memory and Memory-mapped I/O (Dual Address)**

In single-block transfers between memory and I/O (memory-mapped I/O), memory read cycle and I/O (memory-mapped I/O) write cycle, or I/O (memory-mapped I/O) read cycle and memory write cycle, are successively executed to transfer one byte of data.

Notes on the timing for this transfer mode:

①  $\overline{DREQ}$  line input sampling:

The level-sensitive  $\overline{DREQ}$  line is sampled at the rising edge of the  $\phi$  clock two states before the end of a machine cycle.

The edge-sensitive  $\overline{DREQ}$  line is sampled at the rising edge of each  $\phi$  clock. Requests are received when the  $\overline{DREQ}$  falling edge is detected before the  $\phi$  clock rising edge appears two states before the end of the machine cycle.



- ② Wait states can be inserted between the T<sub>2</sub> and T<sub>3</sub> states in each bus cycle (memory read/write cycle and I/O (memory-mapped I/O) read/write cycle) using the  $\overline{\text{WAIT}}$  line or the wait control register. (When the MSCI is specified as the I/O, wait states cannot be inserted to I/O cycle.)
- ③ One T<sub>i</sub> clock cycle is inserted before the first byte is transferred.
- ④ In the last one-byte DMA transfer cycle (BCR = 0000H), the  $\overline{\text{TEND}}$  line remains active from the rising edge of T<sub>2</sub> in a DMA read cycle to the end of the DMA write cycle, as shown in figure 6-15. (The  $\overline{\text{TEND}}$  signal is output in the single-block transfer mode (dual address), independent of memory-to-memory transfer or memory-to/from-I/O (memory-mapped-I/O) transfer.)

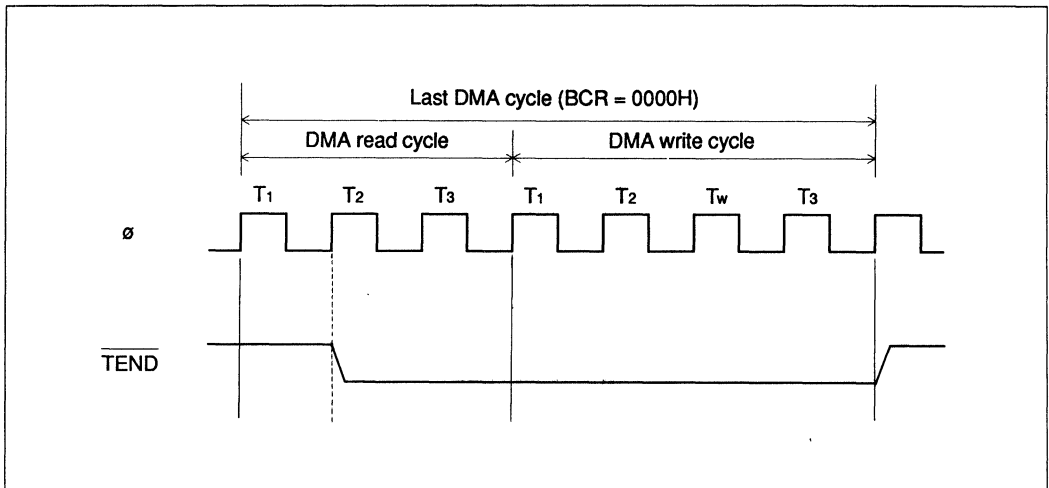


Figure 6-15.  $\overline{\text{TEND}}$  Output Timing

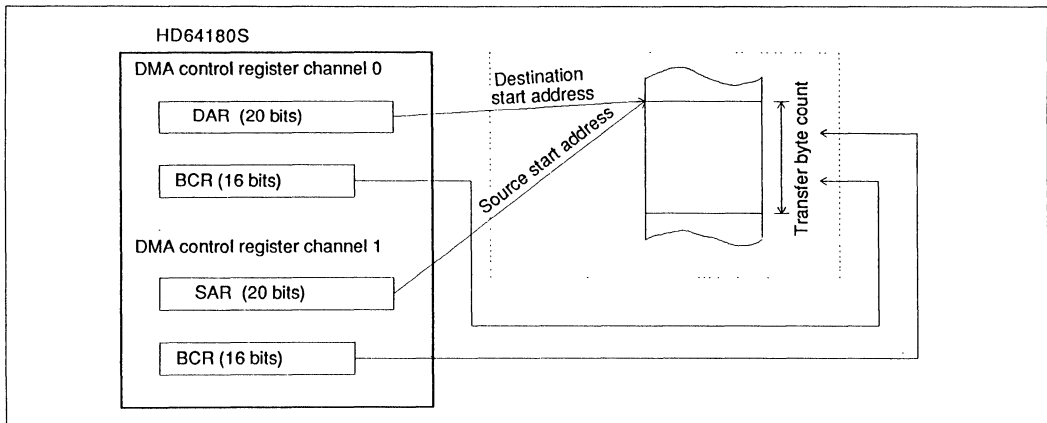
#### 6.4.4 Single-block Transfers Between Memory and the MSCI (Single Address)

##### (1) Operation

The HD64180S allows single-block transfers from the MSCI to memory using DMAC channel 0 and from memory to the MSCI using DMAC channel 1.

For single-block transfers from the MSCI to memory (single address), the destination start address and transfer byte count must be set in the DMAC channel 0 destination address register (DAR) and byte count register (BCR), respectively. For single-block transfers from memory to the MSCI (single address), the source start address and transfer byte count must be set in the DMAC channel 1 source address register (SAR) and byte count register (BCR), respectively.

Figure 6-16 shows single-block transfer between memory and MSCI (single address).



**Figure 6-16. Single-block Transfers between Memory and MSCI  
(Single Address)**

For MSCI-to-memory transfers, the number of bytes of data set in BCR is DMA-transferred in byte units from the MSCI receiver to the memory address specified in DAR of channel 0.

For memory-to-MSCI transfers, the number of bytes of data set in BCR is DMA-transferred in byte units from the memory address set in SAR of channel 1 to the MSCI transmitter.

BCR is decremented each time one byte of data is transferred. When BCR reaches 0000H, the DMAC stops transfer and enters the DMA initial state. At this time, an internal interrupt is generated (if enabled).

In the single-block transfers between memory and MSCI (single address), transfer requests are generated by a level-sensitive MSCI internal request signal.

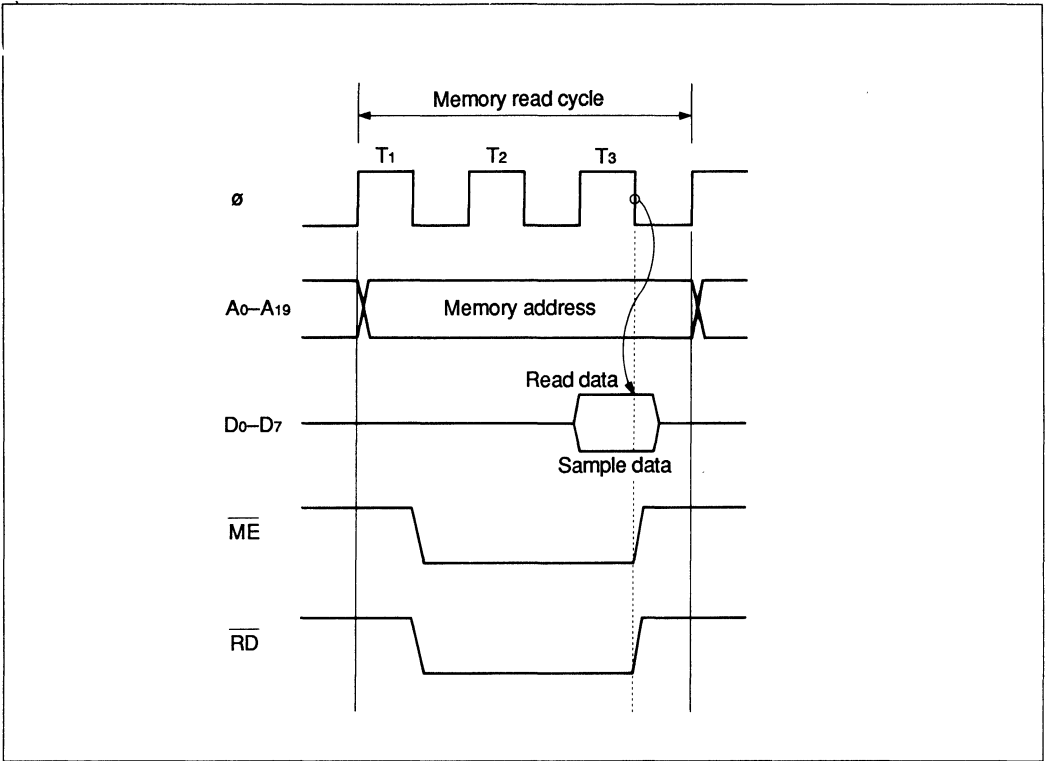
## (2) Register setting

To start a single-block transfer between memory and the MSCI, follow the five steps listed below starting with the DMA in its initial state (Steps ① to ⑤ do not have to be completed in sequence).

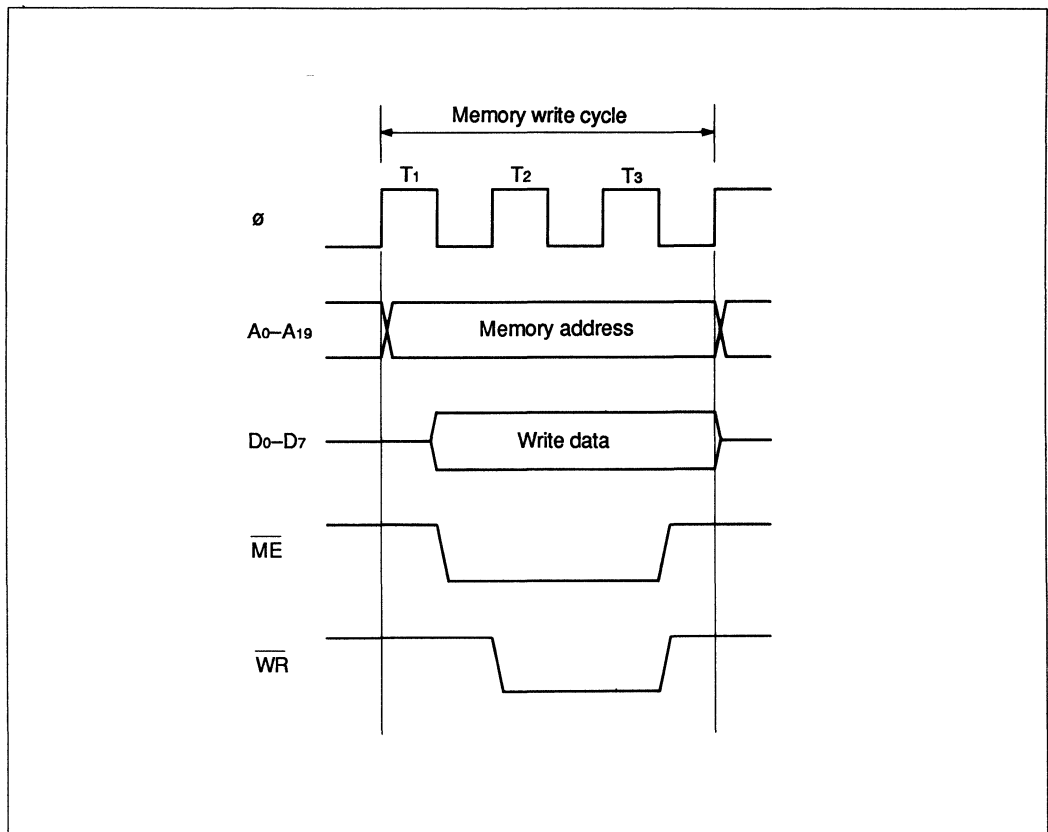
- ① For memory-to-MSCI transfers, load the memory start address of the source into the source address register (SAR). For MSCI to memory transfers, load the memory start address of the destination into the destination address register (DAR).
- ② Load transfer byte count into the byte count register (BCR).
- ③ Specify single-block transfer mode (single address) by clearing the AMOD, TMOD, and CNTE bits in DMRA.
- ④ Set the RSEL1 bit in DMA mode register A (DMRA) to 1 and clear the RSEL0 to specify the MSCI as the transfer requesting source. Clear the DMS bit in DMRA to select the level-sensitive request signal. Set the RT bit to 1 for memory-to-MSCI transfers and to 0 for MSCI-to-memory transfers.
- ⑤ After steps ① to ④ above, set the DE bit in DSR to 1 to start a DMA operation.

## (3) External bus timing

Figures 6-17 (a) and (b) show the external bus timing for single-block transfers between memory and MSCI (single address).



**Figure 6-17. (a) Memory-to-MSCI Single-block Transfer Mode (Single Address)**



**Figure 6-17. (b) MSCI-to-Memory Single-block Transfer Mode  
(Single Address)**

In single-block transfers between memory and the MSCI (single address), one byte of data transfer is completed in one memory read or write cycle. Accordingly, high-speed DMA transfer of 3 states/byte is possible.

Notes on the timing for this transfer mode:

- ① Transfer requests are initiated by MSCI's internal DMA request signals ( $\overline{DREQ}_R$ ,  $\overline{DREQ}_T$ )\*; the  $\overline{DREQ}$  line has no effect on DMA transfers.
- ② Wait states can be inserted between the T2 and T3 states in each bus cycle (memory read cycle and memory write cycle) using the  $\overline{WAIT}$  line or the wait control register.

\* See figure 6-24.

- ③ One  $T_i$  clock cycle is inserted before the first byte is transferred.
- ④ The  $\overline{TEND}$  line output is fixed at the high level and is not set to active level upon completion of transfer.

#### 6.4.5 Chained-block Transfers from Memory to the MSCI

##### (1) Operation

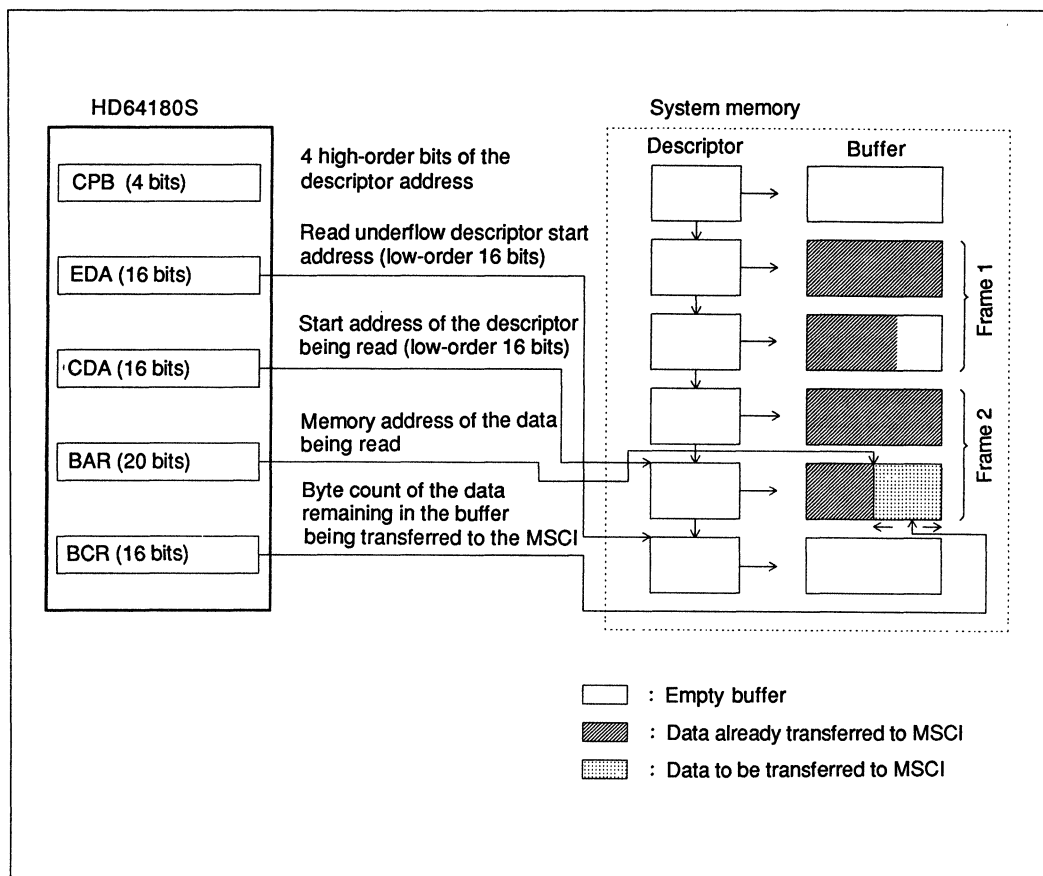
For chained-block transfers from memory to the MSCI\*, frames of data are transmitted from a system memory buffer to the MSCI operated in the bit synchronous mode using the single-address DMA transfer mode. Transfer requests are issued by using a level-sensitive MSCI internal request signal.

\* Chained-block transfer mode is not available when the MSCI is operated in the asynchronous or byte synchronous mode.

Memory-to-MSCI transfer employs DMAC channel 1. For this transfer mode, follow the steps below starting with the DMA in its initial state (Steps ① to ⑤ may be completed in any order).

- ① Specify the chained-block transfer mode using DMA mode register A (DMRA).
- ② Load the four high-order bits of the 20-bit descriptor address into the chain pointer base (CPB).  
Since the CPB value is fixed during operation, the descriptor can be assigned to any consecutive 64 kbyte area in system memory.
- ③ Load the start address (16 low-order bits) of the descriptor, which specifies the buffer next to the last transmit buffer, into the error descriptor address register (EDA).
- ④ Load the start address (16 low-order bits) of the descriptor which specifies the first transmit buffer into the current descriptor address register (CDA).
- ⑤ Specify the values for the chain pointer, buffer pointer, data length, and status in each descriptor.

Figure 6-18 shows an example of chained-block transfer from memory to the MSCI.



**Figure 6-18. Chained-block Transfer from Memory to the MSCI**

At the start of a DMA transfer, data in the buffer corresponding to the descriptor specified by CPB and CDA is transmitted to the MSCI transmitter. At this time, the DMAC writes the 20-bit memory address of the buffer being read to the buffer address register (BAR), and the number of bytes remaining to be read in the buffer to the byte count register (BCR). When the transfer starts, the DMAC writes the value of the buffer pointer of the corresponding descriptor to BAR and the data length value of the corresponding descriptor to BCR.

Each time one byte of data is transferred, BAR is incremented and BCR is decremented. When BCR equals 0000H, data transfer stops and the DMAC updates CDA to indicate the start address of the next descriptor, after which data is read from the buffer specified by the descriptor (the buffer is switched). In this way, by updating the descriptor, data in the buffers specified by the descriptors is transferred.

In the chained-block transfer mode, since transfer is performed in frame units, different frames cannot be saved in the same buffer. If a buffer contains the end of a frame, the EOM bit in the status field of the descriptor specifying the buffer must be set to 1. When single-frame transfer is specified, data up to the end of the frame in a buffer is transferred after which CDA is updated before completion of the DMA transfer. The descriptor, with the EOT bit of status set to 1, notifies the DMAC of the completion of data transfer after transferring the data in the specified buffer. This notification indicates the multiple frames transfer has completed.

At completion of frame or DMA transfer, the DMAC issues internal interrupt DMIB (if enabled).

EDA should initially contain the 16-bit low-order address of the descriptor specifying the first buffer which contains no transmit data. If data has been written to the buffer specified by the descriptor, the CPU updates EDA to indicate the start address of the descriptor specifying the next empty buffer. (EDA can be written to even when DMA is enabled.) This allows transmit data to be added and modified while DMA is enabled.

When the CDA and EDA values are equal and a transfer request is issued, the DMAC terminates transfer. At this time the DMAC issues internal interrupt DMIA (if enabled).

Figure 6-19 shows the operation flow in the memory-to-MSCI chained-block transfer mode. Table 6-6 lists the functions of the registers used for chained-block transfers from memory to the MSCI.

For memory-to-MSCI chained-block transfer mode, either a single-frame transfer or multiple-frame transfer can be selected. For single frame transfers, transfer is completed with one frame, after which the DMA initial state is re-entered. At the same time, the DE bit is automatically cleared. When the DE bit is set again, the DMAC restarts operation.



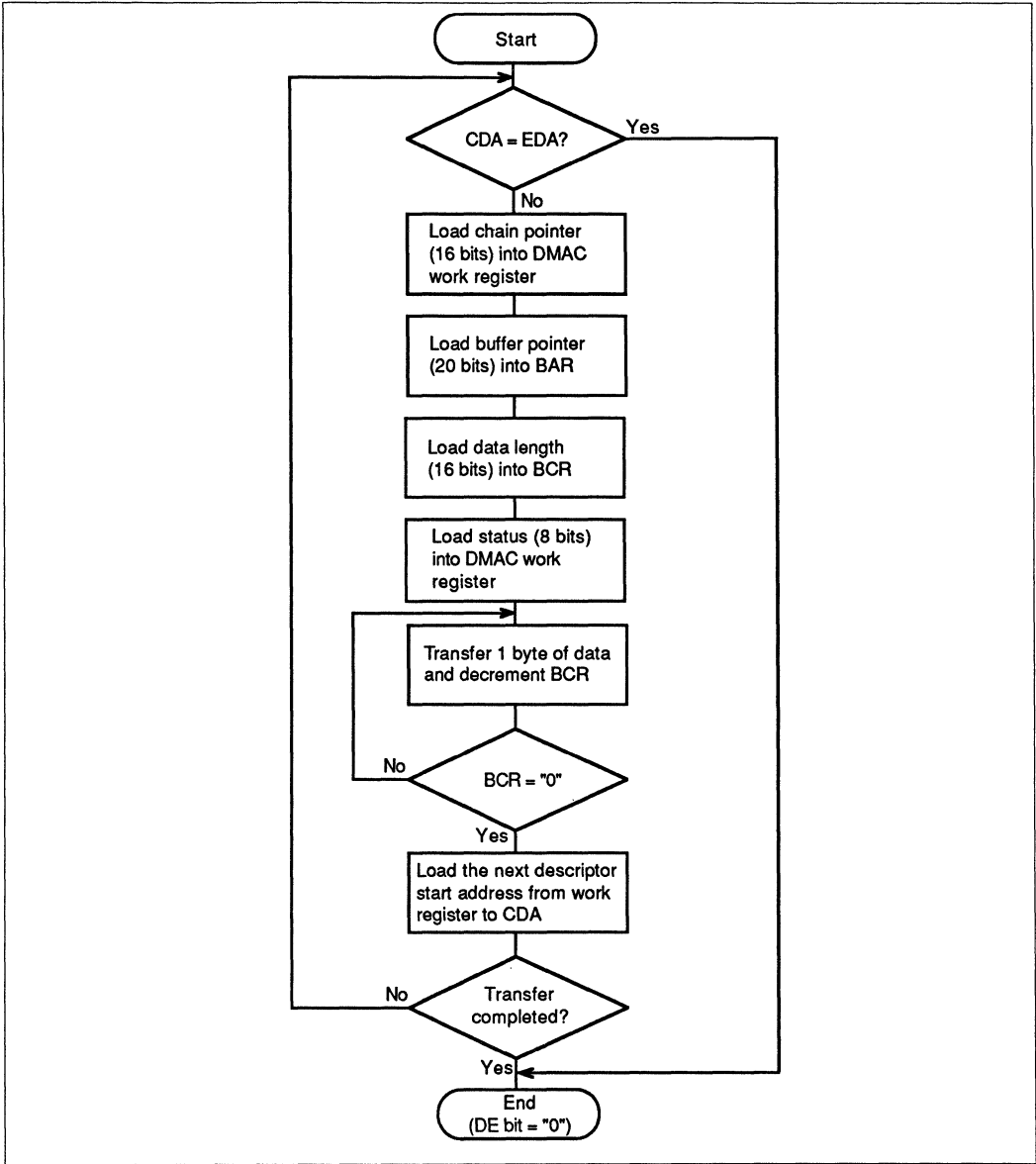


Figure 6-19. Chained-block Transfer from Memory to the MSCI

**Table 6-6. Control Registers Used for Chained-block Transfer from Memory to the MSCI (Transmit)**

<b>Register Name</b>	<b>Chain Pointer Base (CPB)</b>	<b>Error Descriptor Address Register (EDA)</b>	<b>Current Descriptor Address Register (CDA)</b>	<b>Receive Buffer Length (BFL)</b>	<b>Byte Count Register (BCR)</b>	<b>Buffer Address Register (BAR)</b>
Number of Bits	4	16	16	16	16	20
Function description	Specifies the four high-order bits of the 20-bit descriptor start address.	Specifies the 16 low-order bits of the start address of the descriptor preceding the last transmit buffer.	Specifies the 16 low-order bits of the start address of the descriptor corresponding to the first transmit buffer. This address is updated by the DMAC during buffer chaining.		Specifies the byte count of the data to be transferred to the MSCI. Writing to this register by the CPU is inhibited.	Specifies the system memory address of the data being transferred to the MSCI. Writing to this register by the CPU is inhibited.
Role in DMAC operation			After the DMAC starts, it loads the 16 low-order bits of the start address of the descriptor corresponding to the buffer being transferred into the MSCI.		When the contents of this register equal 0000H, reading from the current buffer is completed.	When a transfer request is issued, data is read from the address specified by this register.
<p>-----</p> <p>Transfer ends when a transfer request is issued while the EDA and CDA match. An internal interrupt, if enabled, is generated.</p>						

**Table 6-6. Control Registers Used for Chained-block Transfer from Memory to the MSC1 (Transmit) (cont.)**

Register Name	Chain Pointer Base (CPB)	Error Descriptor Address Register (EDA)	Current Descriptor Address Register (CDA)	Receive Buffer Length (BFL)	Byte Count Register (BCR)	Buffer Address Register (BAR)
Register update	[Under CPU control]	[Under CPU control]	When the current buffer read is completed, the next descriptor start address is automatically loaded into this register.		The contents of this register are decremented by 1 each time one byte is read. When the buffer is switched, the byte length specified by the descriptor is loaded.	The contents of this register are incremented by 1 each time one byte is read. When the buffer is switched, the next buffer start address is loaded.
Register updated by the CPU	This register should be initialized before transmission.	The start address of the descriptor indicating the buffer following the last buffer containing transmit data is loaded. To add transmit data during a transmission, load the start address of the descriptor indicating the next buffer to be written.	The start address of the descriptor indicating the first buffer containing transmit data is loaded before transmission starts.			

Table 6-7 shows a memory-to-MSCI chained-block single-frame transfer using four descriptors and buffers. In this example, data is not added to the buffers during transmission. Frame 1 DMA operation ends after operations ① to ⑥ at which the DMAC enters the DMA initial state. The transfer control register value is retained and thus frame 2 DMA transfer subsequently starts by setting the DE bit. When frame 2 transfer is completed, CDA and EDA contents are equal. Accordingly, transfer is not performed even if an additional request is issued from the MSCI, and an internal interrupt DMIA is generated (if enabled).

Table 6-8 shows memory-to-MSCI chained-block multiple-frame transfer using four descriptors and buffers. In this example, data is added to the buffer during transmission. After operations ① and ②, additional transmit data is written to buffers 2 and 3 by the CPU. At the same time, EDA is also updated to the start address of the descriptor indicating buffer 0. In this way, the data in buffers 2 and 3 is transferred following buffer 1 data.

Since the DMAC remains active after one frame has been transferred in multiple-frame transfer, there might exist multiple DMIB internal interrupts (frame end interrupts) which have not yet been serviced. The number of unserviced interrupts is stored in the frame-end interrupt-counter (FCT). When the FCT value is 1111 and frame transfer continues, counter overflow occurs and the DMAC terminates the transfer after transmitting the current frame. The FCT value is then reset to 0000, and an internal DMIA interrupt is generated (if enabled). For details, see sections 6.2.8 "DMA Mode Register A" and 6.2.10 "Frame-End Interrupt-Counter."

**Table 6-7. Chained-block Transfer from Memory to MSCI (single frame transfer)**  
(Normal transmit operation)

Step	DMAC Operation	CPU Operation	CDA Contents	EDA Contents	DE Bit Value	Remarks
①	—	A0 → CDA A3 → EDA 1 → DE bit	A0	A3	1	CDA specifies the first buffer address containing transmit data and EDA specifies the buffer after the last one to be transmitted.
②	Read data from buffer 0	—	A0	A3	1	

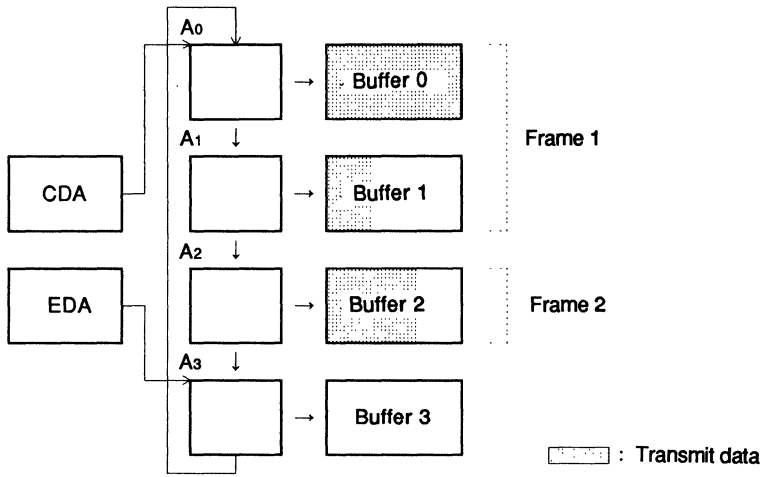
**Table 6-7. Chained-block Transfer from Memory to MSCI (single frame transfer) (cont.)**

(Normal transmit operation)

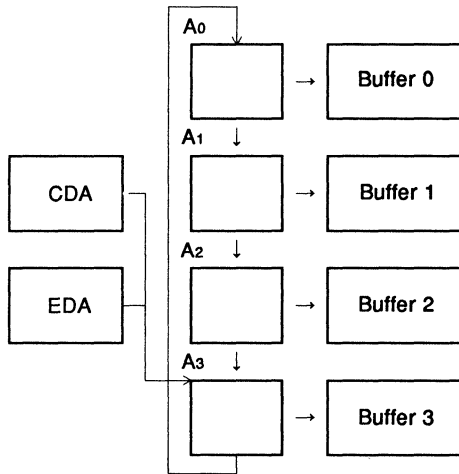
Step	DMAC Operation	CPU Operation	CDA Contents	EDA Contents	DE Bit Value	Remarks
③	A1 → CDA	–	A1	A3	1	
④	Read data from buffer 1	–	A1	A3	1	
⑤	A2 → CDA 0 → DE bit	–	A2	A3	0	After the transfer of one frame, the transfer ends and the DE bit is cleared. When 1 is written to the DE bit, the DMAC can accept a transfer request.
⑥	–	1 → DE bit	A2	A3	1	
⑦	Read data from buffer 2	–	A2	A3	1	
⑧	A3 → CDA 0 → DE bit	–	A3	A3	0	If 1 is written to the DE bit and a transfer request is accepted, a DMIA interrupt is generated.

(An: Start address of each descriptor )

• Status after step ①



• Status after step ②



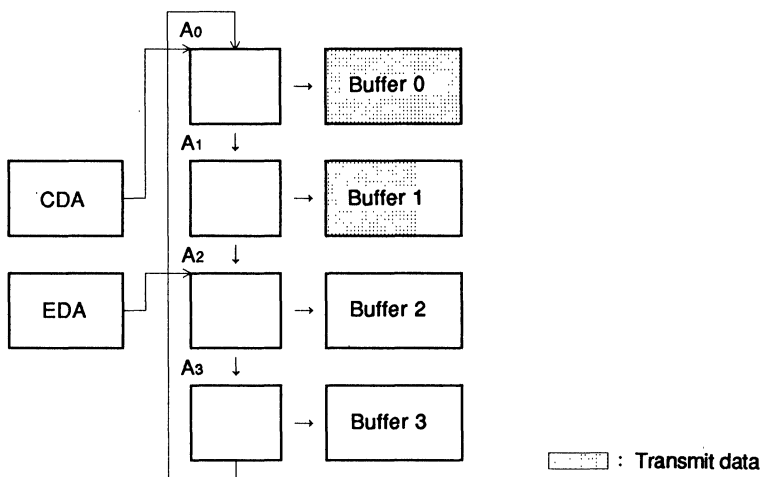
**Table 6-8. Chained-block Transfer from Memory-to-MSCI (multi-frame transfer)**

(When adding transmit data during transmission)

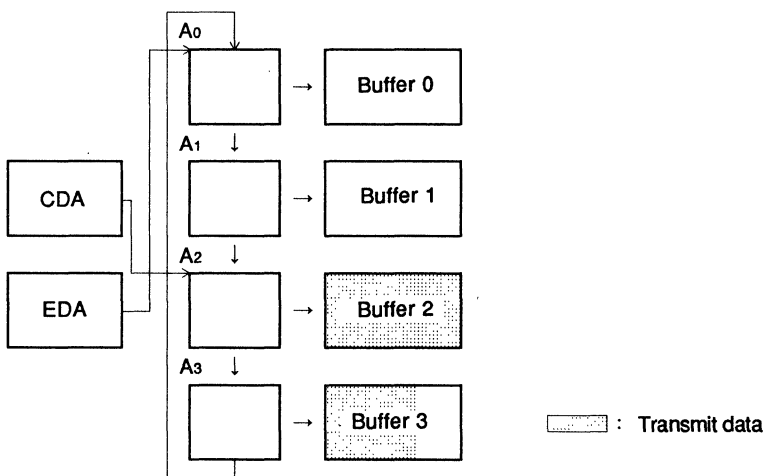
Step	DMAC Operation	CPU Operation	CDA Contents	EDA Contents	DE Bit Value	Remarks
①	–	A0 → CDA A2 → EDA 1 → DE bit	A0	A2	1	CDA specifies the buffer containing the transmit data.
②	Reads data from buffer 0	–	A0	A2	1	
③	A1 → CDA	–	A1	A2	1	
④	–	Loads transmit data into buffer 2. A3 → EDA	A1	A3	1	Adds transmit data to the buffer and rewrites EDA.
⑤	–	Loads transmit data into buffer 3. A0 → EDA	A1	A0	1	
⑥	Reads data from buffer 1.	–	A1	A0	1	
⑦	A2 → CDA	–	A2	A0	1	

(An: Start address of each descriptor)

• Status after step ①



• Status after step ②



(2) Register, descriptor setting

For memory-to-MSCI chained-block transfer mode, follow the steps below starting with the DMA in its initial state (Steps ① to ⑥ may be completed in any order).



① Generate an arbitrary number of descriptors in any system area (64 kbytes or less)\*1. Specify a 16-bit chain pointer, 20-bit buffer pointer, 16-bit data length and status EOM and EOT bits in each descriptor.\*2

\*1 Since the 4 high-order bits of the 20-bit address are specified by the CPB, the 4 high order bits are common to a 64 kbyte area.

\*2 Descriptors may be specified during the DMA halt state.

② Set the bits in the DMA mode register A(DMRA) as follows:

RSEL1 = 1, RSEL0 = 0, AMOD = 0, TMOD = 1, RT = 1, and DMS = 0

③ Clear the NF bit in DMRA for single-frame transfer and set the NF bit for multiple-frame transfer.

④ Load the four high-order bits of the 20-bit descriptor address into the chain pointer base (CPB).

⑤ Load the start address (16 low-order bits) of the descriptor corresponding to the buffer next to the last transmit buffer into the error descriptor address register (EDA).

⑥ Load the start address of the descriptor corresponding to the first transmit buffer into the current descriptor address register (CDA).

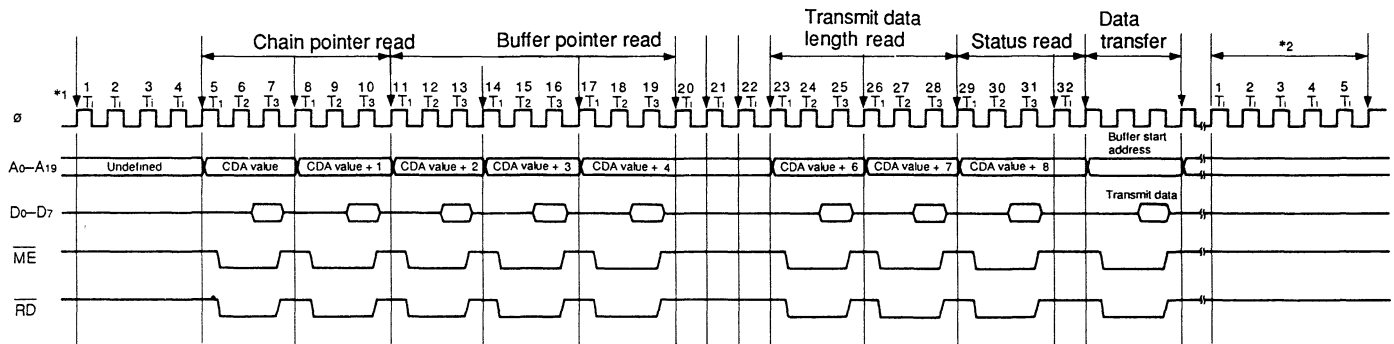
⑦ After completing steps ① to ⑥, set the DE bit to 1 to start DMA operation.

### (3) External bus timing

For memory-to-MSCI chained-block transfer mode, one byte of data transfer is completed in one memory read cycle. The memory read cycle timing is the same as that for the memory-to-MSCI single-block transfer mode (single address) shown in figure 6-17 (a).

Prior to the start of DMA transfer and during buffer switching, this transfer mode requires special cycles for the DMAC to read a descriptor and perform other operations, as shown in figure 6-20. At the start of a DMA transfer, 32 states are inserted.

During buffer switching, one internal state indicated by[\*2] (for the middle of a frame) or five states (for the end of a frame) are inserted. This is followed by 32 states in which the next descriptor is read.



\*1 Downward arrows "↓" indicate where another bus master cycle can be inserted.

\*2 One cycle for the middle of a frame and five cycles for the end of a frame.

**Figure 6-20. Memory-to-MSCI Chained-block Transfer Timing  
(for the Start of Transfer and Buffer Switching)**

## 6.4.6 Chained-block Transfers from the MSCI to Memory

### (1) Operation

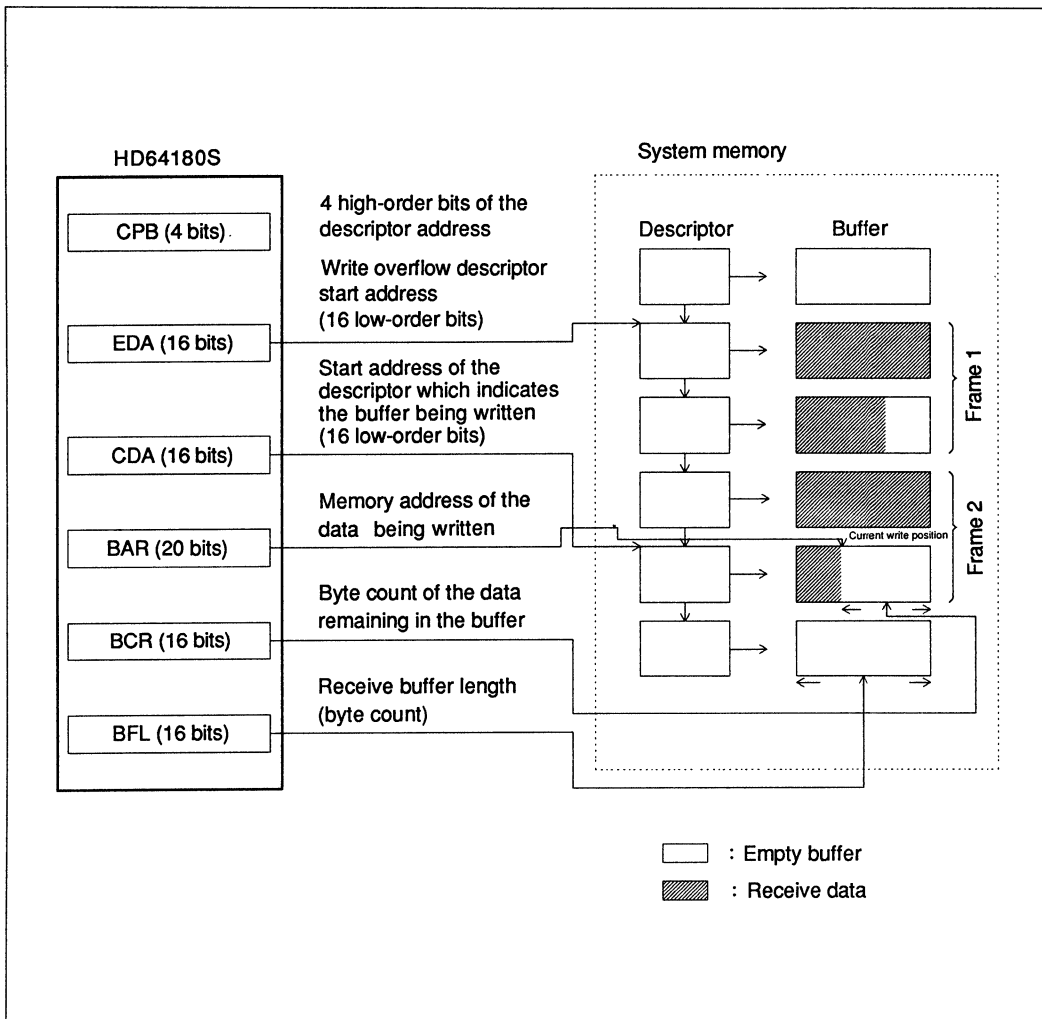
For chained-block transfers from the MSCI to memory, frame-boundary data received by the MSCI receiver (in the bit synchronous mode) is DMA-transferred in byte units using the single address mode. Transfer requests are initiated by the MSCI using a level-sensitive internal request signal.

The chained-block transfer mode is not available when the MSCI is operated in the asynchronous or byte synchronous mode.

MSCI-to-memory transfer employs DMAC channel 0. For this transfer mode, follow the steps below starting with the DMA in its initial state (Steps ① to ⑥ can be completed in any order).

- ① Specify the chained-block transfer mode using the DMA mode register A (DMRA).
- ② Load the four high-order bits of the 20-bit descriptor address into the chain pointer base (CPB).  
Since the CPB value is fixed during operation, the 4 high order bits specify a common 64 kbyte area in system memory.
- ③ Load the start address (16 low-order bits) of the descriptor, which specifies the buffer next to the last write-enabled buffer, into the error descriptor address register (EDA).
- ④ Load the start address (16 low-order bits) of the descriptor indicating the first buffer in which receive data should be written into the current descriptor address register (CDA).
- ⑤ Load the buffer length in bytes into the receive buffer length (BFL). This value is shared by all buffers.
- ⑥ Specify the values for the chain pointer and buffer pointer for each descriptor.

Figure 6-21 shows an example of a chained-block transfer from the MSCI to memory.



**Figure 6-21. Chained-block Transfer from the MSC1 to Memory**

At the start of a DMA transfer, receive data is transferred from the MSC1 receiver to the buffer corresponding to the descriptor specified by CPB and CDA. At this time, the DMAC writes the 20-bit memory address of the buffer being written to the buffer address register (BAR), and the number of bytes remaining to be written in the buffer to the byte count register (BCR). When the transfer starts, the DMAC writes the value of the buffer pointer of the corresponding descriptor to BAR and the value of BFL to BCR.

Each time one byte of data is transferred, BAR is incremented and BCR is decremented. When BCR equals 0000H, data transfer stops and the DMAC writes the receive data length to the descriptor and updates CDA to point to the starting address of the next descriptor (the buffer is switched). At that time, both BAR and BCR are also updated by the DMAC. Thus, the buffer pointer value of the descriptor is written to BAR and the value of BFL is written to BCR. In this way, by updating the descriptor, transfer of data in the buffers specified by the descriptors is accomplished.

If the end of a frame is detected in the buffer being written, the buffer is immediately switched, and the DMAC writes the MSCI frame status register (MFST) value, which is obtained immediately after the data is transferred, into the status area of the corresponding descriptor. (At this time, the receive data length is also written.) For single-frame transfers, DMA transfer ends when CDA is updated. For multiple-frame transfers, the buffer is switched and CDA, BAR, and BCR are updated after which the next buffer is written.

At the completion of frame transfer, the DMAC issues internal interrupt DMIB (if enabled).

EDA initially contains the 16-bit low-order address of the descriptor indicating the first buffer which is disabled for receive data writing. By updating EDA, buffers can be accessed even while DMA is enabled. At this time, EDA should specify the starting address of the descriptor indicating the buffer next to the last write buffer.

If the CDA and EDA values are equal when the transfer request is issued, the DMAC terminates transfer and issues internal interrupt DMIA (if enabled).

Figure 6-22 shows MSCI-to-memory chained-block transfer operation flow.

Table 6-9 lists the functions of the registers used for chained-block transfers from MSCI to memory.

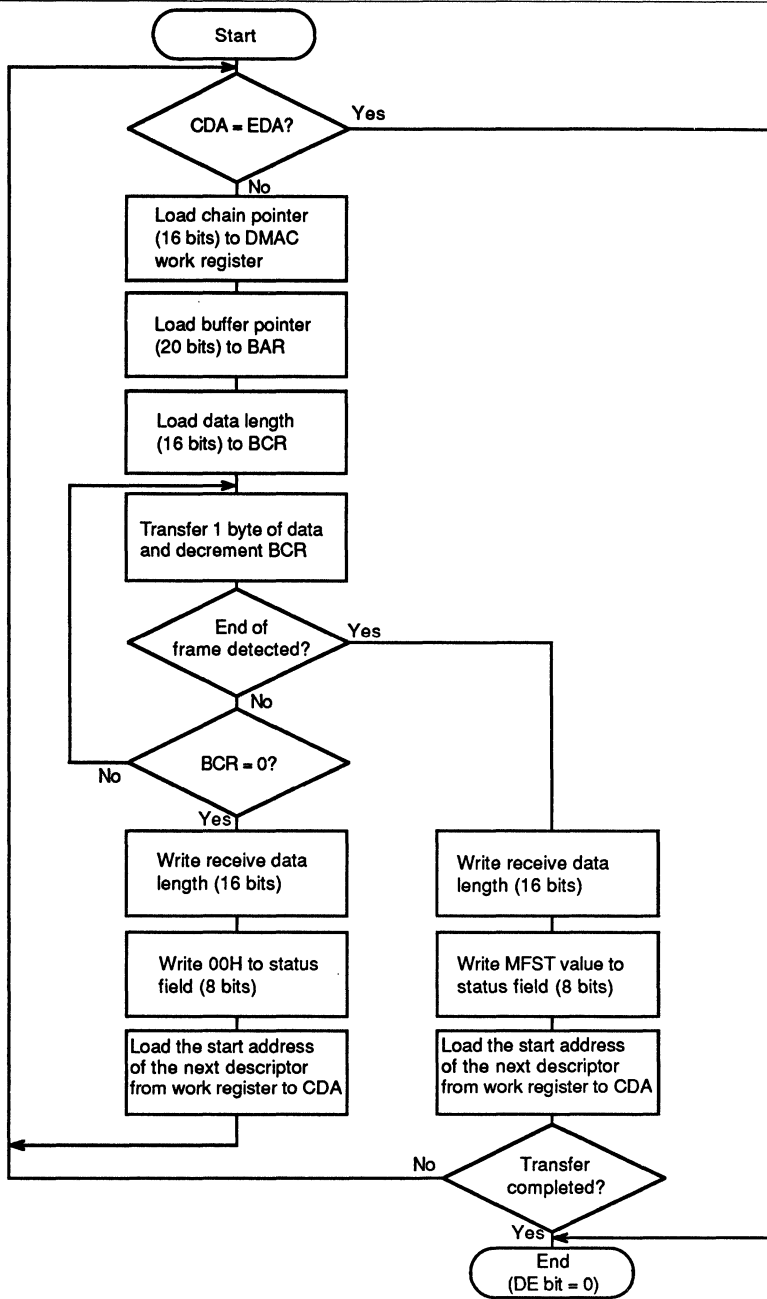


Figure 6-22. Chained-block Transfer from the MSC1 to Memory

**Table 6-9. Control Registers Used for Chained-block Transfers from the MSCI to Memory (Receive)**

Register Name	Chain Pointer Base (CPB)	Error Descriptor Address Register (EDA)	Current Descriptor Address Register (CDA)	Receive Buffer Length (BFL)	Byte Count Register (BCR)	Buffer Address Register (BAR)
Number of Bits	4	16	16	16	16	20
Functional description	Specifies the four high-order bits of the 20-bit descriptor start address.	Indicates the 16 low-order bits of the start address of the descriptor following the descriptor indicating the last write-enabled buffer.	Specifies the 16 low-order start address of the descriptor corresponding to the first receive buffer. This address is updated by the DMAC during buffer chaining.	Indicates the the buffer length in bytes.	Indicates the byte count of the data remaining in the buffer waiting to be written to memory. Writing to this register by the CPU is inhibited.	Indicates the system memory address of the data being loaded into the buffer. Writing to this register by the CPU is inhibited.
Role in DMAC operation			When the DMAC begins receive operation, indicates the 16 low-order bits of the start address of the descriptor corresponding to the buffer being written.		When the contents of this register equal 0000H, writing to the current buffer stops.	When a transfer request is issued, data is loaded into the address specified by this register.
		----- Transfer ends when a transfer request is issued while the EDA and CDA match. An internal interrupt, if enabled, is generated.				

**Table 6-9. Control Registers Used for Chained-block Transfers from the MSC1 to Memory (Receive) (cont.)**

Register Name	Chain Pointer Base Address (CPB)	Error Descriptor Base Address Register(EDA)	Current Descriptor Address Register (CDA)	Receive Buffer Length (BFL)	Byte Count Register (BCR)	Buffer Address Register (BAR)
Register update	[Under CPU control]	[Under CPU control]	When the current buffer write is completed, the next descriptor start address is automatically loaded into this register.	[Under CPU control]	The contents of this register are decremented by 1 each time one byte is written. When the buffer is switched, the BFL value is loaded.	The contents of this register are incremented by 1 each time one byte is written. When the buffer is switched, the next buffer start address is loaded.
Register updated by the CPU	Initial setting before reception starts.	Specifies the start address of the descriptor indicating the buffer following the last write of the buffer. When releasing the buffer, this register indicates the start address of the descriptor for the buffer following the one being released.	When reception begins, indicates the start address of the descriptor which indicates the buffer to be written.	Initial setting		



For MSCI-to-memory chained-block transfer, either single-frame transfer or multi-frame transfer can be selected.

For single-frame transfer, transfer is completed with the transfer of one frame, after which the DMA returns to initial states. At the same time, the DE bit is automatically cleared. When the DE bit is set again, the DMAC restarts operation. Multi-frames are subsequently transferred if a request is issued from the MSCI. When CDA and EDA match, transfer is terminated even if an additional transfer request has been issued.

Table 6-10 shows a typical MSCI-to-memory chained-block multi-frame transfer using four descriptors and buffers. In this example, after a transfer begins, CDA is updated and then the CDA initial value is written to EDA since transfer is disabled when CDA and EDA are equal. As a result, the write-enabled buffer size is maximized. In this example, the CDA and EDA value match after frame 2 has been transferred (operation ⑨). At this time, any additional transfer request is disabled and internal interrupt DMIA is generated (if enabled).

Table 6-11 shows another example of MSCI-to-memory multi-frame transfer using four descriptors and buffers. In this example, in order to rewrite a buffer, the received data saved in the buffer is moved to another area during receive operations and EDA is updated. Operations ① to ⑦ are the same as those in table 6-10.

Since the DMAC remains active after one frame transfer in multi-frame transfer, multiple DMIB internal interrupts (frame end interrupt) might exist which have not yet been serviced. The number of unserviced interrupts is stored in the frame-end interrupt-counter (FCT). When the FCT value is 1111 and frame transfer continues, counter overflow occurs and the DMAC terminates the transfer after transmitting the current frame. The FCT value is then reset to 0000, and internal interrupt DMIA is generated (if enabled). For details, see sections 6.2.8 "DMA Mode Register A" and 6.2.10 "Frame-End Interrupt-Counter."

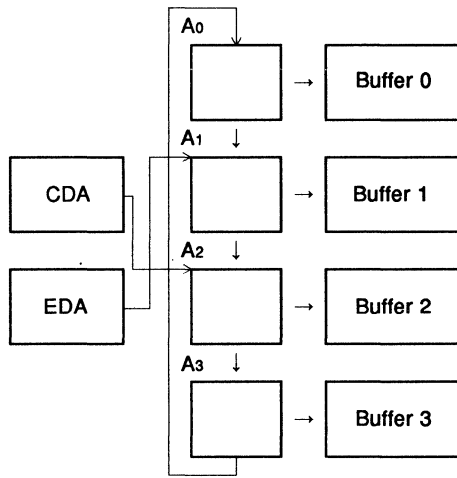
**Table 6-10. Chained-block Transfer from MSCI to Memory (multi-frame transfer)**

(Normal receive operation)

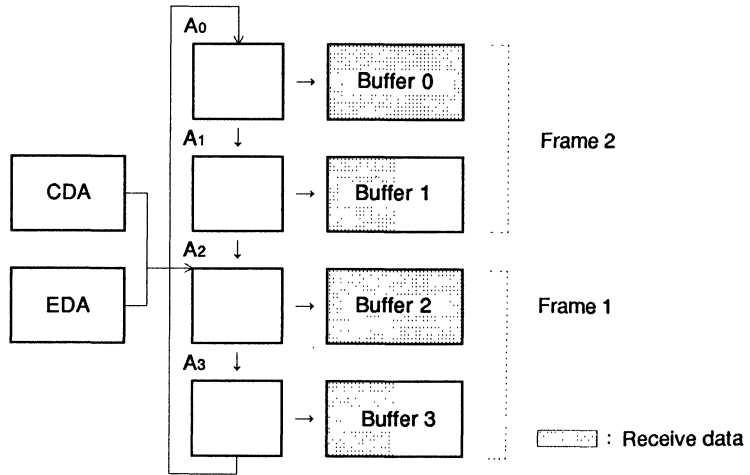
Step	DMAC Operation	CPU Operation	CDA Contents	EDA Contents	DE bit Value	Remarks
①	–	A2 → CDA A1 → EDA 1 → DE bit	A2	A1	1	CDA and EDA specify the buffer where receive data is to be written.
②	Writes data to buffer 2.	–	A2	A1	1	
③	A3 → CDA	A2 → EDA	A3	A2	1	A2 is written to EDA to reserve the maximum buffer size.
④	Writes data to buffer 3.	–	A3	A2	1	
⑤	A0 → CDA	–	A0	A2	1	
⑥	Writes data to buffer 1.	–	A1	A2	1	
⑦	A2 → CDA	–	A2	A2	1	If another write request is accepted in this state, the DMAC generates a DMIA interrupt.

(An: Start address of each descriptor)

• Status after step ①



• Status after step ②



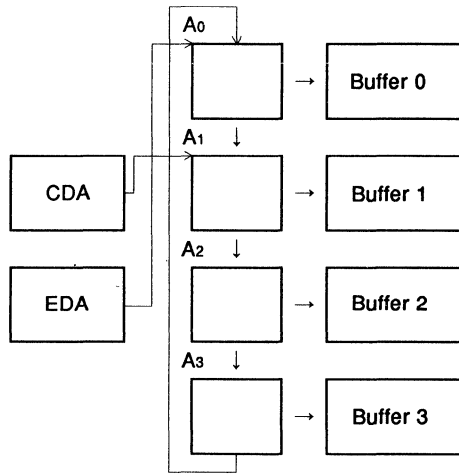
**Table 6-11. Chained-block Transfer from MSC1 to Memory (multi-frame transfer)**

(Releasing part of a buffer during a receive operation )

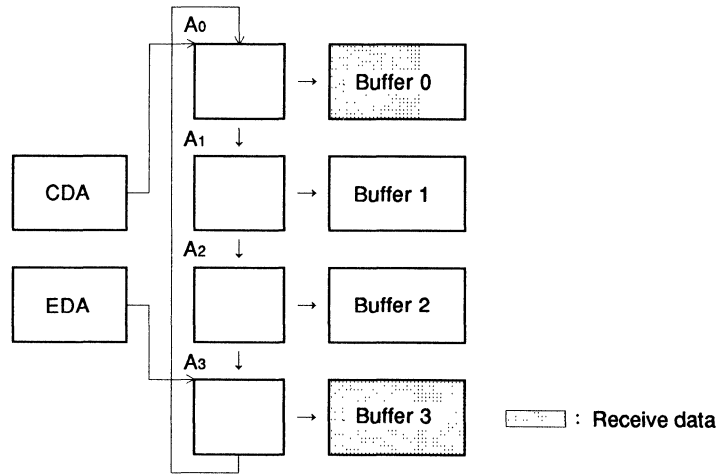
Step	DMAC Operation	CPU Operation	CDA Contents	EDA Contents	DE bit Value	Remarks
①	—	A1 → CDA A0 → EDA 1 → DE bit	A1	A0	1	CDA specifies the buffer where the receive data is to be written.
②	Writes data to buffer 1.	—	A1	A0	1	
③	A2 → CDA	A1 → EDA	A2	A1	1	A1 is written to EDA to reserve the maximum buffer size.
④	Writes data to buffer 2.	—	A2	A1	1	
⑤	A3 → CDA	—	A3	A1	1	
⑥	Writes data to buffer 3.	—	A3	A1	1	
⑦	A0 → CDA	—	A0	A1	1	
⑧	—	Transfers data from buffers 1 and 2 to another area.	A0	A1	1	After transferring receive data to another area, EDA is rewritten by CPU to release the buffer.
⑨	—	A3 → EDA	A0	A3	1	
⑩	Writes data to buffer 0.	—	A0	A3	1	

(An: Start address of each descriptor)

• Status after step ①



• Status after step ②



(2) Register, descriptor setting

For MSCI-to-memory chained-block transfer mode, follow the steps below starting with the DMA in its initial state (Steps ① to ⑦ can be completed in any order).

- ① Generate an arbitrary number of descriptors in any system area (64 kbytes or less)\*1 . Specify a 16-bit chain pointer and a 20-bit buffer pointer for each descriptor.\*2
- ② Set the bits in the DMA mode register A (DMRA) as follows:  
RSEL1 = 1, RSEL0 = 0, AMOD = 0, TMOD = 1, RT = 0, and DMS = 0.
- ③ Clears the NF bit in DMRA for single-frame transfer and sets the bit for multi-frame transfer.
- ④ Load the 4 high-order bits of the 20-bit descriptor into the chain pointer base (CPB).
- ⑤ Load the start address (16 low-order bits) of the descriptor corresponding to the buffer next to the last write-enabled buffer into the error descriptor address register (EDA).
- ⑥ Load the start address of the descriptor corresponding to the first receive buffer into the current descriptor address register (CDA).
- ⑦ Specify buffer length in byte units in the receive buffer length (BFL).
- ⑧ After completing steps ① to ⑦, set the DE bit to 1 to start the DMA operation.

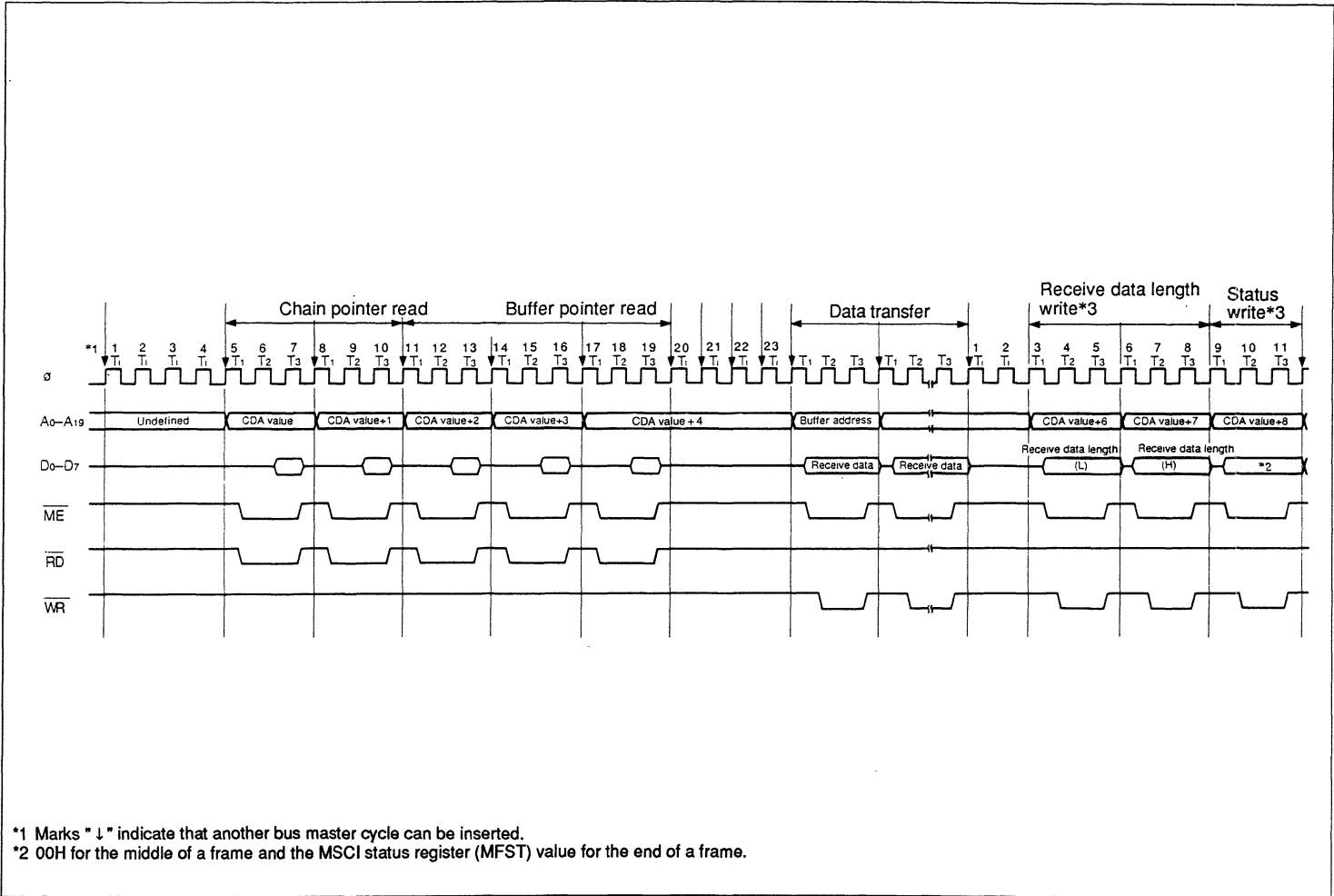
\*1 Since the 4 high-order bits of the 20-bit address are specified by the CPB, the 4 high-order bits are common to a 64 kbyte address.

\*2 Descriptors may be specified during a DMA halt state.

### (3) External bus timing

For MSCI-to-memory chained-block transfer mode, one byte of data transfer is completed in one memory write cycle. The memory write cycle timing is the same as that for MSCI-to-memory single-block transfer mode (single address) shown in figure 6-17 (b).

Prior to the start of DMA transfer, and during buffer switching, this transfer mode requires some set-up cycles for the DMAC to read a descriptor and perform other operations, as shown in figure 6-23. At the start of a DMA transfer, 23 states are inserted. During buffer switching, 11 states indicated by[\*3] are inserted to write receive data length and status fields in the descriptor. This is followed by 23 states in which the next descriptor is read.



**Figure 6-23. MSCI-to-Memory Chained-block Transfer Timing  
(for Transfer Start and Buffer Switching)**

## 6.4.7 Characteristics

Table 6-12 lists the characteristics of the DMAC.

**Table 6-12. Characteristics**

Mode	Item	States/Byte for DMA Transfer*1	DMA Transfer Set-up Time*2	DMAC Buffer Switching Time
Single-block transfer mode	Memory to/from I/O	6		
	Memory to memory	6		
Chained-block transfer mode	Memory to MSCI (transmit)	3	32 *3	33/37 *4
	MSCI to memory (receive)	3	23 *5	34 *6

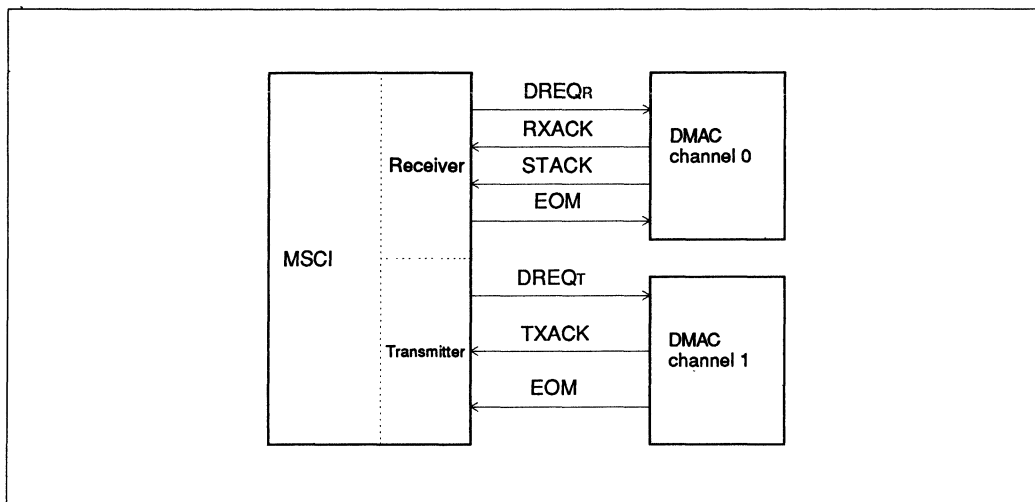
- \*1 The units for a DMA transfer are the number of states/byte (all other units are states).  
The values shown are valid when wait states are not inserted (in the memory or I/O cycle).
- \*2 Before entering the data transfer cycle the DMAC requires some set-up time to read the heading descriptor, etc.
- \*3 32 states = 8 memory cycles (24 states) + 8 internal states
- \*4 33 states = 8 memory cycles (24 states) + 9 internal states (for the middle of a frame)  
37 states = 8 memory cycles (24 states) + 13 internal states (for the end of a frame)
- \*5 23 states = 5 memory cycles (15 states) + 8 internal states
- \*6 34 states = 8 memory cycles (24 states) + 10 internal states

## 6.5 Connections Between the DMAC and MSCI

DMAC channels 0 and 1 are hardwired to the MSCI receiver and transmitter, as shown in figure 6-24.

DMAC channel 0 is connected to the MSCI receiver via four lines: DREQ<sub>R</sub>, RXACK, STACK, and EOM. DMAC channel 1 is connected to the MSCI transmitter via three lines: DREQ<sub>T</sub>, TXACK, and EOM.





**Figure 6-24. Connections Between the DMAC and MSCI**

Table 6-13 lists the function of each line between DMAC channel 0 and the MSCI receiver. Table 6-14 lists the function of each line between DMAC channel 1 and the MSCI transmitter.

**Table 6-13. Lines between DMAC Channel 0 and MSCI Receiver**

Symbol	Name	Transfer Direction	Function
DREQR	DMA receive request	MSCI to DMAC	Used by the MSCI to request a DMA transfer
RXACK	Receive acknowledge	DMAC to MSCI	Used by the DMAC to notify the MSCI to put receive data onto the internal data bus
STACK	Status acknowledge	DMAC to MSCI	Used by the DMAC to notify the MSCI to put the status of the receive data onto the internal data bus
EOM	End of frame transfer	MSCI to DMAC	Used by the MSCI to notify the DMAC that the transferred byte is the end of a frame

**Table 6-14. Lines between DMAC Channel 1 and MSCI Transmitter**

Symbol	Name	Transfer Direction	Function
DREQT	DMA transmit request	MSCI to DMAC	Used by the MSCI to request a DMA transfer
TXACK	Transmit acknowledge	DMAC to MSCI	Used by the DMAC to notify the MSCI to latch the transmit data from the internal data bus
EOM	End of frame transfer	DMAC to MSCI	Used by the DMAC to notify the MSCI that the transferred byte is the end of a frame.

## 6.6 Internal Interrupts

DMAC channels 0 and 1 can issue DMIA (error) and DMIB (normal end) internal interrupts requests. Interrupt request status is indicated by the DMA status register (DSR) and the interrupts are enabled or disabled by the DMA interrupt enable register (DIR). Table 6-15 lists the types, sources and clear procedure of interrupt. Figure 6-25 shows the relationship between the interrupt status bits and enable bits.

**Table 6-15. Internal Interrupts**

Type	Source	Status Bits	Enable Bits	Clear Procedure
Error interrupt (DMIA)*1	When the frame-end interrupt-counter (FCT) is overflow (the number of pending interrupts is 16 or more).	COF	COFE	Write a 1 to the status bit
	When the EDA matches the CDA and a new transfer request is issued (buffer underrun/ overrun).	BOF	BOFE	Write a 1 to the status bit

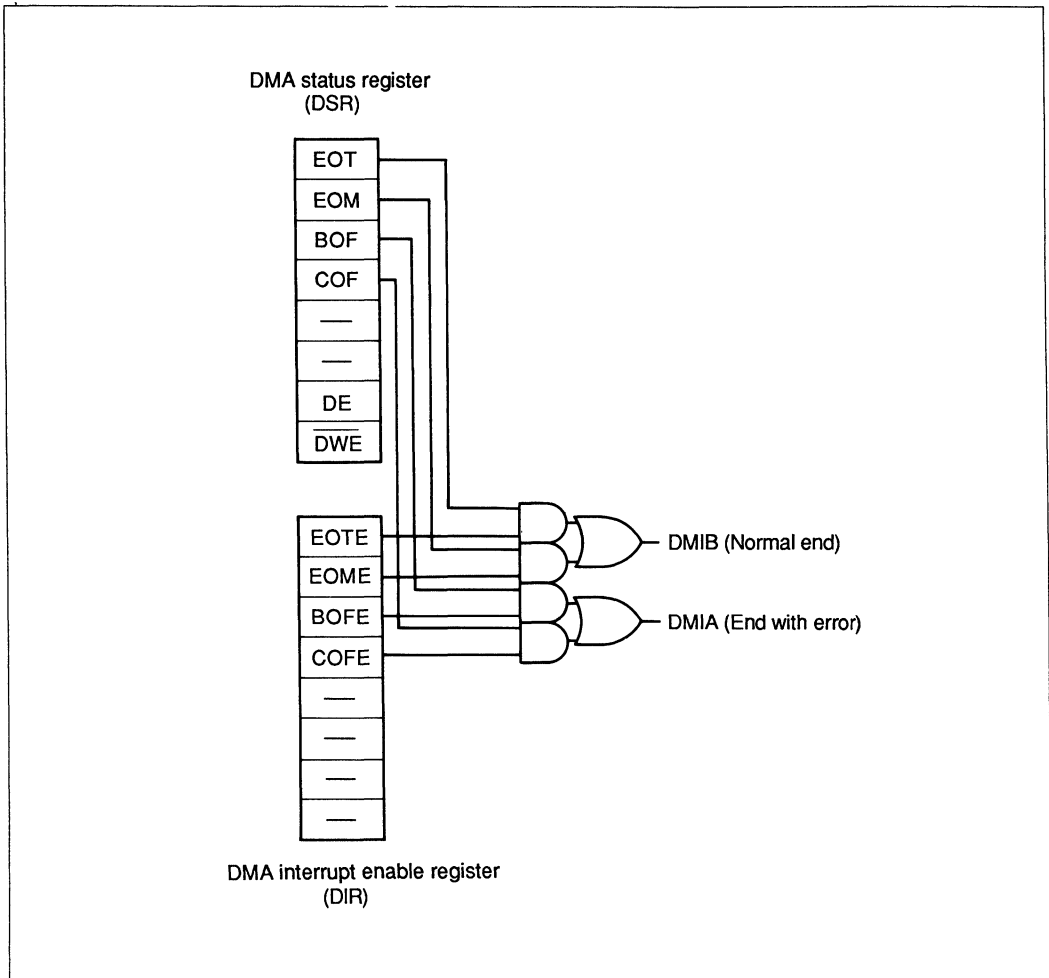
**Table 6-15. Internal Interrupts (cont.)**

Type	Source	Status Bits	Enable Bits	Clear Procedure
Normal end interrupt (DMIB)*1	Frame transfer is completed.*2 (chained-block transfer mode)	EOM	EOME	① Write a 1 to the status bit*3 ② Issue a frame-end interrupt-counter-clear command
	DMA transfer is completed.	EOT	EOTE	Write a 1 to status bit

\*1 Interrupts, once issued, continue to be requested also in the DMA initial state or halt state.

\*2 An interrupt at the end of a 1-frame transfer during chained block multi-frame transfer does not signal the end of a transfer.

\*3 When the frame-end interrupt-counter (FCT) is enable and the FCT value is not 0000, the EOM bit is set to 1. For details, see sections 6.2.7 "DMA Status Register," 6.2.10 "Frame-End Interrupt-Counter" and 6.2.12 "DMA Command Register."



**Figure 6-25. Relationship between Interrupt Status Bits and Enable Bits**

## 6.7 Reset Operation

A reset places the DMAC (both channels 0 and 1) into the following status:

- (1) The DMAC enters the DMA initial state.
- (2) Channel 0 takes priority over channel 1.
- (3) The  $\overline{\text{TEND}}$  line is fixed at the high level.
- (4) Values of the transfer control registers for specifying addresses and of the DMA command register (DCR) are undefined.

(5) The DMA status register (DSR), DMA mode register A (DMRA), DMA mode register B (DMRB), frame-end interrupt-counter (FCT) and DMA interrupt enable register (DIR) are initialized as follows:

- Operation mode: I/O to memory single-block transfer mode (dual address)
- DMA transfer request source: External line (The  $\overline{\text{DREQ}}$  line is level sensitive.)
- Interrupt status bits and enable bits are cleared.
- The FCT value is cleared and FCT is disabled.

## 6.8 Precautions

(1) The DMAC registers must be initialized while in the DMA initial state. When DMAC operation is suspended by writing a 0 to the DE bit in software, the DMAC retains its previous operation status. Thus to initiate new operation, the software abort command must be issued to initialize the status. However, when the DMAC operation is terminated by a transfer completion condition, the software abort command is not necessary. For details, see section 6.2.12 "DMA Command Register."

(2) The DMAC must be disabled when the system stop mode is entered.

(3) When the DE bit is cleared, the transfer request that was received via the edge-sensitive input is cancelled. However, when the DME bit is cleared, the edge-sensitive input request is retained.

## Section 7. Timers

### 7.1 Overview

#### 7.1.1 Functions

The HD64180S incorporates a timer for two functionally-identical channels (channels 0 and 1).

This timer has the following features:

- 8-bit reloadable timer which can count external events
- Operated on Base Clock (BC) ( $\phi$  clock internally divided by 8)  
Count-up intervals in the range  $BC/2^0$ - $BC/2^7$   
An external event count signal can also be used for count-up operation
- Low/high level or toggled timer output selectable  
Toggled output provides an external square wave with a 50% duty cycle
- Internal interrupt can be issued upon count value matching.

## 7.1.2 Configuration and Operation

Figure 7-1 shows a block diagram of a timer.

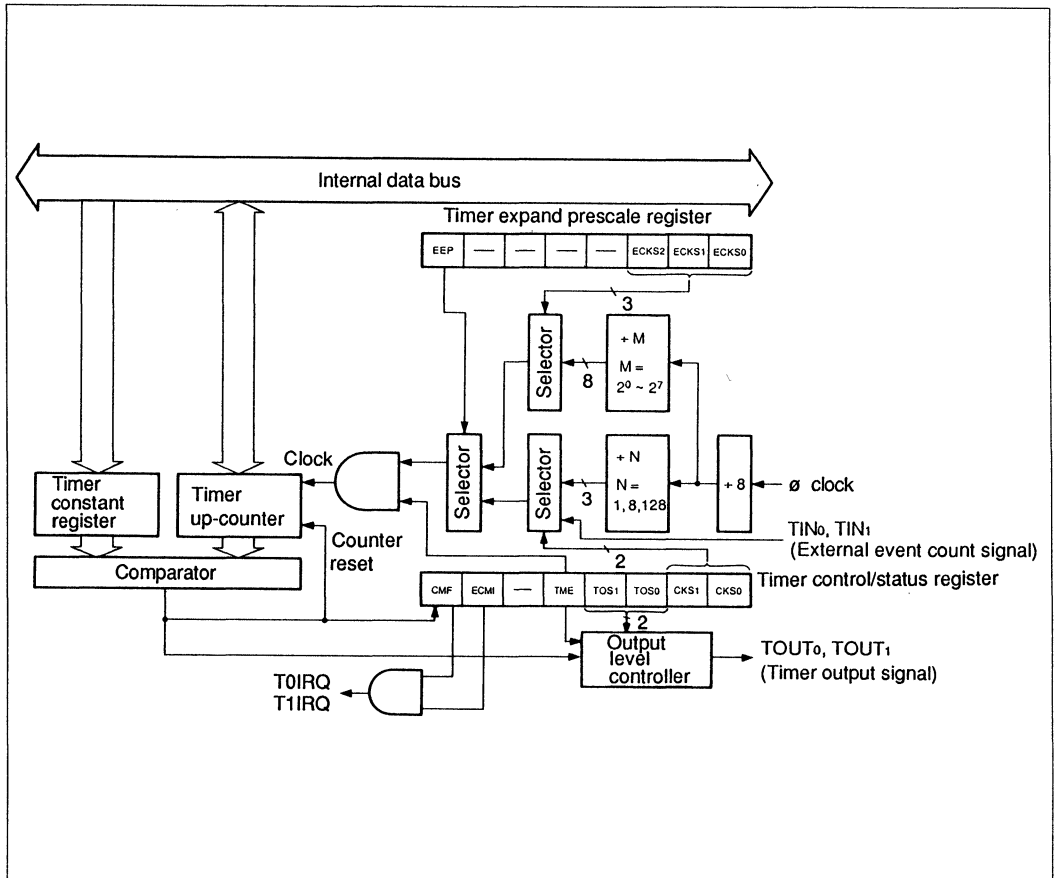


Figure 7-1. Timer Block Diagram

In this timer, the timer up-counter (TCNT) performs a count-up operation based on the specified clock signal. When the TCNT value matches a specified value set in the timer constant register (TCONR), the TOUT line\*1 provides a timer output signal which is either low or high level or in toggled form. In this case, an internal interrupt, if enabled, can be generated.\*2 Also at this time, the TCNT value is cleared to 00H and count-up operation restarts.\*3

\*1 For details on output timing, see 7.3.2 "Output Timing."

\*2 For details on timing, see 7.4 "Internal Interrupt."

\*3 For details on output timing, see 7.3.1 "Timer Count-up Timing."

### 7.1.3 Registers

Table 7-1 lists timer registers.

**Table 7-1. Timer Registers**

Register Name	Symbol	I/O Address		Initial Value	Read/Write
		Channel 0	Channel 1	MSB↔LSB	
Timer up-counter	TCNT	0050H	0054H	00000000	R/W
Timer constant register	TCONR	0051H	0055H	11111111*	W
Timer control/status register	TCSR	0052H	0056H	00000000	R/W
Timer expand prescale register	TEPR	0053H	0057H	00000000	R/W

\* The timer constant register is a write-only register. It always reads 00H.

## 7.2 Registers

### 7.2.1 Timer Up-counter (TCNT)

The TCNT registers for channels 0 and 1 are functionally identical. TCNT counts up in the clock specified by the CKS1–0 bits in TCSR or the ECKS2–0 bits in TEPR. For information regarding clock selection, see section 7.2.3 "Timer Control/Status Register" and section 7.2.4 "Timer Expand Prescale Register."

Software read and write operations do not effect the operation of the counter.

The TCNT is cleared to 00H when its value matches the value in the timer constant register (TCONR).

	7	6	5	4	3	2	1	0
	□	□	□	□	□	□	□	□
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value*	0	0	0	0	0	0	0	0

\* This value is the same as in the system stop mode.



## 7.2.2 Timer Constant Register (TCONR)

The timer constant registers for channels 0 and 1 are functionally identical and can be used to generate output waveforms. The contents of this register are compared with the TCNT contents constantly. When they match, the CMF bit in TCSR is set; this causes the TOUT line to be driven as specified by the TOS1-0 bits in TCSR. (For details, see the explanation about the TOS1-0 bits in section 7.2.3 "Timer Control/Status Register".) The TCNT is cleared and resumes counting up from 00H. (For details, see section 7.3.2 "Output Timing".) In this way, periodic interrupts and output waveforms can be generated without increasing software overhead. The TCONR is set to FFH by a reset or when the CPU enters the system stop mode.

	7	6	5	4	3	2	1	0
Read/Write*	W	W	W	W	W	W	W	W
Initial Value	1	1	1	1	1	1	1	1

\* TCONR is a write-only register. It always reads 00H.

## 7.2.3 Timer Control/Status Register (TCSR)

The function of TCSR is the same for channels 0 and 1. This register is used to request interrupts, control the TCNT, control output value for timer output signals, and select the input clock.

	7	6	5	4	3	2	1	0
Bit Name	CMF	ECMI	- <sup>1</sup>	TME	TOS1	TOS0	CKS1	CKS0
Read/Write	R	R/W	-	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

<p><u>Compare Match Flag</u></p> <p>0: TCNT and TCONR are not equal 1: TCNT and TCONR are equal</p>	<p><u>CMF Interrupt Enable</u></p> <p>0: Disable 1: Enable</p>	<p><u>Timer Enable</u></p> <p>0: Count stop 1: Count start</p>	<p><u>Timer Output Select</u></p> <p>00: Output fixed to 0 01: Toggled output<sup>*4</sup> 10: Output 0 11: Output 1</p>	<p><u>Input Clock Select</u></p> <p>00: BC<sup>*2</sup> 01: BC/8 10: BC/128 11: External event count signal<sup>*3</sup></p>
---	--	--	--	--

<sup>\*1</sup> Reserved. This bit always reads 0 and should be set to 0.

<sup>\*2</sup> BC (Base Clock) is generated internally by dividing the  $\phi$  clock by 8.

<sup>\*3</sup> The external event count is incremented by 1 at the rising edge of the clock.

<sup>\*4</sup> The timer output value is toggled each time the TCNT and TCONR values match. This facilitates the generation and output of square waves with a 50% duty cycle without increasing software overheads.

### Bit 7: CMF (Compare Match Flag)

CMF is set to 1 when the TCNT value matches the content of TCONR. It is cleared by reading TCSR followed by TCNT. Other instructions can be inserted between the TCSR and TCNT read instructions.

The CMF bit is set to 0 after a reset or when the CPU enters the system stop mode.

CMF	Function
0	TCNT and TCONR do not match
1	TCNT and TCONR match. If the ECMI bit (bit 6) has been set, an internal interrupt request (TOIRQ or T1IRQ) is generated

### Bit 6: ECMI (CMF Interrupt Enable)

ECMI specifies whether to enable or disable an interrupt caused by the CMF bit. This bit is cleared after a reset.

ECMI	Function
0	Disables an interrupt caused by the CMF bit
1	Enables an interrupt caused by the CMF bit

**Bit 5:** Reserved. Bit 5 always reads 0 and should be set to 0.

### Bit 4: TME (Timer Enable)

TME specifies whether to start or stop TCNT operation. This bit is set to 0 after a reset or when the CPU enters the system stop mode.

TME	Function
0	Stops TCNT. The TOUT line is set low, but the current TCNT value is retained.*
1	Starts TCNT.

\*TME is again set to 1, and TCNT resumes counting from the retained value.

### Bits 3-2: TOS1-0 (Timer Output Select)

The TOS1–0 bits control the output value to the timer out signal when TCNT and TCONR match. These bits are both set to 0 after a reset.

TOS1	TOS0	TOUT Output
0	0	Fixed to low
0	1	Toggled*
1	0	Low
1	1	High

\* The timer output value is toggled each time the TCNT and TCONR values match. This facilitates the generation and output of square waves with a 50% duty cycle without increasing software overheads.

#### Bits 1-0: CKS1-0 (Input Clock Select)

The CKS bits select the TCNT operating clock source. These bits are set to 0 after a reset.

CKS1	CKS0	TCNT Operating Clock Rate
0	0	BC* <sup>1</sup>
0	1	BC/8
1	0	BC/128
1	1	The external event count signal * <sup>2</sup>

\*<sup>1</sup> The BC (Base Clock) is generated by internally dividing the  $\phi$  clock by 8.

\*<sup>2</sup> An external event signal is counted at the rising edge of the clock. In order for an external event count signal to be counted accurately, this signal must be at least two or more  $\phi$  clock cycles wide for both high and low levels, and must be less than 1/4 of the  $\phi$  clock.

## 7.2.4 Timer Expand Prescale Register (TEPR)

The function of TEPR is the same for channels 0 and 1. The TEPR register selects the clock for the TCNT and the input for the expanded clock.

	7	6	5	4	3	2	1	0
Bit Name	EEP	—*	—*	—*	—*	ECKS2	ECKS1	ECKS0
Read/Write	RW	—	—	—	—	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0

<p><u>Enable Expand Prescaler</u></p> <p>0: Clock is selected by the CKS1-0 bits in TCSR            1: Clock is selected by the ECKS2-0 bits in TEPR</p>	<p><u>Expand Clock Input Select</u></p> <p>000: BC            001: BC/2            010: BC/4            011: BC/8            100: BC/16            101: BC/32            110: BC/64            111: BC/128</p>
--	--

\* Reserved. These bits always read 0 and should be set to 0.

### Bit 7: EEP (Enable Expand Prescaler)

EEP selects either the CKS1-0 bits in TCSR or the ECKS2-0 bits in TEPR to specify the TCNT operating clock. It is set to 0 after a reset.

EEP	Function
0	The CKS1-0 bits in TCSR specify the TCNT operating clock
1	The ECKS2-0 bits in TEPR specify the TCNT operating clock

**Bits 6-3:** Reserved. Bits 6-3 always read 0 and should be set to 0.

### Bits 2-0: ECKS2-0 (Expand Clock Input Select)

When the EEP bit in TEPR is 1, the ECKS bits select the TCNT clock as shown in the following table.

These bits are set to 0 after a reset.

ECKS2	ECKS1	ECKS0	TCNT Clock Rate
0	0	0	BC
0	0	1	BC/2
0	1	0	BC/4
0	1	1	BC/8
1	0	0	BC/16
1	0	1	BC/32
1	1	0	BC/64
1	1	1	BC/128

## 7.3 Operation Timing

### 7.3.1 Timer Count-up Timing

(1) Figure 7-2 shows the timing when the counter operating rate is BC. Counting-up is initiated by writing 1 to the TME bit of the TCSR after TCNT and TCONR have been set.

When the TCNT and TCONR values match, the CMF bit is set to 1 and internal interrupts (TOIRQ and T1IRQ), if enabled, are generated. (The CMF bit can be cleared by reading TCSR followed by TCNT.) At this time, TCNT is initialized to 00H, and then count-up operation restarts. TCNT can be written to during count-up. In this case, count-up is performed from the written value.

When the TME bit is cleared during count-up, TCNT stops counting and retains its current contents. When the TME bit is again set to 1, count-up resumes from the retained value.

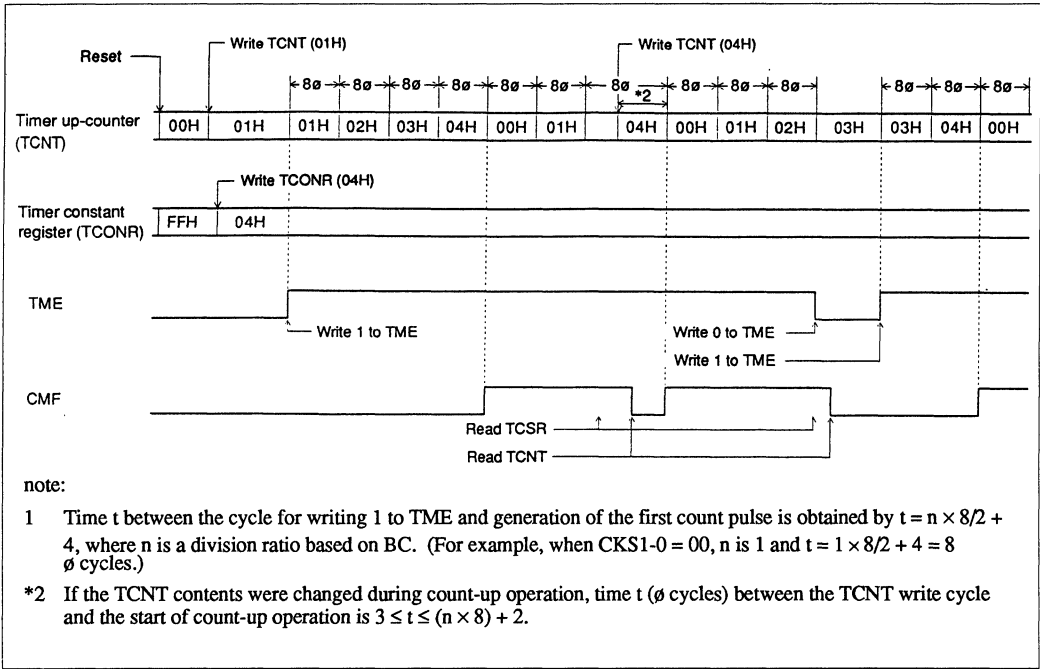


Figure 7-2. Timer Count-up Timing (Example 1)

(2) Figure 7-3 shows the timing when the counter operating rate is BC/4.

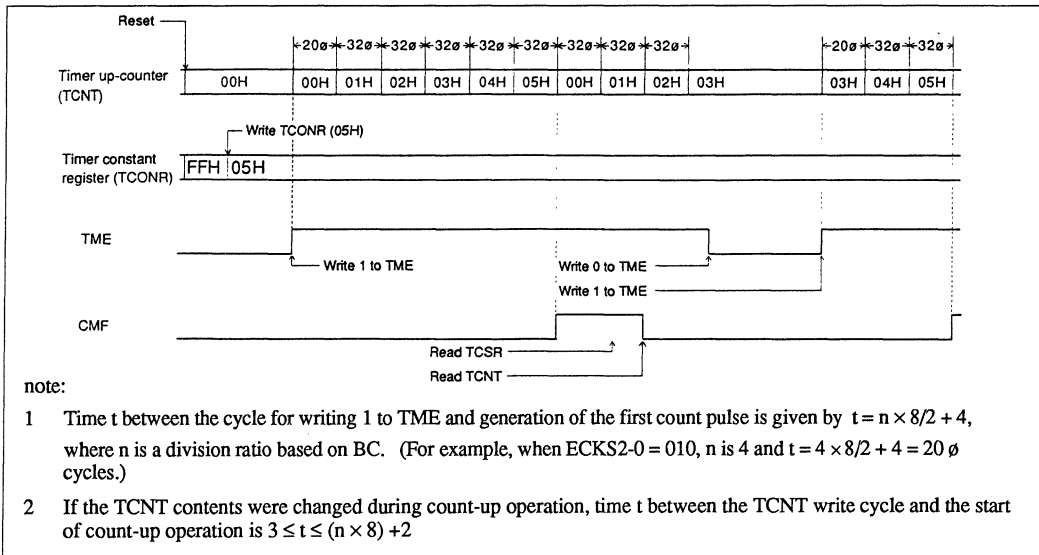


Figure 7-3. Timer Count-up Timing (Example 2)

(3) Figure 7-4 shows the timing when the counter operating rate is determined by an external event count signal.

Counting is performed at the rising edge of an external event count signal. The event count signal must be two or more  $\phi$  clock cycles wide. Before count-up operation is started, 1 must be written to the TME bit when the external event count signal level is low.

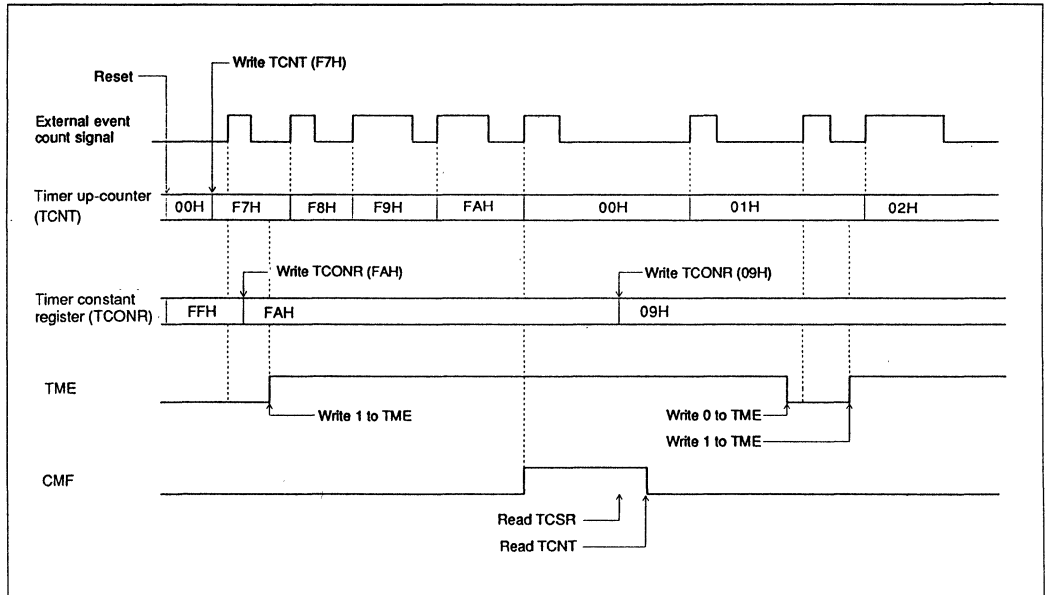


Figure 7-4. Timer Count-up Timing (Example 3)

### 7.3.2 Output Timing

Figure 7-5 shows the timing when the output of the timer is changing. When TCNT and TCONR match and TCNT is subsequently initialized to 00H, 1  $\phi$  clock cycle thereafter, the CMF bit is set to 1 and the TOUT line provides a timer output signal.

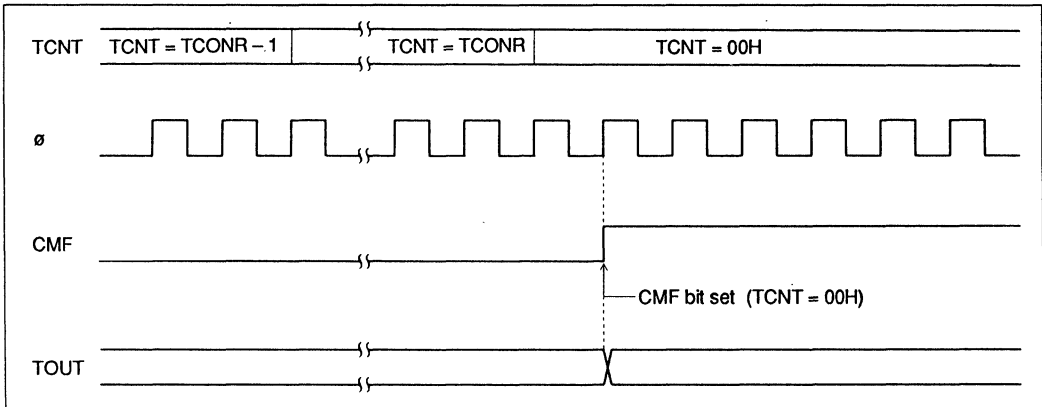


Figure 7-5. Timer Output Timing

### 7.4 Internal Interrupt

When the TCNT and TCONR match, the CMF bit of the TCSR is set to 1. If the interrupt enable bit is set, an internal interrupt is generated\*.

Figure 7-6 shows the internal interrupt circuit. Figure 7-7 shows internal interrupt timing.

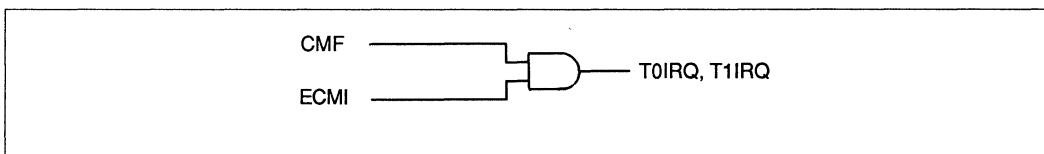
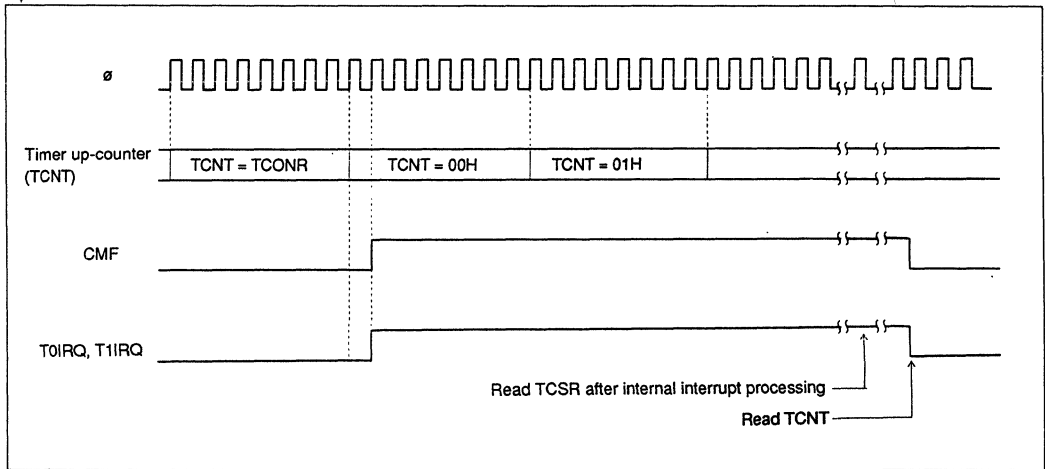


Figure 7-6. Internal Interrupt Circuit

\* Interrupts originating in the CMF bit are enabled/disabled by the ECMI bit in TCSR.





**Figure 7-7. Internal Interrupt Timing (when the counter operating rate is BC and internally synchronized)**

## 7.5 Low Power Dissipation Mode

In the sleep mode, timers continue to operate normally. In the system stop mode, timers are initialized as follows:

- The control bits of the TCSR and TEPR hold their current contents, except that the TME bit in TCSR is cleared to 0.
- The CMF bit in TCSR is cleared.
- TCNT stops; it is then initialized to 00H.
- The TOUT signal retains its previous value. After leaving the system stop mode, this signal is initialized to the low level.
- Any interrupt requests in TOIRQ and T1IRQ are cleared.

## 7.6 Reset Operation

The timers are initialized by a reset as follows:

- TCSR and TEPR are initialized to 00H.
- TCNT stops and is initialized to 00H.
- TCONR is initialized to FFH.
- The TOUT line is set low, and any interrupt requests in TOIRQ and T1IRQ are cleared.

## 7.7 Precautions

When using the timers, observe the following precautions:

- Be sure to clear the TME bit before changing the timer operating clock.
- When using the external event count function, drive the external event count signal low, then set the TME bit to 1.
- Reserved bits in the TCSR and TEPR read 0.
- The TOUT line also goes low after the TME bit in TCSR is cleared.

## Section 8. Refresh Controller

### 8.1 Overview

#### 8.1.1 Functions

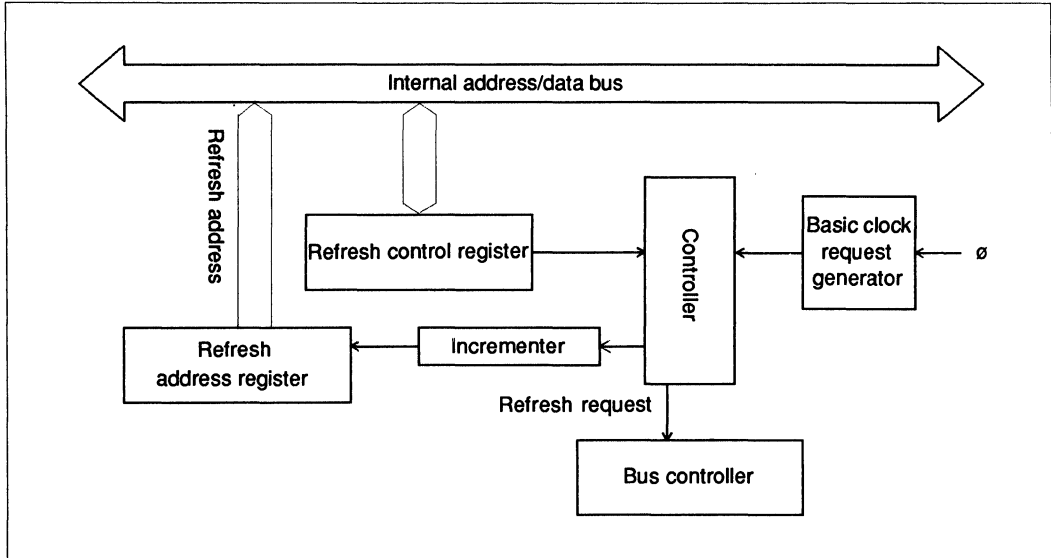
The HD64180S contains a programmable DRAM refresh controller. This refresh controller has the following functions:

- Refresh enable/disable
- Interval between refresh cycles selectable in eight steps in the range of 32 to 256 states
- Insertion of 0 to 7 wait states in a refresh cycle, when used in combination with the wait controller (See section 9.2.5 "Refresh Wait Control Register" and section 9.3.2 "Wait State Insertion Using Register Control.")

The refresh controller can be programmed to determine an optimum refresh cycle given the DRAM specifications, CPU operating frequency, and the application system, thus eliminating the need for an external refresh circuit.

## 8.1.2 Configuration and Operation

Figure 8-1 shows the block diagram of the refresh controller.



**Figure 8-1. Refresh Controller Block Diagram**

DRAM refresh is performed at the interval specified by the refresh control register (RCR). The basic refresh cycle (which consists of two states with no wait states) is executed between each machine cycle.

### 8.1.3 Register

The refresh controller register is shown in table 8-1.

**Table 8-1 Refresh Controller Register**

Register Name	Symbol	I/O	Initial Value*	Read/Write
		Address	MSB↔LSB	
Refresh controller Register	RCR	0018H	10000000	R/W

\*This is the initial value after a hardware reset.

## 8.2 Register

### 8.2.1 Refresh Control Register (RCR)

This register specifies whether or not refresh cycles are to be inserted, and the length of the interval between refresh cycles.

	7	6	5	4	3	2	1	0
Bit Name	REFE	_*2	_*2	_*2	_*2	CYC2	CYC1	CYC0
Read/Write	R/W	-	-	-	-	R/W	R/W	R/W
Initial Value <sup>*1</sup>	1	0	0	0	0	0	0	0

<u>Refresh Enable</u>	<u>Cycle Select</u>
0: Refresh cycles not inserted 1: Refresh cycles inserted	• Insertion interval 000: 32 states 001: 64 states 010: 96 states 011: 128 states 100: 160 states 101: 192 states 110: 224 states 111: 256 states

\*1 After a reset, this register is initialized to 80H and the refresh address is initialized to 00000H.

\*2 Reserved. These bits always read 0 and should be set to 0.

#### Bit 7: REFE (Refresh Enable)

REFE specifies whether or not DRAM refresh cycles are to be inserted.

REFE	Function
0	Refresh cycles not inserted.
1	Refresh cycles inserted. Interval specified by bits 2-0.

**Bits 6-3:** Reserved. These bits always read 0 and should be set to 0.

## Bits 2-0: CYC2-0 (Cycle Select)

The CYC bits are used to specify the interval between refresh cycles. Table 8-2 lists bit set up values and insertion intervals.

**Table 8-2. Refresh Cycles and their Intervals**

CYC2	CYC1	CYC0	Insertion Interval	Time Interval (examples at typical CPU clock frequencies)			
				ø: 4 MHz	ø: 6 MHz	ø: 8 MHz	ø: 10 MHz
0*	0*	0*	32 states	<u>8.0 μs</u>	5.3 μs	4.0 μs	3.2 μs
0	0	1	64 states	16.0 μs	<u>10.66 μs</u>	8.0 μs	6.4 μs
0	1	0	96 states	24.0 μs	16.0 μs	<u>12.0 μs</u>	9.6 μs
0	1	1	128 states	32.0 μs	21.3 μs	16.0 μs	<u>12.8 μs</u>
1	0	0	160 states	40.0 μs	26.66 μs	20.0 μs	16.0 μs
1	0	1	192 states	48.0 μs	32.0 μs	24.0 μs	19.2 μs
1	1	0	224 states	56.0 μs	37.3 μs	28.0 μs	22.4 μs
1	1	1	256 states	64.0 μs	42.6 μs	32.0 μs	25.6 μs

\* Initial value

For DRAMs requiring 128 refresh cycles every 2 μs (or 256 refresh cycles every 4 μs), the required refresh interval is 15.625 μs. The underlined values in table 8-2 represent the optimum refresh interval for various CPU clock frequencies. However, the actual refresh interval may differ from the interval specified by CYC2-0 because the refresh cycle is executed between machine cycles. If wait states are inserted, the actual refresh interval length will be variant with the values stated here.

## 8.3 Operation

The refresh controller periodically generates refresh request signals to the CPU. When the CPU detects this signal, it enters the refresh cycle at the end of the current machine cycle.

Figure 8-2 shows an example of refresh cycle timing. During the refresh cycle, the  $\overline{\text{REF}}$  signal goes low and a 12-bit refresh address is output on address lines A0 - A11. (The A12 - A19 lines are held low.)

During a DMA operation, refresh occurs at the end of the current bus cycle.  
No refresh cycles occur in the bus release mode or wait mode.

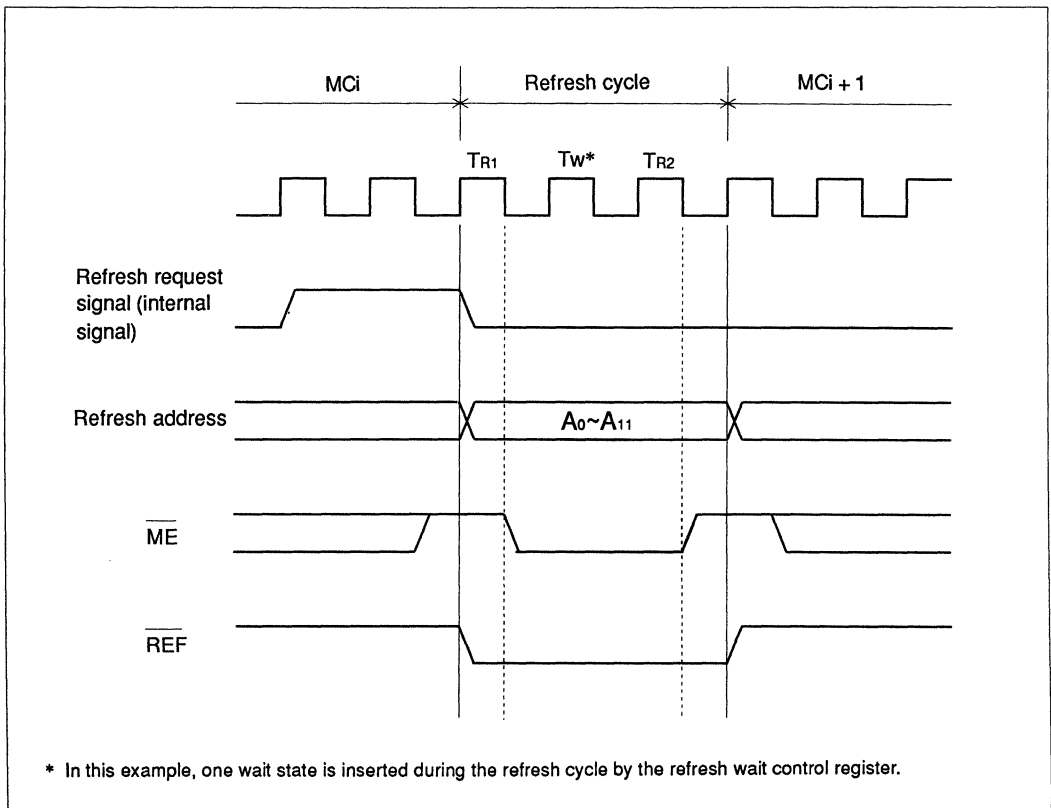


Figure 8-2. Refresh Timing

## 8.4 Refresh Controller Operation in Low Power Dissipation Mode

The refresh controller continues to operate the sleep mode, but stops in the system stop mode. In this mode, the refresh control register and refresh address hold their previous values.

## 8.5 Reset Operation

The refresh controller stops during a reset. The refresh control register is initialized to 80H (32-state refresh interval and refresh enable) and the refresh address to 00000H. Thus, after a reset, refresh is restarted from address 00000H by 32-state interval.

## 8.6 Precautions

When using the refresh functions, observe the following precautions.

- When the CPU is placed in the bus release mode, system stop mode or during a wait state, refresh controller operation is inhibited. It is thus necessary to consider alternate DRAM refresh methods.
- If several refresh requests are generated internally while the CPU is in the bus release mode, only one will be executed after exiting this mode. Figure 8-3 shows the bus cycle timing in this mode.
- If a refresh request is generated internally during a wait state, the request is retained until a new request is generated. After exiting the wait state, a refresh cycle occurs at the end of the current machine cycle.
- After leaving the bus release mode or a wait state, the next refresh cycle begins at the address where the last refresh cycle left off.

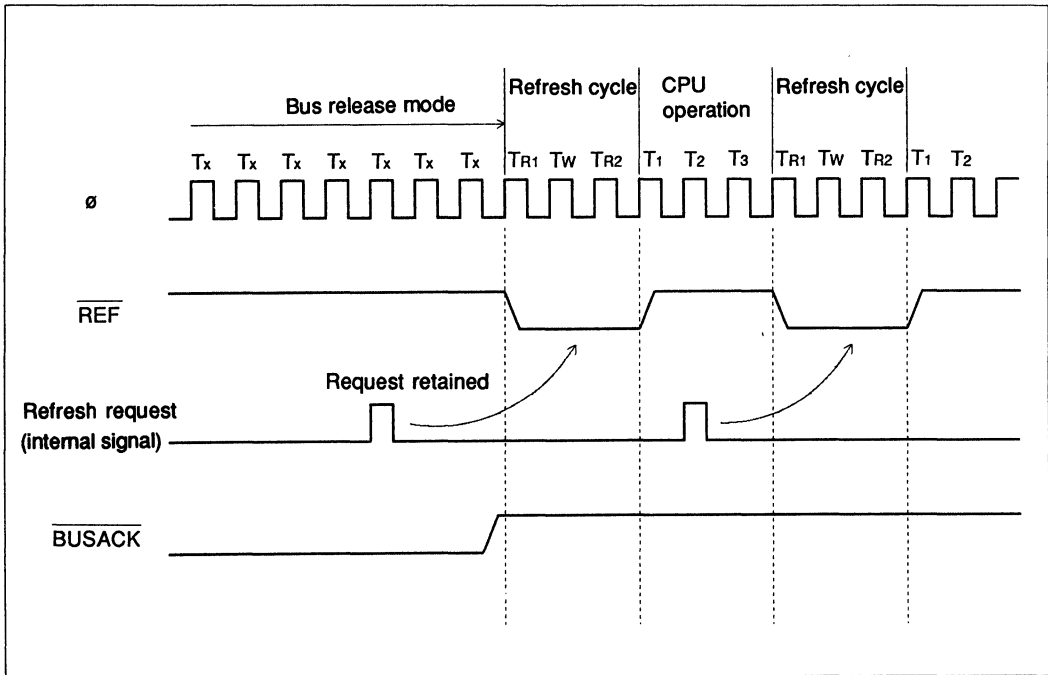


Figure 8-3. Refresh Requests in Bus Release Mode



## Section 9. Wait Controller

### 9.1 Overview

#### 9.1.1 Functions

The HD64180S incorporates a wait controller. It extends bus cycles by inserting wait states. This allows low-speed memory and I/O devices to be interfaced.

The wait controller incorporates the following wait state insertion functions:

- Either  $\overline{\text{WAIT}}$  line (hardware) or register (software) control can be used for wait state insertion.
- Registers for inserting 0 to 7 wait states can be independently specified for each of three different memory areas when each area is accessed.
- Register-controlled insertion of 0 to 7 wait states in I/O cycles when external I/O space is accessed.
- Register-controlled insertion of 2 to 9 wait states in  $\overline{\text{INT0}}$  interrupt acknowledge cycles.
- Register-controlled insertion of 0 to 7 wait states in refresh cycles by internal refresh controller.

## 9.1.2 Configuration and Operation

Figure 9-1 shows a block diagram of the wait controller. The wait controller consists of a wait control unit, an I/O wait control register, wait control registers (L, M, and H), an interrupt wait control register, a refresh wait control register, and physical address boundary registers 0 and 1.

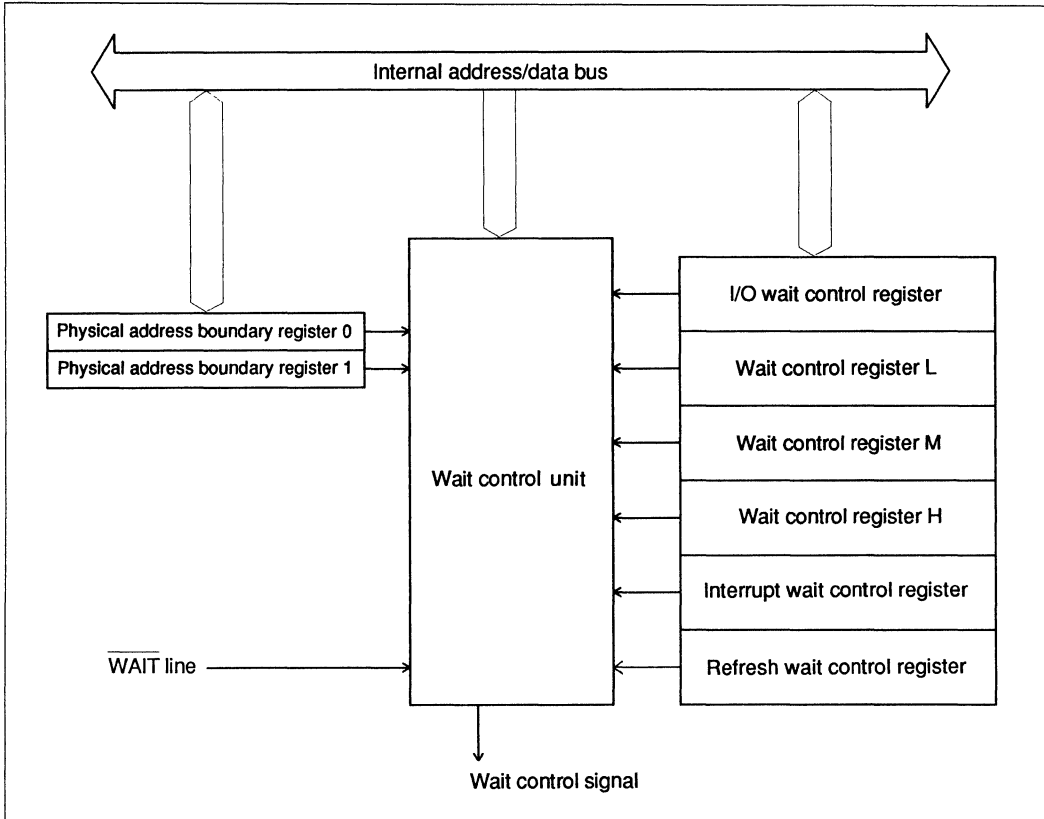


Figure 9-1. Wait Controller Block Diagram

Wait state insertion using  $\overline{\text{WAIT}}$  line control is implemented by driving the  $\overline{\text{WAIT}}$  line low. Wait state insertion using register control is accomplished by specifying the number of wait states to be inserted in the wait control register (L, M and H), I/O wait control register, interrupt wait control register or refresh wait control register.

Wait states are inserted between the T2 and T3 states of each type of bus cycle. Also by setting the boundary addresses in physical address boundary registers 0 and 1, the memory space can be partitioned into three memory areas. The number of wait cycles inserted when each of these areas is accessed can then be specified independently for each area.

### 9.1.3 Registers

The eight registers which comprise the wait controller are listed in table 9-1. For details about these registers, see section 9.2 "Registers."

**Table 9-1. Registers**

Register Name	Symbol	I/O	Initial Value*	Read/Write
		Address	MSB↔LSB	
Physical address boundary register 0	PABR0	0008H	00000000	R/W
Physical address boundary register 1	PABR1	0009H	00000000	R/W
Wait control register L	WCRL	000AH	00000111	R/W
Wait control register M	WCRM	000BH	00000111	R/W
Wait control register H	WCRH	000CH	00000111	R/W
I/O wait control register	IOWCR	000DH	01110111	R/W
Interrupt wait control register	INTWR	000EH	00000111	R/W
Refresh wait control register	RWCR	000FH	00000111	R/W

\* These are the initial values after a hardware reset.

## 9.2 Registers

### 9.2.1 Physical Address Boundary Registers 0 and 1 (PABR0 and PABR1)

The PABR registers specify the boundaries which divide the memory space into three areas.

**Physical Address Boundary Register 0 (PABR0):** The PABR0 register specifies the high-order 8 bits of the boundary between the physical address low (PAL) and the physical address middle (PAM) areas.

	7	6	5	4	3	2	1	0
Bit Name	PB07	PB06	PB05	PB04	PB03	PB02	PB01	PB00
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

|  
PAL/PAM Boundary Address (8 high-order bits)

This register can specify only the 8 high-order bits (A19 - A12) of the boundary address; the remaining 12 low-order bits (A11 - A0) are fixed to 000H. (i.e. each area boundary address corresponds to a 4 kbyte boundary).

When the PABR0 register is set to 00H, the boundary is specified as the physical address space upper limit address.

**Physical Address Boundary Register 1 (PABR1):** The PABR1 register specifies the high-order 8 bits of the boundary between the physical address middle (PAM) and physical address high (PAH) areas.

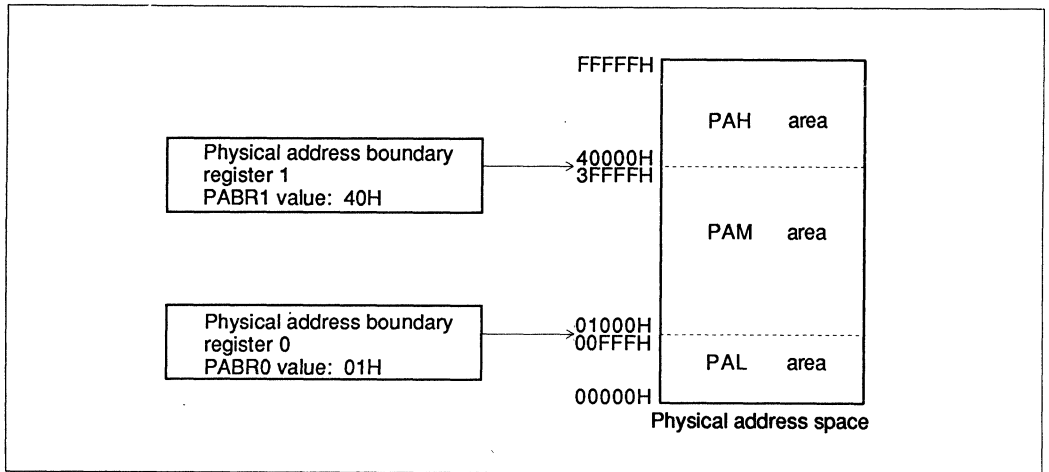
	7	6	5	4	3	2	1	0
Bit Name	PB17	PB16	PB15	PB14	PB13	PB12	PB11	PB10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

|  
PAM/PAH Boundary Address (8 high-order bits)

This register can specify only the 8 high-order bits (A19 - A12) of the boundary address; the remaining 12 low-order bits (A11 - A0) are fixed to 000H. (i.e. each area boundary address corresponds to a 4 kbyte boundary).

When the PABR1 register is set to 00H, the border is specified as the upper limit address of the physical memory space FFFFFH.

Set-up example: The physical address space shown in figure 9-2 is divided into three areas: PAL area, PAM area, and PAH area. PABR0 and PABR1 specify the boundaries of these areas and can be set in 4 kbyte units.



**Figure 9-2 Memory Space Partitioned by PABR0 and PABR1**

**Boundary address setting examples:** When the PABR0 and PABR1 registers are set to 01H and 40H, the boundaries for each memory area are specified as follows:

	Upper Limit Address	Lower Limit Address
PAH	FFFFFFH	40000H
PAM	3FFFFH	01000H
PAL	00FFFH	00000H

When either the PABR0 or PABR1 register is set to 00H, the border is set at the upper limit address of the memory area above it. In this example, PABR1 is set to 00H and each area border is specified as follows:

	Upper Limit Address	Lower Limit Address
PAH	—	—*
PAM	FFFFFFH	01000H
PAL	00FFFH	00000H

\*The physical address space consists of PAL and PAM areas only because the PAM upper limit address is FFFFFFFH.

Figures 9-3 (a) to (d) show examples when the physical address space is not partitioned, when it is partitioned into PAL and PAM, into PAL and PAH, and when it is partitioned into three areas (PAL, PAM and PAH).

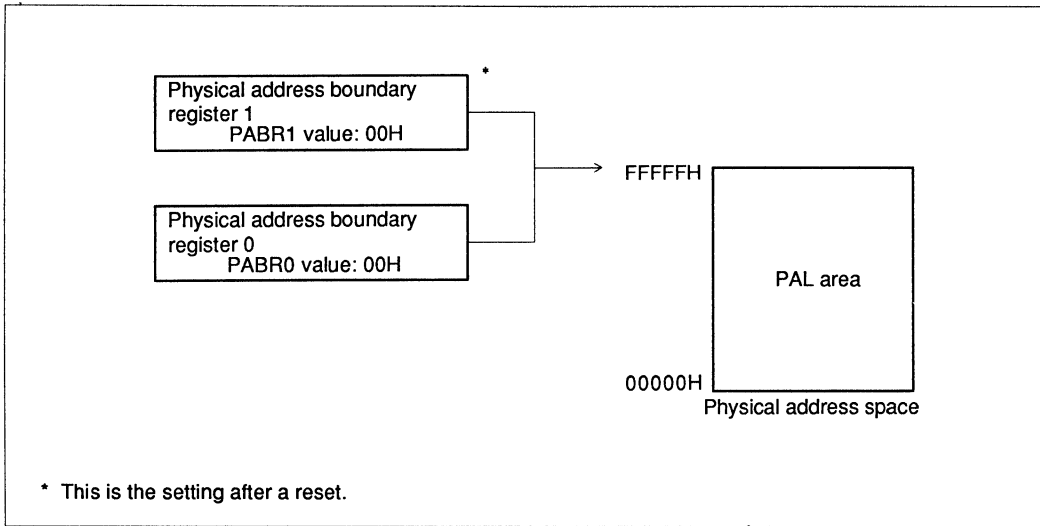


Figure 9-3. (a) Setting Example when the Physical Address Space is not Partitioned

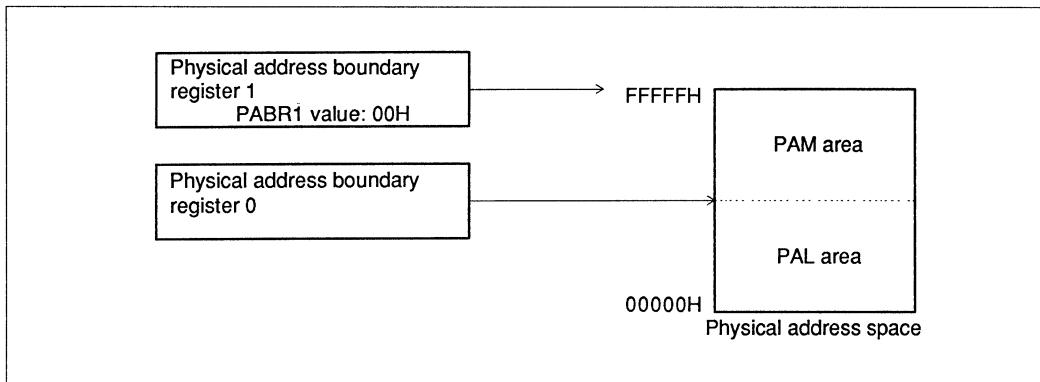
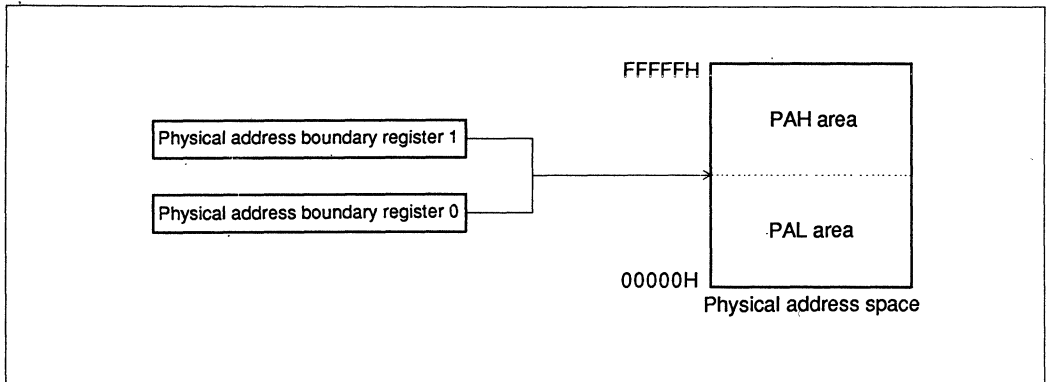
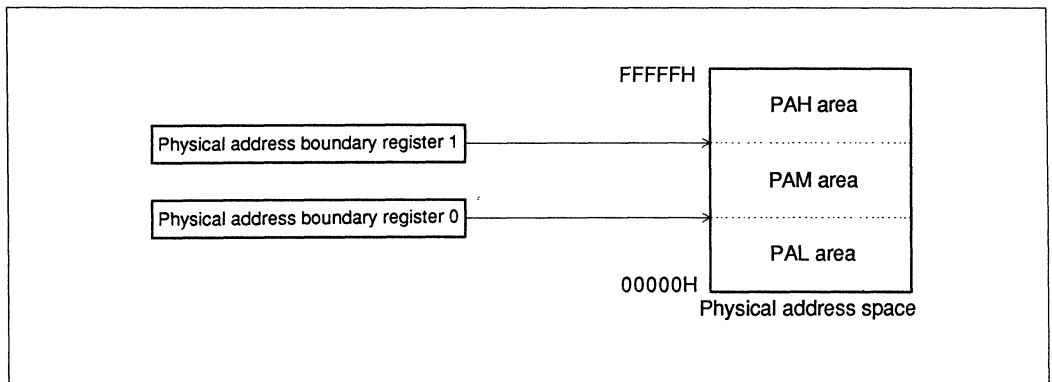


Figure 9-3. (b) Setting Example when the Physical Address Space is Partitioned into PAM and PAL

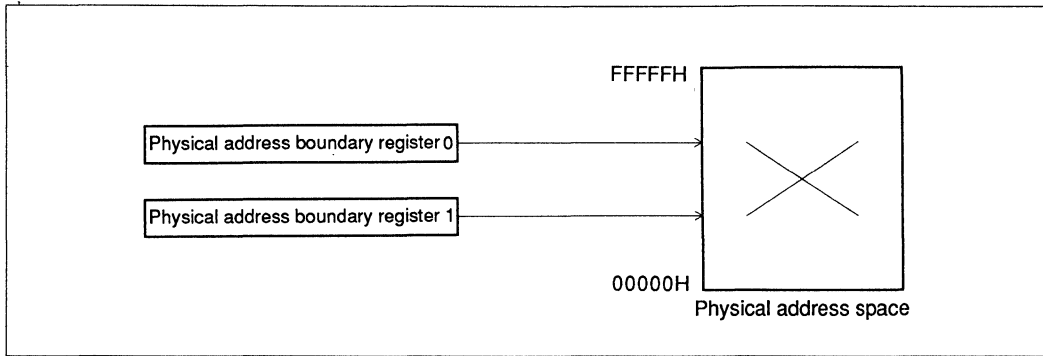


**Figure 9-3. (c) Setting Example when the Physical Address Space is Partitioned into PAH and PAL**



**Figure 9-3. (d) Setting Example when the Physical Address Space is Partitioned into PAH, PAM and PAL**

**Cautions on Use:** Normal operation is not guaranteed if the boundary specified by PABR0 is higher than that specified by PABR1. An example of this type of incorrect setting is shown in figure 9-4.



**Figure 9-4. Example of Incorrect Boundary Specification**

### 9.2.2 Wait Control Registers L, M, and H (WCRL, WCRM, and WCRH)

The WCR registers specify the number of wait states to be inserted for each of the memory areas (PAH, PAM, and PAL).

**Wait Control Register L (WCRL):** WCRL specifies the number of wait states to be inserted in the memory cycle when the PAL area is accessed.

	7	6	5	4	3	2	1	0
Bit Name	—*	—*	—*	—*	—*	PALW2	PALW1	PALW0
Read/Write	—	—	—	—	—	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	1

PAL Area Wait

\* Reserved. These bits always read 0 and should be set to 0.

**Bits 7-3:** Reserved bits. These bits always read 0 and should be set to 0.

#### **Bits 2-0: PALW 2-0 (PAL area wait)**

The table below lists bit values and number of wait states.



PALW2	PALW1	PALW0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1*	1*	1*	7*

\* These are the initial values after a reset.

**Wait Control Register M (WCRM):** WCRM specifies the number of wait states to be inserted in the memory cycle when the PAM area is accessed.

	7	6	5	4	3	2	1	0
Bit Name	—*	—*	—*	—*	—*	PAMW2	PAMW1	PAMW0
Read/Write	—	—	—	—	—	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	1

PAM Area Wait

\* Reserved. These bits always read 0 and should be set to 0.

**Bits 7-3:** Reserved bits. These bits always read 0 and should be set to 0.

**Bits 2-0:** PAMW2-0 (PAM area wait)

The table below lists bit values and number of wait states.

PAMW2	PAMW1	PAMW0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1*	1*	1*	7*

\* These are the initial values after a reset.

**Wait Control Register H (WCRH):** WCRH register specifies the number of wait states to be inserted in the memory cycle when the PAH area is accessed.

	7	6	5	4	3	2	1	0
Bit Name	-*	-*	-*	-*	-*	PAHW2	PAHW1	PAHW0
Read/Write	-	-	-	-	-	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	1

|
|
|  
 PAH Area Wait

\* Reserved. These bits always read 0 and should be set to 0.

**Bits 7-3:** Reserved bits. These bits always read 0 and should be set to 0.

**Bits 2-0:** PAHW2-0:(PAH area wait)

The table below lists bit values and number of wait states.

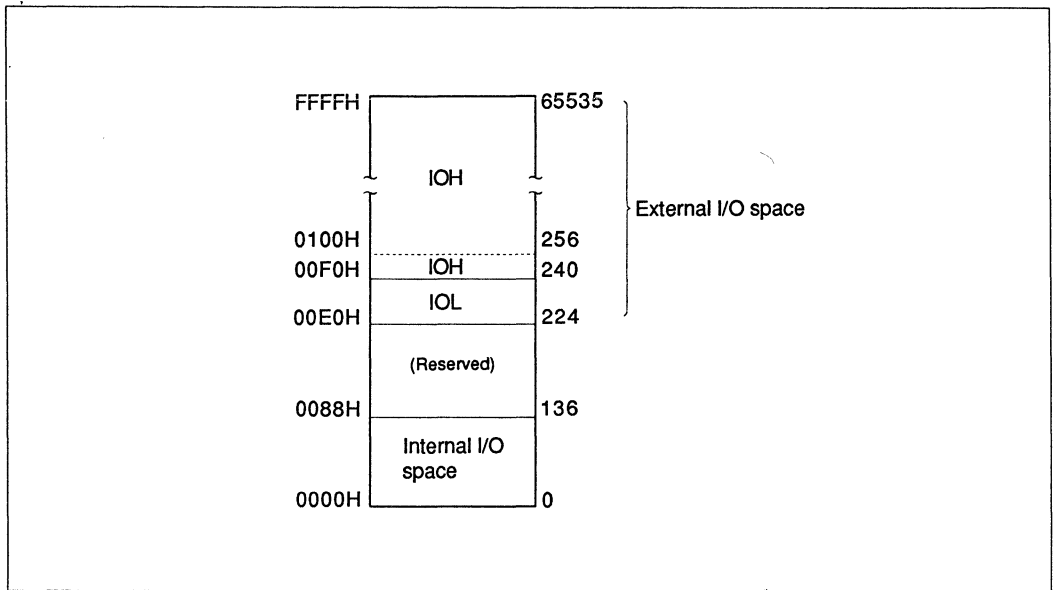
PAHW2	PAHW1	PAHW0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1*	1*	1*	7*

\* These are the initial values after a reset.

### 9.2.3 I/O Wait Control Register (IOWCR)

IOWCR specifies the number of wait states to be inserted in the I/O cycle when an external I/O space is accessed.

The I/O space is divided into internal and external I/O areas. The external I/O area is further divided at address 00F0H into IOH and IOL (figure 9-5). The IOWCR specifies the number of wait states for IOH and IOL in the external I/O area. When accessing the internal I/O area, the wait states are not inserted.



**Figure 9-5. Internal/External I/O Space Partition**

	7	6	5	4	3	2	1	0
Bit Name	—*	IOH2	IOH1	IOH0	—*	IOL2	IOL1	IOL0
Read/Write	—	R/W	R/W	R/W	—	R/W	R/W	R/W
Initial Value	0	1	1	1	0	1	1	1
		 I/O High				 I/O Low		

\* Reserved. These bits always read 0 and should be set to 0.

**Bit 7:** Reserved. This bit always reads 0 and should be set to 0.

**Bits 6-4: IOH2-0 (I/O High)**

The IOH bits specify the number of wait states to be inserted in the I/O cycle when an IOH address in an external I/O space is accessed. The table below lists bit values and the number of wait states.

IOH2	IOH1	IOH0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1*	1*	1*	7*

\* These are the initial values after a reset.

**Bit 3:** Reserved. This bit always reads 0 and should be set to 0.

#### Bits 2-0: IOL2-0 (I/O Low)

The IOL bits specify the number of wait states to be inserted in the I/O cycle when an IOL address in an external I/O space is accessed. The table below lists bit values and the number of wait states.

IOL2	IOL1	IOL0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1*	1*	1*	7*

\* These are the initial values after a reset.

#### 9.2.4 Interrupt Wait Control Register (INTWR)

INTWR specifies the number of wait states to be inserted in the first machine cycle of an  $\overline{\text{INT0}}$  interrupt acknowledge cycle.

Notes:

- For  $\overline{\text{INT}}_1$ ,  $\overline{\text{INT}}_2$ , or internal interrupts except TRAP, two wait states are automatically inserted. Insertion of more than two wait states is not possible using either the register setting or an external line control.
- For the first machine cycle of an  $\overline{\text{NMI}}$  interrupt acknowledge cycle, the area is determined according to the address value specified and the number of wait states specified by the corresponding wait control register are inserted, as they would be in an ordinary memory cycle. Wait state insertion by line control is done the same as for ordinary memory cycles.

	7	6	5	4	3	2	1	0
Bit Name	-*	-*	-*	-*	-*	INTW2	INTW1	INTW0
Read/Write	-	-	-	-	-	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	1

Interrupt Wait

\* Reserved. These bits always read 0 and should be set to 0.

**Bits 7-3:** Reserved. These bits always read 0 and should be set to 0.

**Bits 2-0: INTW2-0 (Interrupt Wait)**

The INTW bits specify the number of wait states to be inserted in the first machine cycle of the  $\overline{\text{INT}}_0$  interrupt acknowledge cycle. The table below lists bit values and the number of wait states.

INTW2	INTW1	INTW0	Number of Wait States
0	0	0	2
0	0	1	3
0	1	0	4
0	1	1	5
1	0	0	6
1	0	1	7
1	1	0	8
1*	1*	1*	9*

\* These are the initial values after a reset.

Figure 9-6 shows insertion timing of the wait state by  $\overline{\text{WAIT}}$  line control or programmable wait state in the first machine cycle of an  $\overline{\text{INT0}}$  interrupt acknowledge cycle.

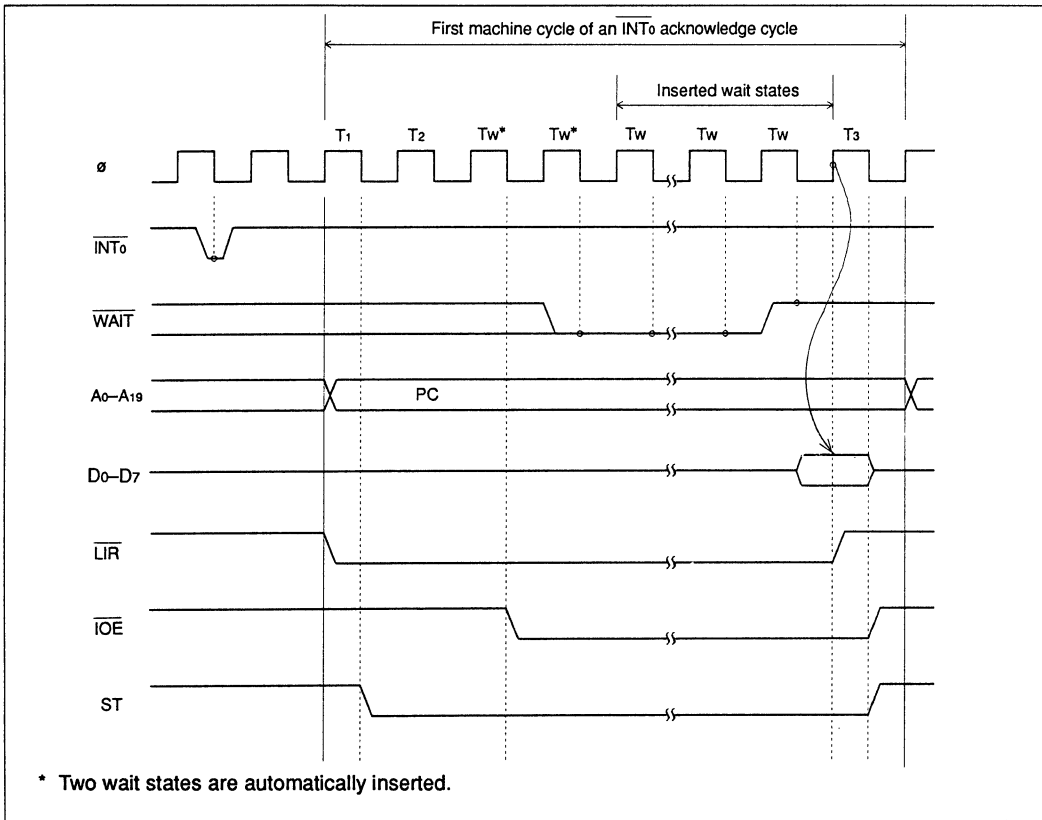


Figure 9-6. Wait State Insertion Timing in an  $\overline{\text{INT0}}$  Acknowledge Cycle

### 9.2.5 Refresh Wait Control Register (RWCR)

RWCR specifies the number of wait states to be inserted in a refresh cycle.

Note:

If 0 is specified as the number of wait states to be inserted in a refresh cycle using register control, wait insertion using  $\overline{\text{WAIT}}$  line control is not accepted.

	7	6	5	4	3	2	1	0
Bit Name	-*	-*	-*	-*	-*	REFW2	REFW1	REFW0
Read/Write	-	-	-	-	-	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	1

Refresh Wait

\* Reserved. These bits always read 0 and should be set to 0.

**Bits 7-3:** Reserved. These bits always read 0 and should be set to 0.

### Bits 2-0: REFW 2-0 (Refresh Wait)

The REFW bits specify the number of wait states to be inserted in a refresh cycle. The table below lists bit values and the number of wait states inserted.

REFW2	REFW1	REFW0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1*	1*	1*	7*

\* These are the initial values after a reset.

## 9.3 Operation

### 9.3.1 Wait State Insertion Using $\overline{\text{WAIT}}$ Line Control

In this wait state insertion using  $\overline{\text{WAIT}}$  line control, the wait state is inserted between T2 state and T3 state in the bus cycle T1-T3.

When the  $\overline{\text{WAIT}}$  line is held low, a wait state ( $T_w$ ) is inserted between bus cycles T2 and T3.

When the  $\overline{\text{WAIT}}$  line goes high, the cycle resumes in the T3 state.

Figure 9-7 shows the timing for inserting wait states using  $\overline{\text{WAIT}}$  line control.

The  $\overline{\text{WAIT}}$  line level is sampled at the falling edge of the  $\phi$  clock in the  $T_2$  or  $T_w$  state. During a refresh cycle, the  $\overline{\text{WAIT}}$  line level is sampled only at the falling edge of a  $T_w$  state, therefore sampling operation is not performed if wait state is not inserted by register control. Each time the  $\overline{\text{WAIT}}$  line is low at the falling edge of the  $\phi$  clock in the  $T_w$  state, another  $T_w$  state will be inserted.

There is no limit on the number of wait states that can be inserted.

Notes:

- When the  $\overline{\text{WAIT}}$  line signal is set at a low level, the set up time and hold time for falling edge of  $\phi$  clock must be accounted for by synchronizing to the rising  $\phi$  clock edge. If not, normal operation is not guaranteed.
- With the exception given above, wait states cannot be inserted in the acknowledge cycles for  $\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ , internal interrupts except TRAP, or internal I/O cycle. The number of wait states for these interrupts is fixed at 2 (for  $\overline{\text{INT1}}$ ,  $\overline{\text{INT2}}$ , and internal interrupts except TRAP) and 0 (for internal I/O cycles).
- Refresh cycles cannot be inserted between continuous wait cycles. When the refresh function is being used, this fact must be taken into account in determining the maximum number of wait states inserted.

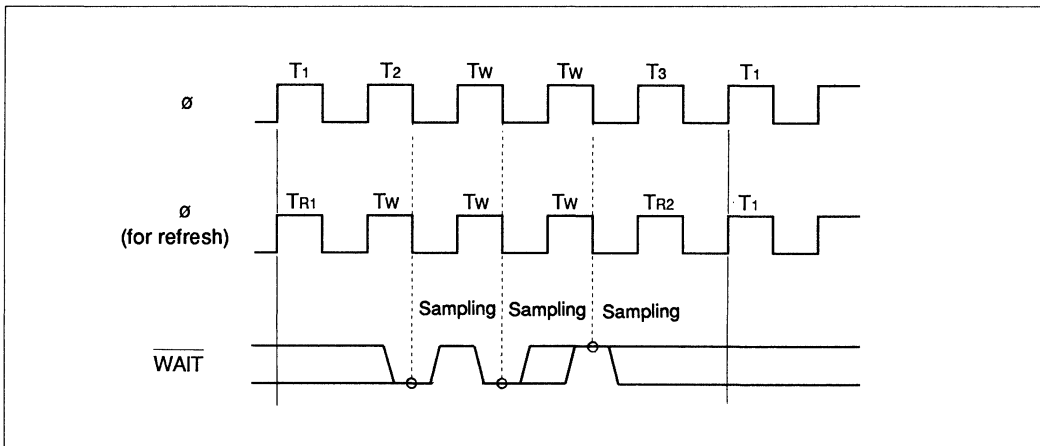


Figure 9-7. Wait State Insertion Timing Using  $\overline{\text{WAIT}}$  Line Control



### 9.3.2 Wait State Insertion Using Register Control

The wait control register inserts wait states in memory cycle, I/O cycle,  $\overline{\text{INT0}}$  interrupt acknowledge cycle and on refresh cycles, and specifies the number of wait states, obviating the need for an external circuit for this purpose. The number of wait states inserted in each bus cycle is programmable.

**Inserting Wait States in a Memory Cycle:** Wait states can be inserted according to memory specifications. Figure 9-8 shows an example for interfacing three different types of memory. In this example, wait states can be independently specified for each of the three types of memory. Physical address boundaries for dividing the three memory spaces are specified by the physical address boundary registers 0 and 1. For details, see section 9.2.1 "Physical Address Boundary Registers 0 and 1." The number of wait states to be inserted for each memory area is specified in the wait control registers: L, M, and H. For details, see section 9.2.2 "Wait Control Registers L, M, and H."

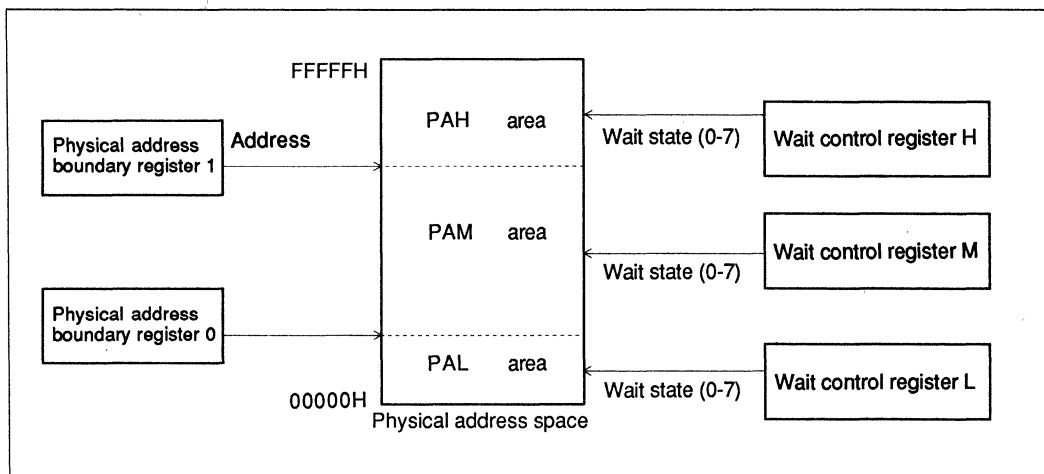
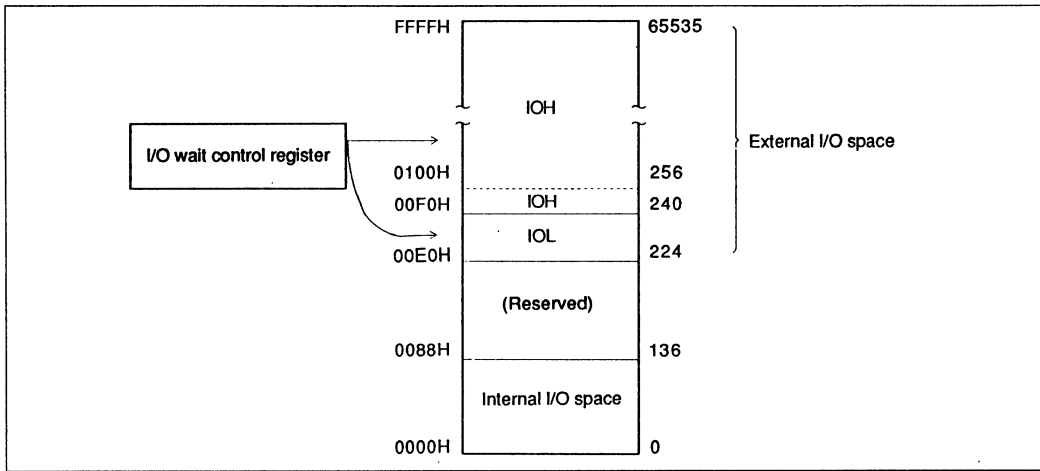


Figure 9-8. Memory Space Division and Wait State Insertion

**Inserting Wait States in an I/O Cycle:** The number of wait states to be inserted in an external I/O cycle is specified in the I/O wait control register. For details, see section 9.2.3 "I/O Wait Control Register." The external I/O space is partitioned at physical address 00F0H into the higher and lower address areas (IOH and IOL, respectively). Wait states can be independently specified for each IOH and IOL access. Wait states cannot be inserted in an internal I/O cycle.



**Figure 9-9. I/O Space Division and Wait States Insertion**

**Inserting Wait States in an  $\overline{INT0}$  Interrupt Acknowledge Cycle:** The number of wait states to be inserted in the first machine cycle of an  $\overline{INT0}$  acknowledge cycle is specified by the interrupt wait control register. For details, see section 9.2.4 "Interrupt Wait Control Register." This function allows a flexible access time for an interrupt exclusive device.

**Inserting Wait States in a Refresh Cycle:** The number of wait states to be inserted in a refresh cycle is specified by the refresh wait control register. For details, see section 9.2.5 "Refresh Wait Control Register" and figure 8-2. "Refresh Timing."

### 9.3.3 Wait State Controls

Table 9-2 summarizes wait state insertion conditions.

**Table 9-2 Wait State Insertion Conditions**

Operation cycle	$\overline{\text{WAIT}}$ Line wait insertion	Programmable number of wait states insertions	Register specifying number of wait states
Memory cycle	possible	0-7	Wait control registers L,M, and H
External I/O cycle	possible	0-7	I/O wait control register
Internal I/O cycle	not possible	0	–
Refresh cycle	possible* <sup>1</sup>	0-7	Refresh wait control register
1st machine cycle of $\overline{\text{INT0}}$ acknowledge cycle	possible	2-9 * <sup>2</sup>	Interrupt wait control register
1st machine cycle of $\overline{\text{INT1}}$ , $\overline{\text{INT2}}$ and internal interrupts except TRAP	not possible	2 * <sup>2</sup>	–
1st machine cycle of $\overline{\text{NMI}}$ acknowledge cycle	possible	0-7	Wait control registers L, M, and H

\*<sup>1</sup> Wait state insertion using the  $\overline{\text{WAIT}}$  line is possible only when programmable wait insertion is performed.

\*<sup>2</sup> Two wait states are automatically inserted.

## 9.4 Operation in Low Power Dissipation Mode

The wait controller continues operating in the sleep mode. Thus wait states can be inserted when DMAC operates in the sleep mode. In the system stop mode, the wait controller stops and retains the current register contents.

## 9.5 Reset Operation

Reset stops the wait controller and initializes the registers as follows:

- The wait control registers L, M, and H, I/O wait control register, interrupt wait control register and refresh wait control register are initialized so that the maximum number of wait states are inserted.

- The physical address boundary registers 0 and 1 are initialized to 00H. This results in the physical address space consisting of the PAL area only. Accordingly, the number of wait states specified in the wait control register L is inserted in a memory cycle.

## 9.6 Precautions

If wait state insertion is requested by register control simultaneously with  $\overline{\text{WAIT}}$  line control, wait states of the number specified in the register are inserted. If the  $\overline{\text{WAIT}}$  line requests more wait states, the additional wait states are inserted.

## Section 10. Chip Select Control

### 10.1 Chip Select Line Operation

Chip select lines  $\overline{CS_0}$ ,  $\overline{CS_1}$ , and  $\overline{CS_2}$  indicate that physical address spaces PAL area, PAM area, and PAH area are being accessed. (For details, see section 9.2.1 "Physical Address Boundary Registers 0 and 1" and section 9.3.2 "Wait State Insertion Using Register Control").

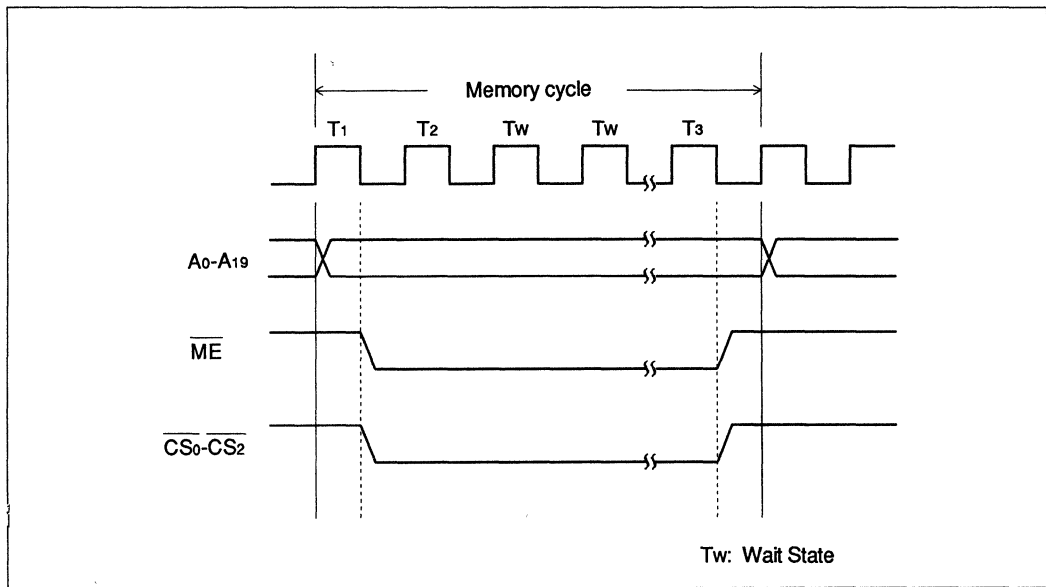
Note that FP-80A package version does not have  $\overline{CS_2}$  line.

Table 10-1 shows the relationship between lines and their associated physical address spaces.

**Table 10-1.  $\overline{CS}$  Lines and Associated Physical Address Spaces**

$\overline{CS}$ Line Asserted	Associated Physical Address Area
$\overline{CS_0}$ line	PAL area
$\overline{CS_1}$ line	PAM area
$\overline{CS_2}$ line	PAH area

Figure 10-1 shows chip select timing.



**Figure 10-1. Chip Select Timing**

## 10.2 Operation in Low Power Dissipation Mode

While the DMAC is operating in the sleep mode, the chip select line corresponding to the accessed physical address area is driven low. When the DMAC is not operating in the sleep mode or when the system stop mode is entered, the chip select lines are set to 1.

## 10.3 Reset Operation

The chip select lines go high after a reset. Immediately after a reset is cleared, the physical address space equals the physical address lower area and thus the corresponding  $\overline{CS}_0$  line is driven low. (For details, see section 9.2.1 "Physical Address Boundary Registers 0 and 1").

## 10.4 Precautions

The chip select lines are asserted only during memory cycles (not during refresh cycles and I/O cycles).

## Section 11. Low Power Dissipation Modes

The HD64180S supports two low power dissipation modes: sleep and system stop. These modes are selected using the I/O control register (IOCR). The IOCR is contained in the CPU and allocated to I/O address 0005H. For details, see section 3.4.5 "Sleep Mode."

### 11.1 Sleep Mode

Executing an SLP instruction when the IOSTP bit (in IOCR) is 0 causes the HD64180S to enter the sleep mode. In this mode, the CPU stops, but other areas (MSCI, ASCI/CSIO, DMAC, refresh controller, and timer) remain active.

Asserting the  $\overline{\text{BUSREQ}}$  signal causes the HD64180S to enter the bus release mode. The HD64180S leaves the sleep mode when a reset or interrupt is detected.

For details, see section 3.4.5 "Sleep Mode."

### 11.2 System Stop Mode

Executing an SLP instruction when the IOSTP bit (in IOCR) is 1 causes the HD64180S to enter the system stop mode. In this mode, clocks for the CPU and other functions stop. Less power is dissipated in this mode compared to the sleep mode.

In the system stop mode, asserting the  $\overline{\text{BUSREQ}}$  line causes the HD64180S to enter the bus release mode. The HD64180S leaves the system stop mode when a reset or external interrupt is detected. For details, see section 3.4.6 "System Stop Mode."

## Section 12. Oscillator Circuit

### 12.1 Crystal Resonator and Oscillator Circuit

The HD64180S contains an on-chip oscillator. The  $\phi$  clock can be generated by connecting a crystal resonator to the XTAL and EXTAL lines. (The crystal must be an AT-cut parallel resonator). The output of the oscillator is connected to a driver (see figure 12-1) that divides the oscillator frequency by two. Therefore, the oscillator frequency must be double that of the required  $\phi$  clock frequency.

An external clock can also be supplied directly via the EXTAL line (for details, see section 12.3 "Operation Using an External Clock").

Note that the external clock is also divided by two to generate the  $\phi$  clock.

A baud rate generator (BRG) is contained in both the MSCI and ASCI/CSIO. The  $\phi$  clock is used as the reference clock for the BRG.

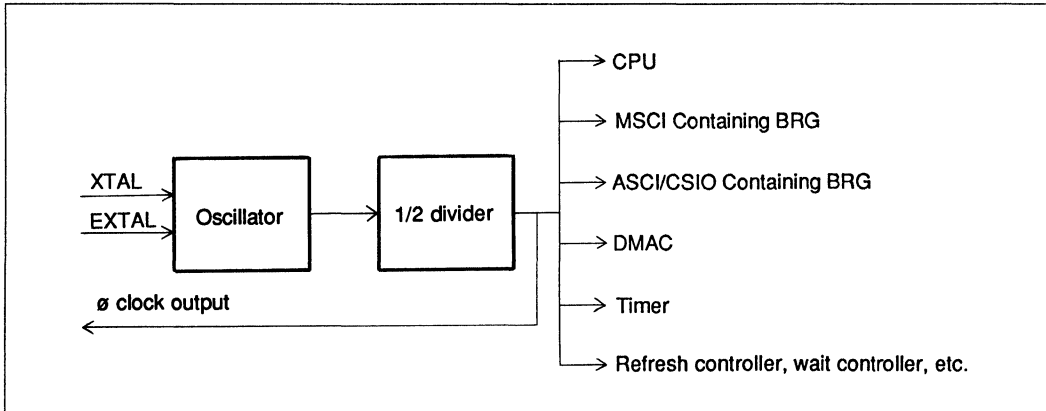


Figure 12-1.  $\phi$  Clock Generation and Supplies to Each Function Block



Figure 12-2 shows an example of a crystal connection circuit.

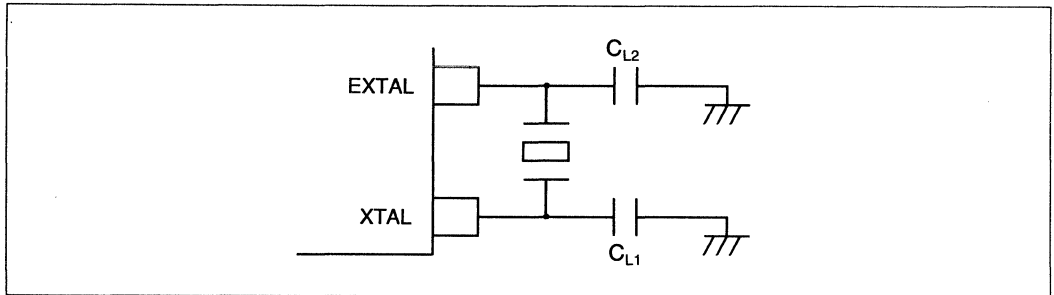


Figure 12-2. Crystal Connection Circuit

The crystal resonator and load capacitor must satisfy the characteristics listed in table 12-1.

Table 12-1. Recommended Characteristics of Crystal Resonator and Load Capacitor

Item	Oscillation Frequency	
	1 MHz < f ≤ 12.288 MHz	12.288 MHz < f ≤ 20 MHz
C <sub>0</sub>	< 7 pF	< 7 pF
R <sub>s</sub>	< 60 Ω	< 35 Ω
C <sub>L1</sub> , C <sub>L2</sub>	10-22 pF±10%	10-22 pF±10%

## 12.2 Oscillator Circuit Board Design

When connecting the output of a crystal resonator to the XTAL and EXTAL lines, the following design precautions must be observed:

1. Locate the crystal resonator and load capacitors (C<sub>L1</sub> and C<sub>L2</sub>) as close to the chip as possible. If noise is introduced on the XTAL line or EXTAL line, normal operation cannot be guaranteed.
2. The signal leads to the XTAL pin and φ pin should be positioned as far apart as possible and not parallel to one another. If the φ signal induces noise in the XTAL input, normal oscillation cannot be guaranteed. (See figure 12-3 (a).)
3. Do not run signal or power lines near the oscillator circuit. Figure 12-3 (b) shows an undesirable layout. Induced noise may cause operating errors. Isolation between XTAL, EXTAL, and adjacent lines must be 10 MΩ or more.

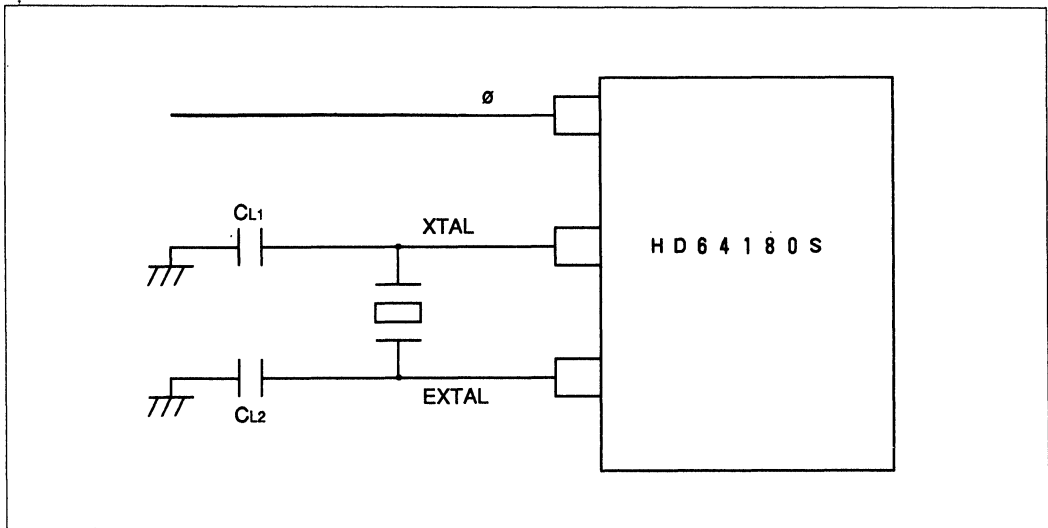


Figure 12-3. (a) Undesirable Oscillator Circuit Layout

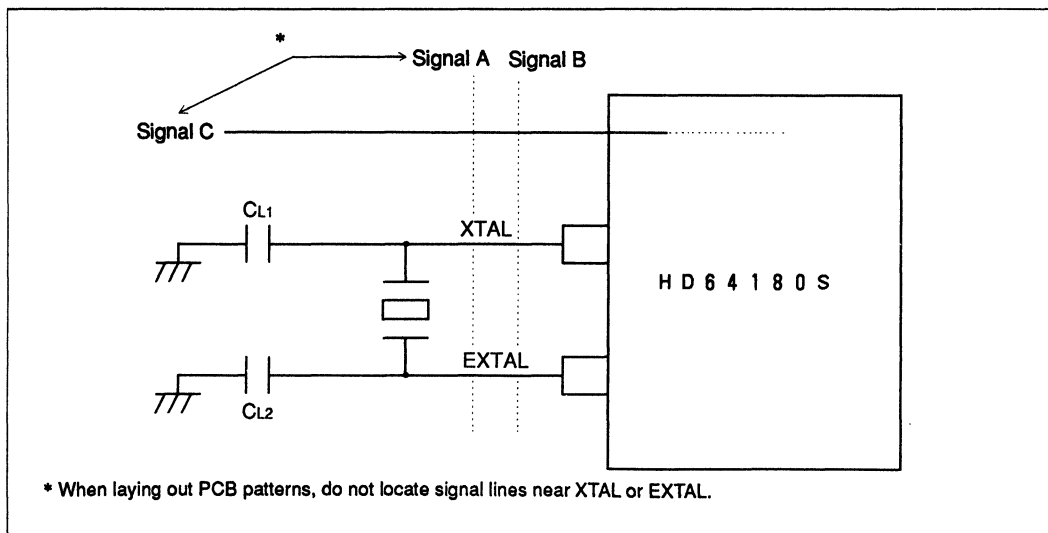


Figure 12-3. (b) Undesirable Oscillator Circuit Layout

Figure 12-4 shows an example of oscillator circuit board design.

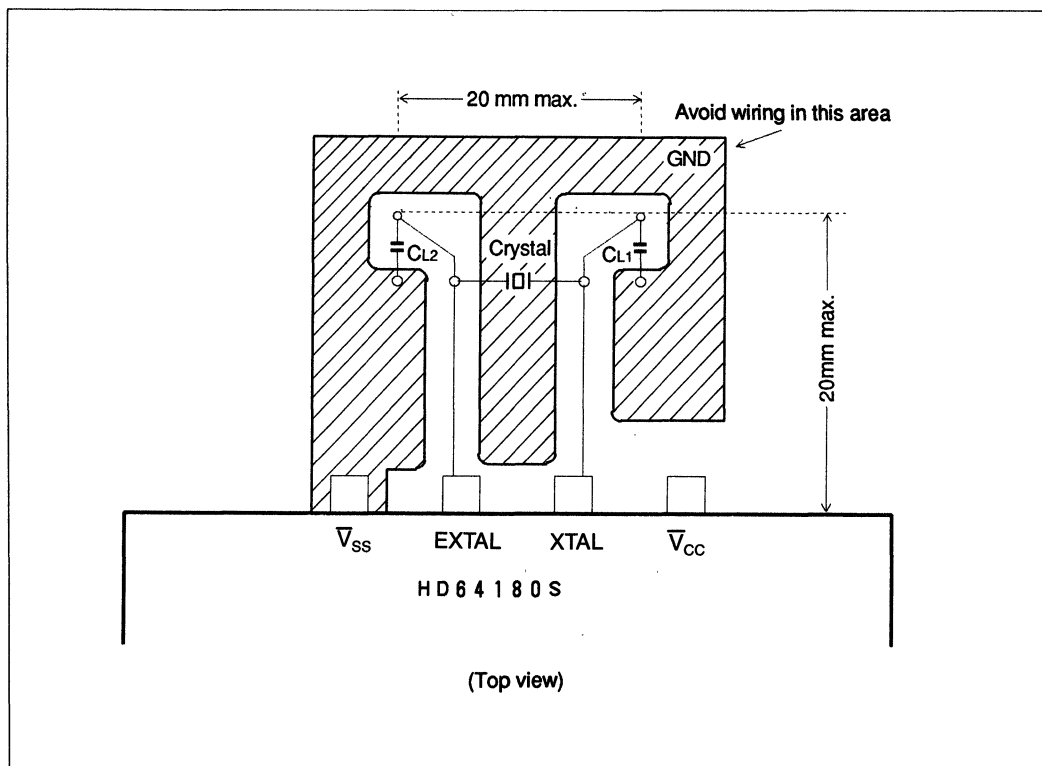
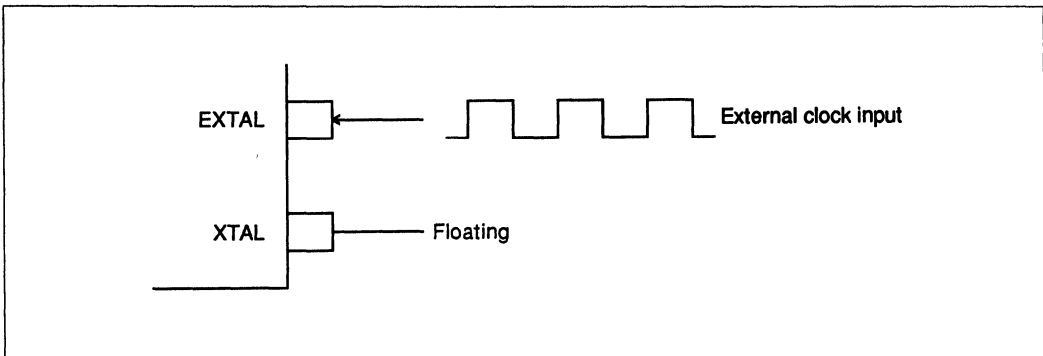


Figure 12-4. Example of Oscillator Circuit Layout

### 12.3 Operation Using an External Clock

The HD64180S can be operated using an external clock (connected to the EXTAL line). In this case, the input frequency must be double that of the  $\phi$  clock and the XTAL line must be left floating.

Figure 12-5 shows a typical connection for an external clock.



**Figure 12-5. Example of Connection for External Clock**

## Section 13. Electrical Specifications

### 13.1 Absolute Maximum Ratings

Table 13-1. Absolute Maximum Ratings

Item	Symbol	Rating	Unit
Supply voltage	V <sub>cc</sub>	-0.3 to +7.0	V
Input voltage	V <sub>in</sub>	-0.3 to V <sub>cc</sub> + 0.3	V
Operating temperature	T <sub>opr</sub>	-20 to +75	°C
Storage temperature	T <sub>stg</sub>	-55 to +150	°C

**Caution:** Permanent damage to the HD64180S may result if it is subjected to conditions that exceed the absolute maximum ratings. To assure normal operation, the following conditions should be satisfied:

$$V_{ss} \leq V_{in} \leq V_{cc}$$

### 13.2 DC Characteristics

Table 13-2. DC Characteristics

(V<sub>cc</sub> = 5V ± 10%, V<sub>ss</sub> = 0V, T<sub>a</sub> = -20 to +75°C unless otherwise specified)

Item	Symbol	min	typ	max	Unit	Conditions
Input high level voltage at $\overline{\text{RESET}}$ , $\overline{\text{EXTAL}}$ , and $\overline{\text{NMI}}$	V <sub>IH1</sub>	V <sub>cc</sub> -0.6	----	V <sub>cc</sub> +0.3	V	
Input high level voltage at lines other than $\overline{\text{RESET}}$ , $\overline{\text{EXTAL}}$ , and $\overline{\text{NMI}}$	V <sub>IH2</sub>	2.2	----	V <sub>cc</sub> +0.3	V	
Input low level voltage at $\overline{\text{RESET}}$ , $\overline{\text{EXTAL}}$ , and $\overline{\text{NMI}}$	V <sub>IL1</sub>	-0.3	----	0.6	V	
Input low level voltage at lines other than $\overline{\text{RESET}}$ , $\overline{\text{EXTAL}}$ , and $\overline{\text{NMI}}$	V <sub>IL2</sub>	-0.3	----	0.8	V	
Output high level voltage at all output lines	V <sub>OH</sub>	2.4 V <sub>cc</sub> - 1.2	----	----	V	I <sub>OH</sub> = -200 μA I <sub>OH</sub> = -20 μA
Output low level voltage at all output lines	V <sub>OL</sub>	----	----	0.45	V	I <sub>OL</sub> = 2.2 mA

**Table 13-2. DC Characteristics (cont.)**

( $V_{CC} = 5V \pm 10\%$ ,  $V_{SS} = 0V$ ,  $T_a = -20$  to  $+75^\circ C$  unless otherwise specified)

Item	Symbol	min	typ	max	Unit	Conditions
Input leakage current at all input lines other than XTAL and EXTAL	$I_{IL}$	----	----	1.0	$\mu A$	$V_{in} = 0.5$ to $V_{CC} - 0.5$
Three state leakage current	$I_{TL}$	----	----	1.0	$\mu A$	$V_{in} = 0.5$ to $V_{CC} - 0.5$
Current dissipation* (normal operation)	$I_{CC}$	----	36	72	mA	$f = 6$ MHz
		----	48	96		$f = 8$ MHz
		----	60	120		$f = 10$ MHz
Current dissipation* (system stop mode)		----	6	12	mA	$f = 6$ MHz
		----	8	16		$f = 8$ MHz
		----	10	20		$f = 10$ MHz
Pin capacitance	$C_p$	----	----	20	pF	$V_{in} = 0V$ , $f = 1$ MHz, $T_a = 25^\circ C$

\* Input signal

RESET, EXTAL,  $\overline{NMI}$ :  $V_{IHmin} = V_{CC} - 0.6V$ ,  $V_{ILmax} = 0.6V$  the others:  $V_{IHmin} = V_{CC} - 1.0V$ ,  $V_{ILmax} = 0.8V$   
All output terminals are at no load.

### 13.3 AC Characteristics

Note that the specifications related to  $\overline{CS}_2$  pin is specified only in CP-84 package version.

#### 13.3.1 Bus Timing

**Table 13-3. Bus Timing**

( $V_{CC} = 5V \pm 10\%$ ,  $V_{SS} = 0V$ ,  $T_a = -20$  to  $+75^\circ C$  unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
Clock cycle time	$t_{CYC}$	162	—	2000	125	—	2000	100	—	2000	ns	See figures
Clock high-level pulse width	$t_{CHW}$	65	—	—	50	—	—	38	—	—	ns	13-1, 13-2, 13-3, and
Clock low-level pulse width	$t_{CLW}$	65	—	—	50	—	—	38	—	—	ns	13-4.
Clock fall time	$t_{CF}$	—	—	15	—	—	15	—	—	12	ns	
Clock rise time	$t_{CR}$	—	—	15	—	—	15	—	—	12	ns	
Address delay time	$t_{AD}$	—	—	90	—	—	80	—	—	55	ns	

**Table 13-3. Bus Timing (cont.)**

 (V<sub>CC</sub> = 5V ± 10%, V<sub>SS</sub> = 0V, T<sub>a</sub> = -20 to +75°C unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
Address set-up time (vis-a-vis falling edge of $\overline{ME}$ , $\overline{IOE}$ , or $\overline{CS2} - \overline{CS0}$ )	t <sub>AS</sub>	20	-	-	15	-	-	15	-	-	ns	See figures 13-1, 13-2, 13-3, and 13-4.
$\overline{ME}$ delay time 1	t <sub>MED1</sub>	-	-	60	-	-	50	-	-	50	ns	
$\overline{RD}$ delay time 1	t <sub>RDD1</sub>	-	-	60	-	-	50	-	-	50	ns	
$\overline{LIR}$ delay time 1	t <sub>LD1</sub>	-	-	80	-	-	70	-	-	55	ns	
Address hold time (vis-a-vis rising edge of $\overline{ME}$ , $\overline{IOE}$ , $\overline{RD}$ , $\overline{WR}$ or $\overline{CS2} - \overline{CS0}$ )	t <sub>AH</sub>	35	-	-	20	-	-	10	-	-	ns	
$\overline{ME}$ delay time 2	t <sub>MED2</sub>	-	-	60	-	-	50	-	-	50	ns	
$\overline{RD}$ delay time 2	t <sub>RDD2</sub>	-	-	60	-	-	50	-	-	50	ns	
$\overline{RD}$ delay time 3	t <sub>RDD3</sub>	-	-	65	-	-	60	-	-	55	ns	
$\overline{LIR}$ delay time 2	t <sub>LD2</sub>	-	-	80	-	-	70	-	-	55	ns	
Data read set-up time	t <sub>DRS</sub>	40	-	-	30	-	-	30	-	-	ns	
Data read hold time*	t <sub>DRH</sub>	0	-	-	0	-	-	0	-	-	ns	
ST delay time 1	t <sub>STD1</sub>	-	-	90	-	-	70	-	-	60	ns	
ST delay time 2	t <sub>STD2</sub>	-	-	90	-	-	70	-	-	60	ns	
WAIT set-up time	t <sub>WS</sub>	40	-	-	40	-	-	30	-	-	ns	

 \* Defined against the first signal to go high level of  $\overline{ME}$ ,  $\overline{RD}$  and  $\overline{CS2} - \overline{CS0}$

**Table 13-3. Bus Timing (cont.)**

(V<sub>CC</sub> = 5V ± 10%, V<sub>SS</sub> = 0V, T<sub>a</sub> = -20 to +75°C unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
WAIT hold time	t <sub>WH</sub>	40	-	-	40	-	-	30	-	-	ns	See figures 13-1, 13-2, 13-3, and 13-4.
Write data floating delay time	t <sub>WDZ</sub>	-	-	95	-	-	70	-	-	60	ns	
WR delay time 1	t <sub>WRD1</sub>	-	-	65	-	-	60	-	-	50	ns	
Write data delay time	t <sub>WDD</sub>	-	-	90	-	-	80	-	-	60	ns	
Write data set-up time (vis-a-vis falling edge of $\overline{WR}$ )	t <sub>WDS</sub>	40	-	-	20	-	-	15	-	-	ns	
$\overline{WR}$ delay time 2	t <sub>WRD2</sub>	-	-	80	-	-	60	-	-	55	ns	
$\overline{WR}$ pulse width	t <sub>WRP</sub>	170	-	-	130	-	-	110	-	-	ns	
Write data hold time (vis-a-vis rising edge of $\overline{WR}$ )	t <sub>WDH</sub>	40	-	-	15	-	-	10	-	-	ns	
$\overline{IOE}$ delay time 1	t <sub>IOD1</sub>	-	-	60	-	-	50	-	-	50	ns	
$\overline{IOE}$ delay time 2	t <sub>IOD2</sub>	-	-	60	-	-	50	-	-	50	ns	
$\overline{IOE}$ delay time 3 (from falling edge of $\overline{LIR}$ )	t <sub>IOD3</sub>	340	-	-	250	-	-	200	-	-	ns	
$\overline{IOE}$ delay time 4	t <sub>IOD4</sub>	-	-	65	-	-	60	-	-	55	ns	
INT set-up time (vis-a-vis falling edge of $\emptyset$ )	t <sub>INTS</sub>	40	-	-	40	-	-	30	-	-	ns	
INT hold time (vis-a-vis falling edge of $\emptyset$ )	t <sub>INTH</sub>	40	-	-	40	-	-	30	-	-	ns	



**Table 13-3. Bus Timing (cont.)**

(Vcc = 5V ± 10%, Vss = 0V, Ta = -20 to +75°C unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
$\overline{\text{NMI}}$ pulse width	t <sub>NMIW</sub>	120	–	–	100	–	–	80	–	–	ns	See figures
$\overline{\text{BUSREQ}}$ set-up time (vis-a-vis falling edge of $\phi$ )	t <sub>BRS</sub>	40	–	–	40	–	–	30	–	–	ns	13-1, 13-2, 13-3, and 13-4.
$\overline{\text{BUSREQ}}$ hold time (vis-a-vis falling edge of $\phi$ )	t <sub>BRH</sub>	40	–	–	40	–	–	30	–	–	ns	
$\overline{\text{BUSACK}}$ delay time 1	t <sub>BAD1</sub>	–	–	95	–	–	70	–	–	60	ns	
$\overline{\text{BUSACK}}$ delay time 2	t <sub>BAD2</sub>	–	–	95	–	–	70	–	–	60	ns	
Bus floating delay time	t <sub>BZD</sub>	–	–	125	–	–	90	–	–	80	ns	
$\overline{\text{ME}}$ high-level pulse width	t <sub>MEWH</sub>	110	–	–	90	–	–	70	–	–	ns	
$\overline{\text{ME}}$ low-level pulse width	t <sub>MEWL</sub>	125	–	–	100	–	–	80	–	–	ns	
$\overline{\text{REF}}$ delay time 1	t <sub>RFD1</sub>	–	–	90	–	–	80	–	–	60	ns	
$\overline{\text{REF}}$ delay time 2	t <sub>RFD2</sub>	–	–	90	–	–	80	–	–	60	ns	
$\overline{\text{HALT}}$ delay time 1	t <sub>HAD1</sub>	–	–	90	–	–	80	–	–	50	ns	
$\overline{\text{HALT}}$ delay time 2	t <sub>HAD2</sub>	–	–	90	–	–	80	–	–	50	ns	
$\overline{\text{RESET}}$ set-up time	t <sub>RES</sub>	120	–	–	100	–	–	80	–	–	ns	
$\overline{\text{RESET}}$ hold time	t <sub>REH</sub>	80	–	–	80	–	–	80	–	–	ns	
Oscillator stabilize time	t <sub>OSC</sub>	–	–	20	–	–	20	–	–	40	ms	

**Table 13-3. Bus Timing (cont.)**

(V<sub>CC</sub> = 5V ± 10%, V<sub>SS</sub> = 0V, T<sub>a</sub> = -20 to +75°C unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
RESET rise time	t <sub>RE</sub>	-	-	50	-	-	50	-	-	50	ms	See figures
RESET fall time	t <sub>RF</sub>	-	-	50	-	-	50	-	-	50	ms	13-1, 13-2,
CS delay time 1	t <sub>CSD1</sub>	-	-	60	-	-	55	-	-	50	ns	13-3, and
CS delay time 2	t <sub>CSD2</sub>	-	-	60	-	-	55	-	-	50	ns	13-4.

### 13.3.2 MSCI Timing

**Table 13-4. MSCI Timing**

(V<sub>CC</sub> = 5V ± 10%, V<sub>SS</sub> = 0V, T<sub>a</sub> = -20 to +75°C unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
TXCM cycle time (TXCM input)	t <sub>RCYM</sub>	1.4*	-	-	1.4*	-	-	1.4*	-	-	t <sub>CYC</sub>	See figures 13-5, 13-6,
TXCM rise time (TXCM input)	t <sub>RCM</sub>	-	-	20	-	-	15	-	-	10	ns	13-7, 13-8, 13-9, 13-
TXCM fall time (TXCM input)	t <sub>RCM</sub>	-	-	20	-	-	15	-	-	10	ns	10, 13-11, 13-12, and
TXCM high-level pulse width (TXCM input)	t <sub>RCHWM</sub>	0.55	-	-	0.55	-	-	0.55	-	-	t <sub>CYC</sub>	13-13.
TXCM low-level pulse width (TXCM input)	t <sub>RCLWM</sub>	0.55	-	-	0.55	-	-	0.55	-	-	t <sub>CYC</sub>	
TXDM delay time (TXCM input)	t <sub>TDD1M</sub>	-	-	130	-	-	100	-	-	80	ns	
TXDM delay time (TXCM output)	t <sub>TDD2M</sub>	-	-	80	-	-	65	-	-	50	ns	
RXCM cycle time	t <sub>RCYM</sub>	1.4*	-	-	1.4*	-	-	1.4*	-	-	t <sub>CYC</sub>	

\* In asynchronous mode, loop mode, t<sub>RCYM</sub>, t<sub>RCYM</sub> = 2.5 t<sub>CYC</sub> (min).

**Table 13-4. MSCI Timing (cont.)**

(V<sub>CC</sub> = 5V ± 10%, V<sub>SS</sub> = 0V, T<sub>a</sub> = -20 to +75°C unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
RXCM rise time	t <sub>RCM</sub>	-	-	20	-	-	15	-	-	10	ns	See figures
RXCM fall time	t <sub>RCM</sub>	-	-	20	-	-	15	-	-	10	ns	13-5, 13-6,
RXCM high-level pulse width	t <sub>RCHWM</sub>	0.55	-	-	0.55	-	-	0.55	-	-	t <sub>cy</sub>	13-7, 13-8, 13-9, 13-
RXCM low-level pulse width	t <sub>RCLWM</sub>	0.55	-	-	0.55	-	-	0.55	-	-	t <sub>cy</sub>	10, 13-11, 13-12, and
RXDM-RXCM set-up time (RXCM input)	t <sub>RDS1M</sub>	50	-	-	40	-	-	30	-	-	ns	13-13.
RXCM-RXDM hold time (RXCM input)	t <sub>RDH1M</sub>	40	-	-	30	-	-	20	-	-	ns	
RXDM-RXCM set-up time (RXCM output)	t <sub>RDS2M</sub>	130	-	-	100	-	-	80	-	-	ns	
RXCM-RXDM hold time (RXCM output)	t <sub>RDH2M</sub>	40	-	-	30	-	-	20	-	-	ns	
ADPLL operating clock cycle time	t <sub>PLCYM</sub>	120	-	-	80	-	-	57	-	-	ns	
ADPLL operating clock rise time	t <sub>PLM</sub>	-	-	15	-	-	10	-	-	8	ns	
ADPLL operating clock fall time	t <sub>PLM</sub>	-	-	15	-	-	10	-	-	8	ns	
ADPLL operating clock high-level pulse width	t <sub>PLHWM</sub>	25	-	-	15	-	-	10	-	-	ns	
ADPLL operating clock low-level pulse width	t <sub>PLLWM</sub>	25	-	-	15	-	-	10	-	-	ns	
φ-BRG* output delay time	t <sub>BGDM</sub>	-	-	150	-	-	120	-	-	95	ns	

\* f<sub>BRG</sub> ≠ f<sub>φ</sub> (f<sub>BRG</sub> is the baud rate generator output frequency; f<sub>φ</sub> is the CPU operating clock frequency).

**Table 13-4. MSCI Timing (cont.)**

(Vcc = 5V ± 10%, Vss = 0V, Ta = -20 to +75°C unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
TXCM/RXCM output rise time	tBGM <sub>r</sub>	-	-	50	-	-	40	-	-	30	ns	See figures
TXCM/RXCM output fall time	tBGM <sub>f</sub>	-	-	50	-	-	40	-	-	30	ns	13-5, 13-6, 13-7, 13-8, 13-9, 13-
RXCM- $\overline{\text{SYNC}}$ set-up time	tsYSU	2.5	-	-	2.5	-	-	2.5	-	-	tcyc	10, 13-11, 13-12, and
RXCM- $\overline{\text{SYNC}}$ hold time	tsYHD	2.5	-	-	2.5	-	-	2.5	-	-	tcyc	13-13.
CTSM high-level pulse width	tCTSHWM	2.0	-	-	2.0	-	-	2.0	-	-	tcyc	
CTSM low-level pulse width	tCTSLWM	2.0	-	-	2.0	-	-	2.0	-	-	tcyc	
DCDM high-level pulse width	tDCDHWM	2.0	-	-	2.0	-	-	2.0	-	-	tcyc	
DCDM low-level pulse width	tDCDLWM	2.0	-	-	2.0	-	-	2.0	-	-	tcyc	
$\phi$ -RTSM delay time	tRTSDM	-	-	100	-	-	85	-	-	70	ns	

### 13.3.3 ASCI/CSIO Timing

**Table 13-5. ASCI/CSIO Timing**

(V<sub>CC</sub> = 5V ± 10%, V<sub>SS</sub> = 0V, T<sub>a</sub> = -20 to + 75°C unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
TXCA input cycle time	tr <sub>CYC</sub>	2.5	-	-	2.5	-	-	2.5	-	-	tcyc	See figures 13-14, 13-
TXCA input high-level pulse width	tr <sub>CHW</sub>	0.55	-	-	0.55	-	-	0.55	-	-	tcyc	15, 13-16, 13-17, 13-
TXCA input low-level pulse width	tr <sub>CLW</sub>	0.55	-	-	0.55	-	-	0.55	-	-	tcyc	18, 13-19, and 13-20.
TXCA input rise time	tr <sub>CR</sub>	-	-	30	-	-	20	-	-	10	ns	
TXCA input fall time	tr <sub>CF</sub>	-	-	30	-	-	20	-	-	10	ns	
TXDA delay time 1	tr <sub>DD1</sub>	1.5	-	3.0	1.5	-	3.0	1.5	-	3.0	tcyc	
TXDA delay time 2	tr <sub>DD2</sub>	-	-	50	-	-	40	-	-	30	ns	
RXCA input cycle time	tr <sub>CYC</sub>	2.5	-	-	2.5	-	-	2.5	-	-	tcyc	
RXCA input high-level pulse width	tr <sub>CHW</sub>	0.55	-	-	0.55	-	-	0.55	-	-	tcyc	
RXCA input low-level pulse width	tr <sub>CLW</sub>	0.55	-	-	0.55	-	-	0.55	-	-	tcyc	
RXCA input rise time	tr <sub>CR</sub>	-	-	30	-	-	20	-	-	10	ns	
RXCA input fall time	tr <sub>CF</sub>	-	-	30	-	-	20	-	-	10	ns	

**Table 13-5. ASCII/CSIO Timing (cont.)**

(V<sub>CC</sub> = 5V ± 10%, V<sub>SS</sub> = 0V, T<sub>a</sub> = -20 to +75°C unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
RXDA set-up time 1	trDS1	50	-	-	40	-	-	30	-	-	ns	See figures 13-14, 13-15, 13-16, 13-17, 13-18, 13-19, and 13-20.
RXDA hold time 1	trDH1	40	-	-	30	-	-	20	-	-	ns	
RXDA set-up time 2	trDS2	130	-	-	100	-	-	80	-	-	ns	
RXDA hold time 2	trDH2	40	-	-	30	-	-	20	-	-	ns	
φ-BRG output delay time	tBGDA	-	-	80	-	-	70	-	-	60	ns	
TXCA/RXCA output rise time	tBGA <sub>r</sub>	-	-	50	-	-	40	-	-	30	ns	
TXCA/RXCA output fall time	tBGA <sub>f</sub>	-	-	50	-	-	40	-	-	30	ns	
CTSA high-level pulse width	tCTSHW	2.0	-	-	2.0	-	-	2.0	-	-	tcyc	
CTSA low-level pulse width	tCTSLW	2.0	-	-	2.0	-	-	2.0	-	-	tcyc	
DCDA high-level pulse width	tDCDHW	2.0	-	-	2.0	-	-	2.0	-	-	tcyc	
DCDA low-level pulse width	tDCDLW	2.0	-	-	2.0	-	-	2.0	-	-	tcyc	
RTSA delay time	trTSD	-	-	100	-	-	85	-	-	70	ns	

### 13.3.4 DMAC Timing

**Table 13-6. DMAC Timing**

(Vcc = 5V ± 10%, Vss = 0V, Ta = -20 to +75°C unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
DREQ set-up time	tDREQS	40	-	-	40	-	-	30	-	-	ns	See figure 13-21.
DREQ hold time	tDREQH	40	-	-	40	-	-	30	-	-	ns	
TEND delay time 1	tTED1	-	-	70	-	-	60	-	-	50	ns	
TEND delay time 2	tTED2	-	-	70	-	-	60	-	-	50	ns	
ST delay time 1	tSTD1	-	-	90	-	-	70	-	-	60	ns	
ST delay time 2	tSTD2	-	-	90	-	-	70	-	-	60	ns	

### 13.3.5 Timer Timing

**Table 13-7. Timer Timing**

(Vcc = 5V ± 10%, Vss = 0V, Ta = -20 to +75°C unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
Timer input pulse width	tPWT	2.0	-	-	2.0	-	-	2.0	-	-	tcvc	See figure 13-22.
Timer input set-up time	tPDSU	40	-	-	40	-	-	30	-	-	ns	
Timer input hold time	tPDH	40	-	-	40	-	-	30	-	-	ns	
Timer output delay time	tTOD	-	-	100	-	-	85	-	-	70	ns	

### 13.3.6 EXTAL Input Clock Signal Timing

**Table 13-8. EXTAL Input Clock Signal Timing**

(VCC = 5V ± 10%, VSS = 0V, Ta = -20 to +75°C unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
External clock cycle time	t <sub>CYC</sub>	81	-	1000	62	-	1000	50	-	1000	ns	See figure 13-23.
External clock high-level pulse width	t <sub>CHW</sub>	20	-	-	15	-	-	10	-	-	ns	
External clock low-level pulse width	t <sub>CLW</sub>	20	-	-	15	-	-	10	-	-	ns	
External clock fall time	t <sub>CF</sub>	-	-	25	-	-	25	-	-	15	ns	
External clock rise time	t <sub>CR</sub>	-	-	25	-	-	25	-	-	15	ns	

### 13.3.7 Miscellaneous

**Table 13-9. Rise and Fall Times of Input Signals with No Characteristics Specified**

(VCC = 5V ± 10%, VSS = 0V, Ta = -20 to +75°C unless otherwise specified)

Item	Symbol	HD64180SCP6			HD64180SCP8			HD64180SCP10			Unit	Timing
		min	typ	max	min	typ	max	min	typ	max		
Input line rise time (no characteristics specified)	t <sub>r</sub>	-	-	100	-	-	100	-	-	100	ns	See Figure 13-24.
Input line fall time (no characteristics specified)	t <sub>f</sub>	-	-	100	-	-	100	-	-	100	ns	



# 13.4 Timing Diagrams

## 13.4.1 Bus Timing

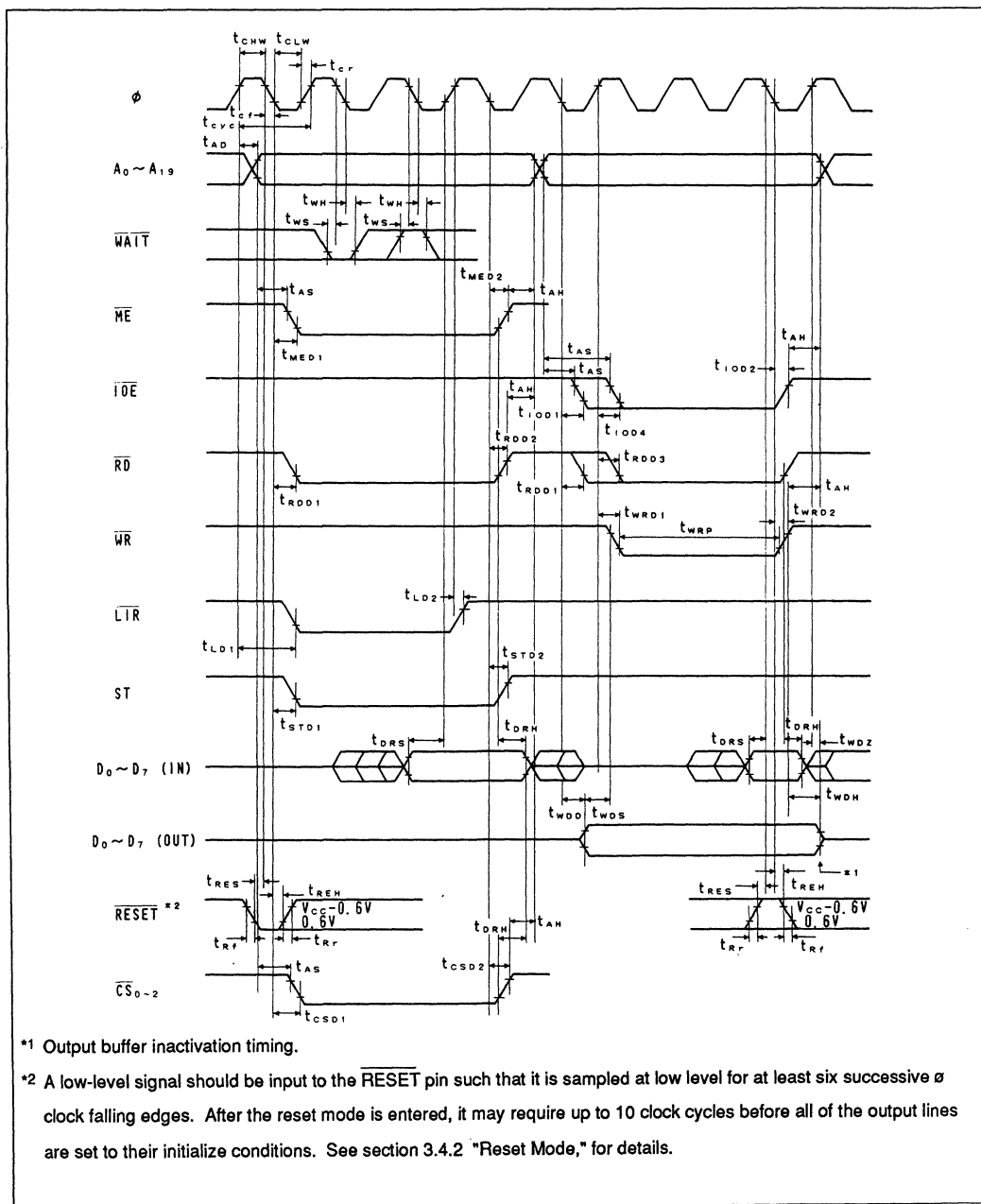


Figure 13-1. Bus Timing (1)

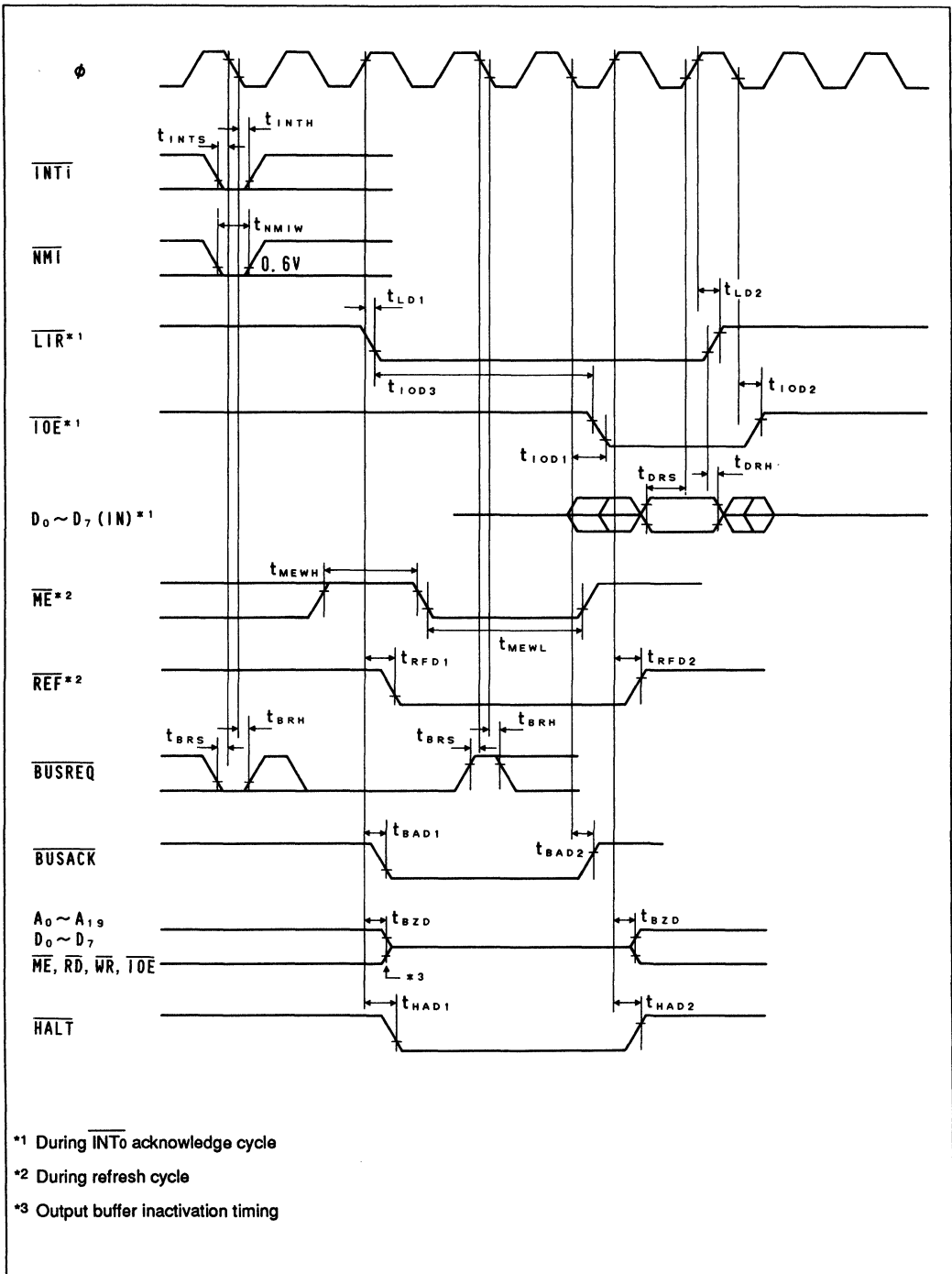


Figure 13-2. Bus Timing (2)

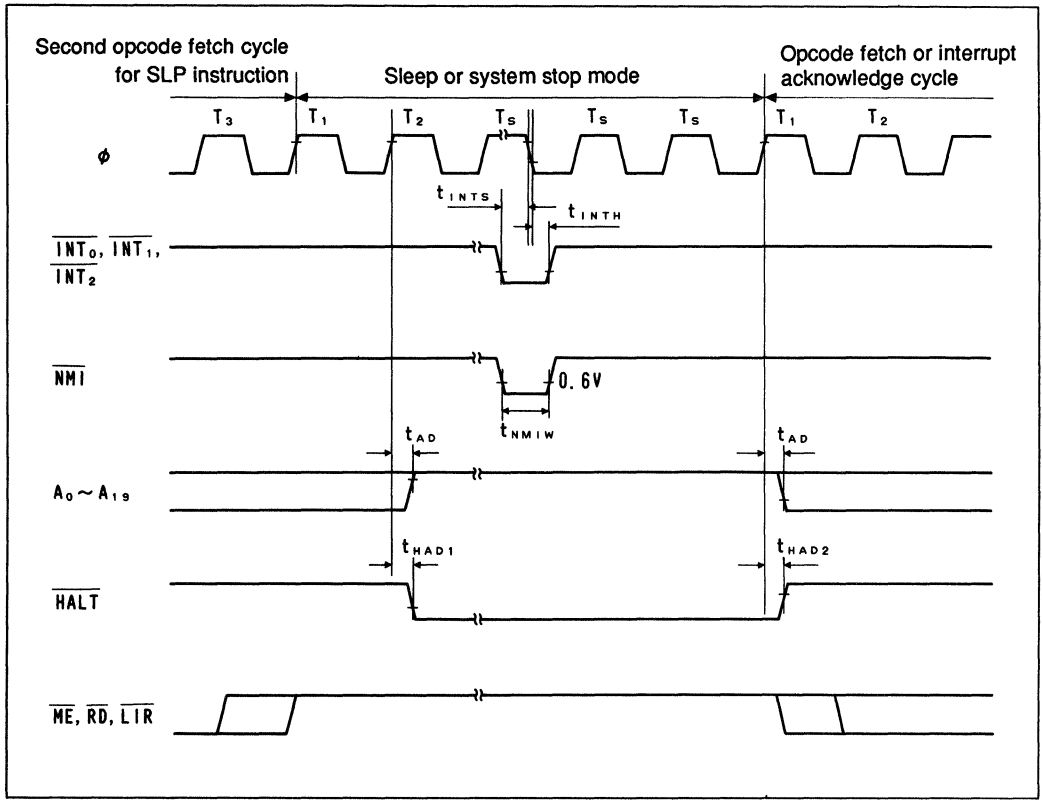


Figure 13-3. Bus Timing (3) (sleep or system stop mode)

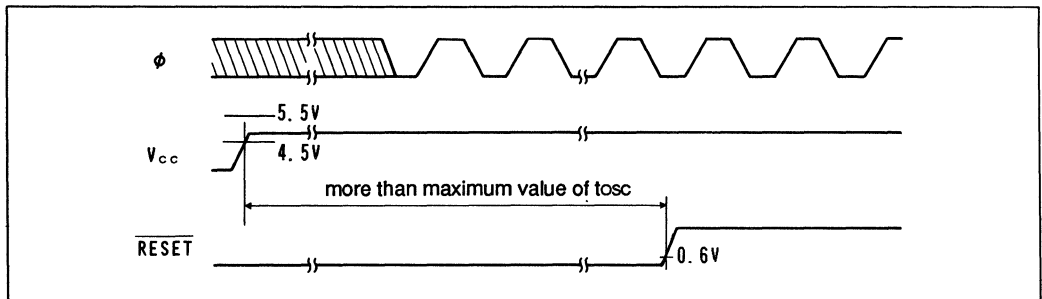


Figure 13-4. Bus Timing (4)

## 13.4.2 MSCI Timing

### (1) Transmit timing (TXCM input)

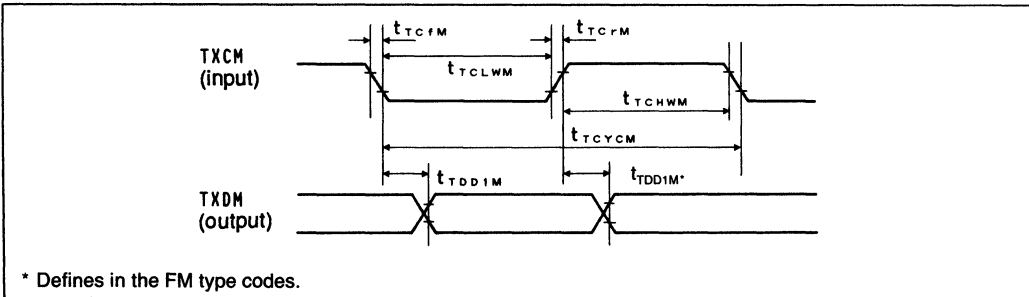


Figure 13-5. Transmit Timing (TXCM input)

### (2) Transmit timing (TXCM output)

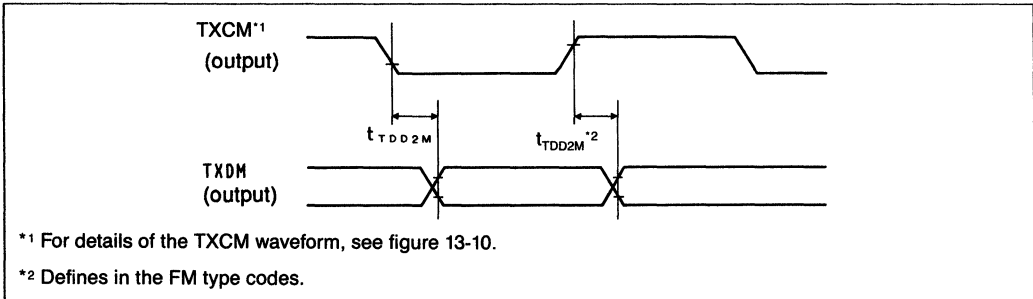


Figure 13-6. Transmit Timing (TXCM output)

### (3) Receive Timing (RXCM input)

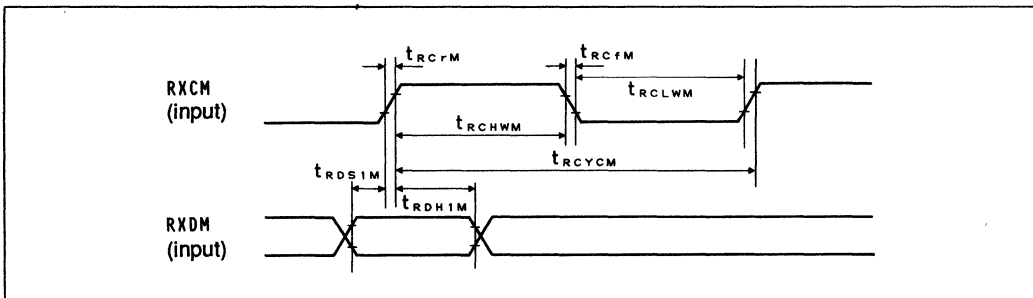


Figure 13-7. Receive Timing (RXCM input)

(4) Receive Timing (RXCM output)

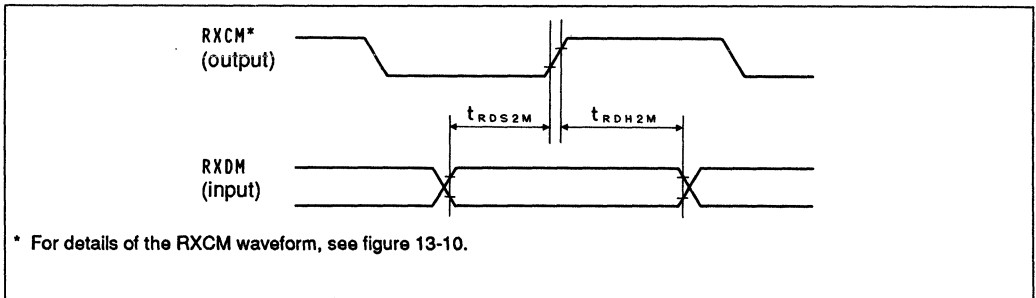


Figure 13-8. Receive Timing (RXCM output)

(5) ADPLL Operating Clock Timing

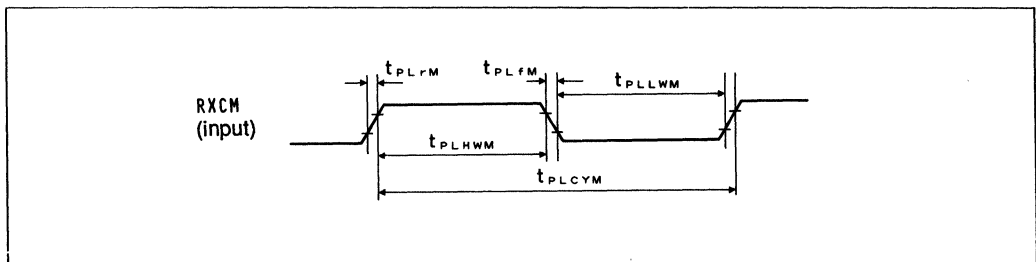


Figure 13-9. ADPLL Operating Clock Timing

(6) Baud Rate Generator Output Timing

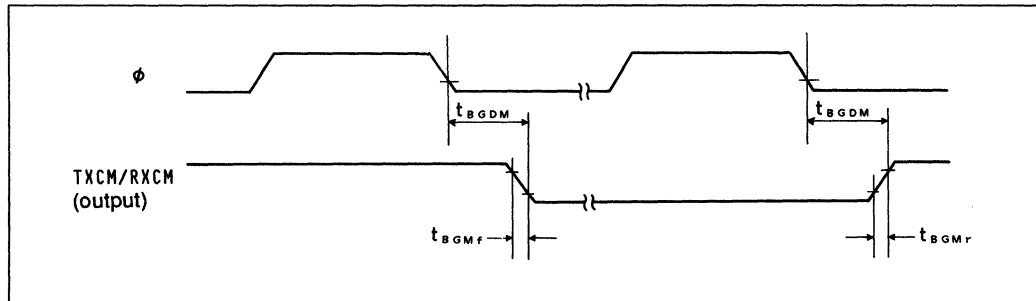


Figure 13-10. Baud Rate Generator Output Timing ( $f_{BRG} \neq f_{\phi}$ )

(7)  $\overline{\text{SYNC}}$  Timing

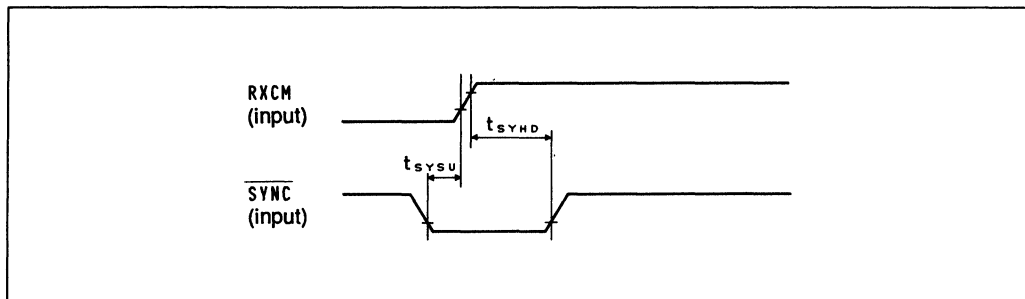


Figure 13-11.  $\overline{\text{SYNC}}$  Timing

(8)  $\overline{\text{CTSM}}$  and  $\overline{\text{DCDM}}$  Timing

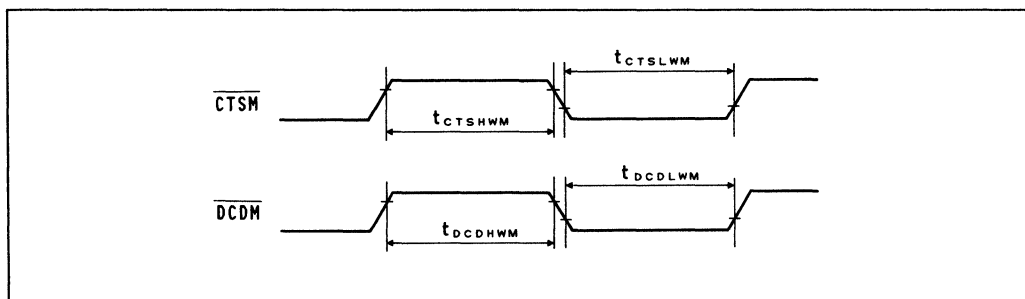


Figure 13-12.  $\overline{\text{CTSM}}$  and  $\overline{\text{DCDM}}$  Timing

(9)  $\overline{\text{RTSM}}$  Timing

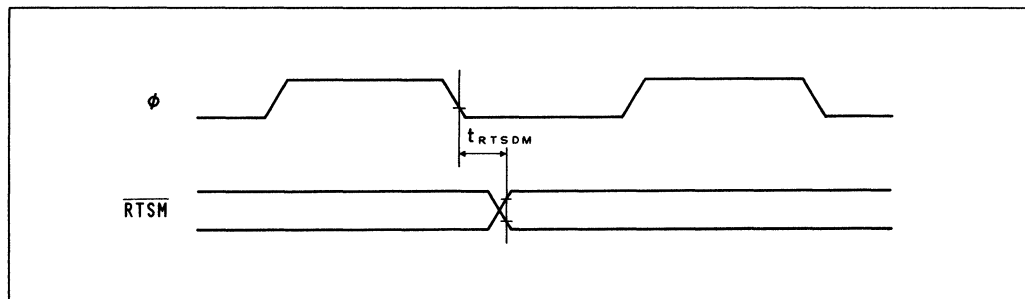


Figure. 13-13.  $\overline{\text{RTSM}}$  Timing

### 13.4.3 ASCI/CSIO Timing

#### (1) Transmit Timing (TXCA input)

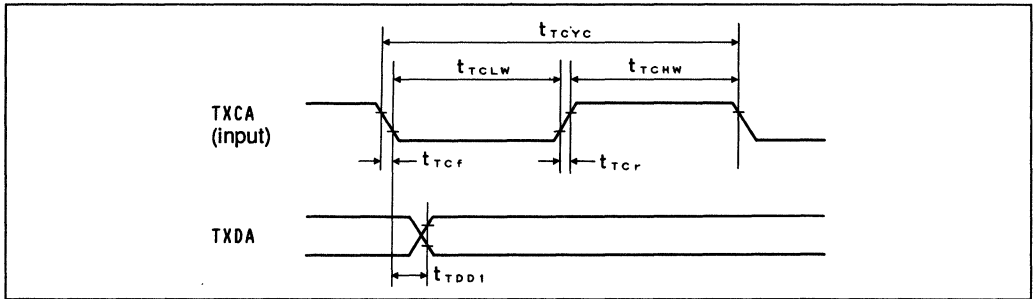


Figure 13-14. Transmit Timing (TXCA input)

#### (2) Transmit Timing (TXCA output)

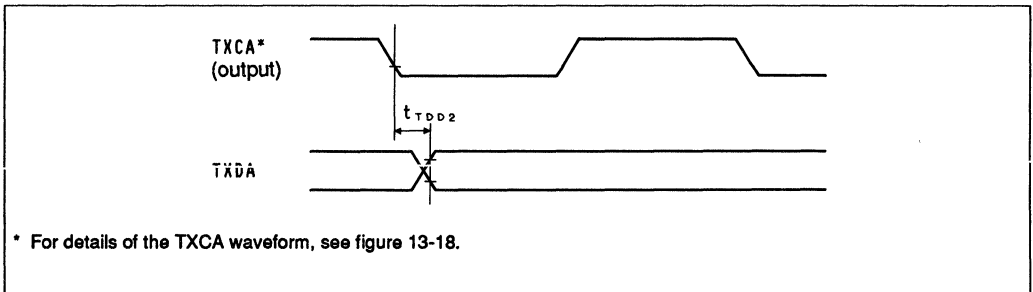


Figure 13-15. Transmit Timing (TXCA output)

#### (3) Receive Timing (RXCA input)

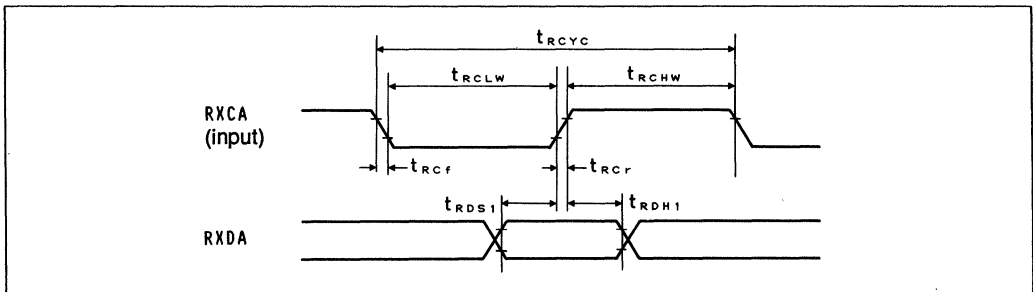


Figure 13-16. Receive Timing (RXCA input)

(4) Receive Timing (RXCA output)

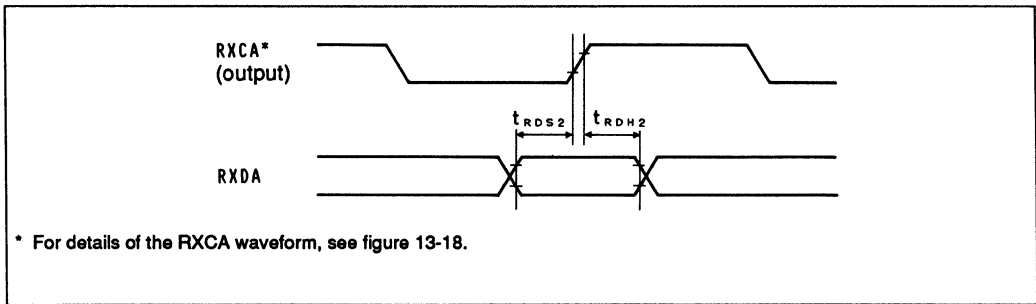


Figure 13-17. Receive Timing (RXCA output)

(5) Baud Rate Generator Timing

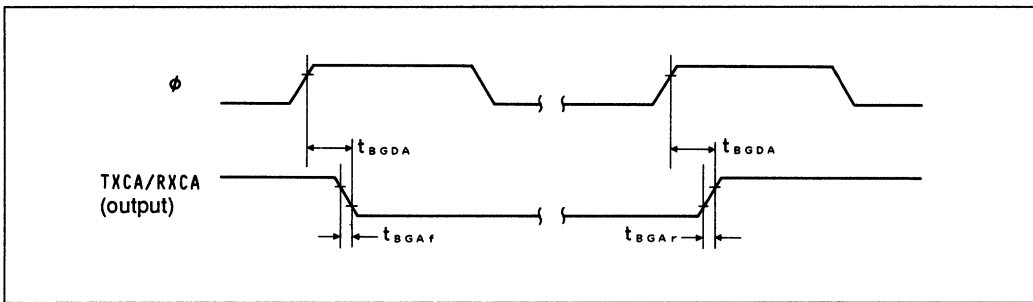


Figure 13-18. Baud Rate Generator Timing

(6)  $\overline{\text{CTSA}}$  and  $\overline{\text{DCDA}}$  Timing

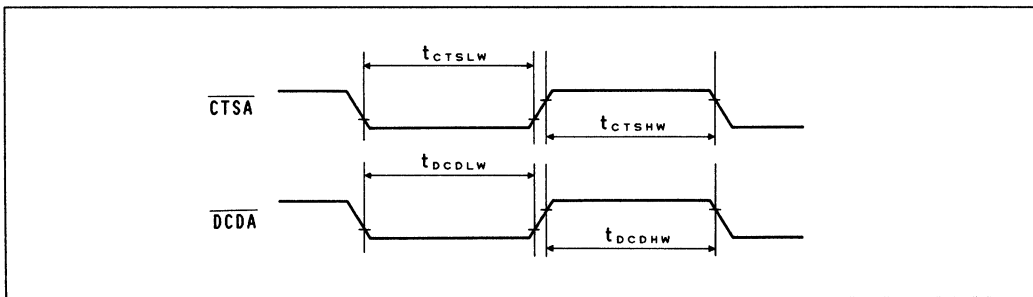


Figure 13-19.  $\overline{\text{CTSA}}$  and  $\overline{\text{DCDA}}$  Timing



(7)  $\overline{\text{RTSA}}$  Timing

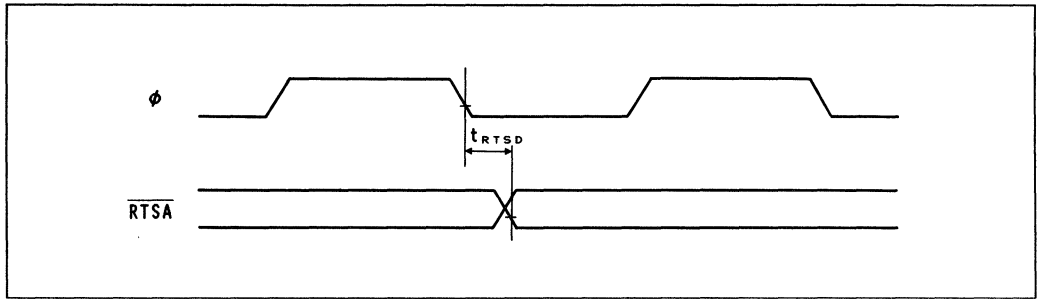


Figure 13-20.  $\overline{\text{RTSA}}$  Timing

### 13.4.4 DMAC Timing

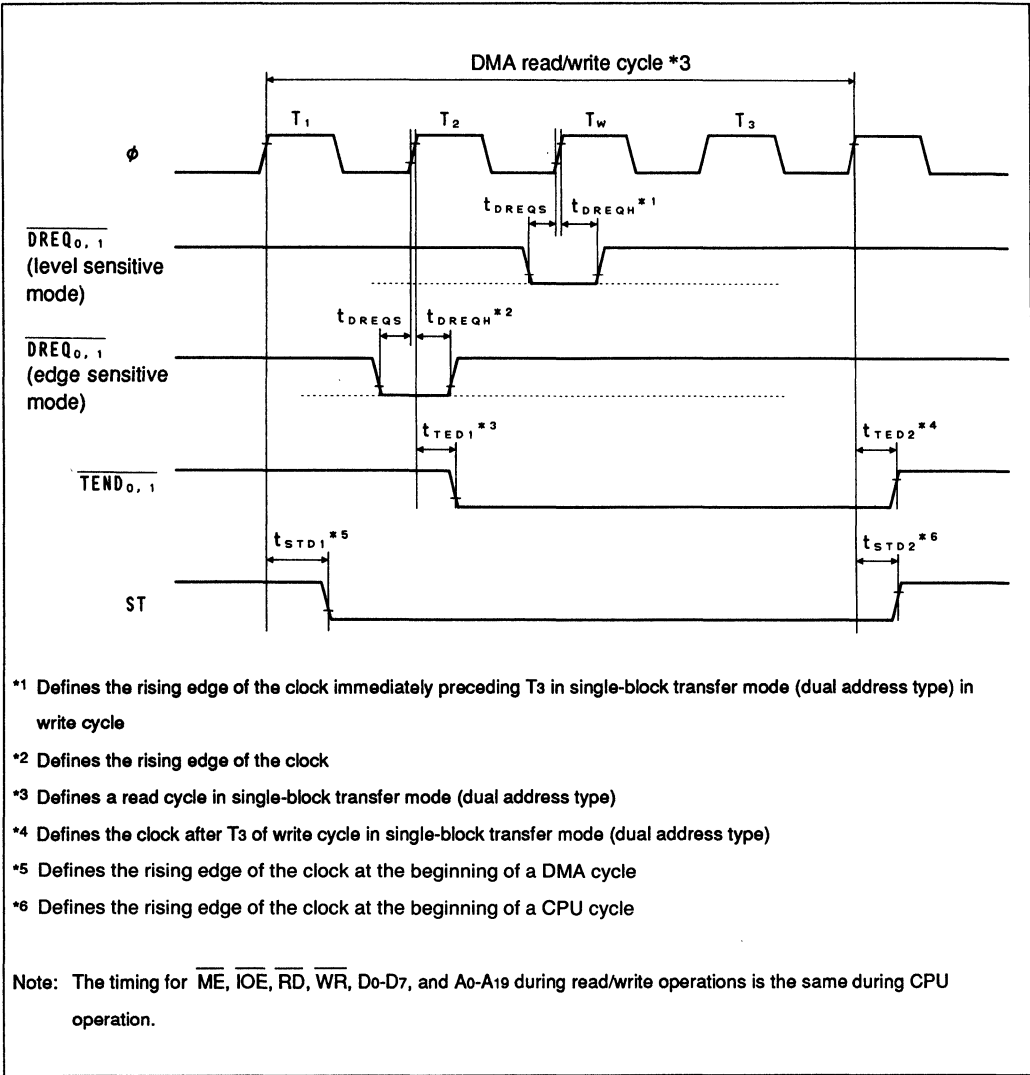


Figure 13-21. DMAC Timing

### 13.4.5 Timer Timing

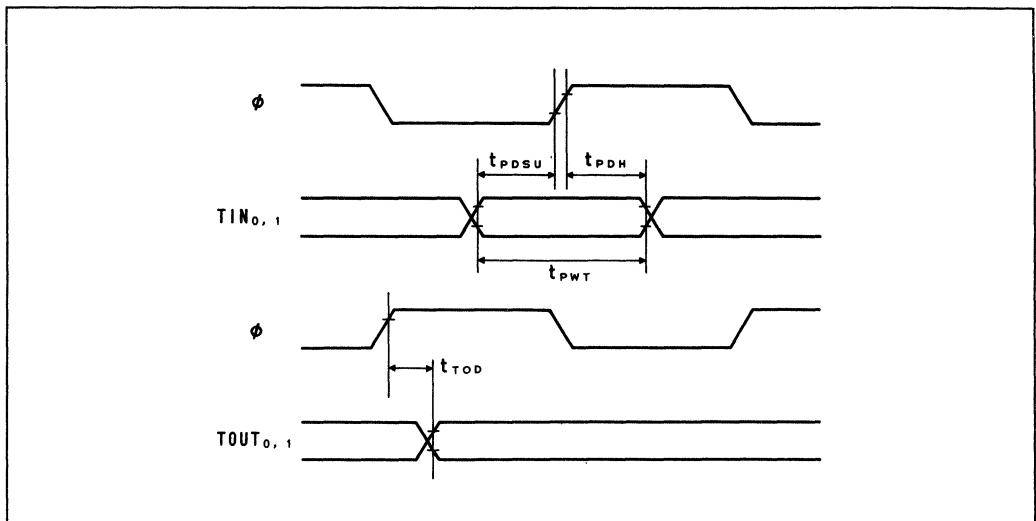


Figure 13-22. Timer Timing

### 13.4.6 EXTAL Input Clock Signal Timing

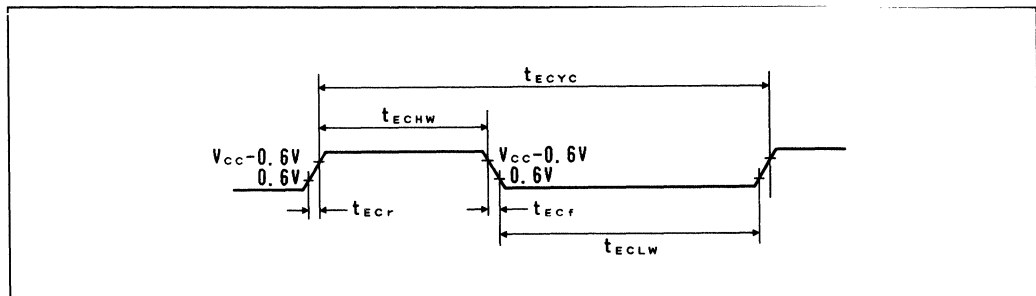


Figure 13-23. EXTAL Input Clock Signal Timing

### 13.4.7 Miscellaneous

#### (1) Rise and Fall Times of Input Signals with No Characteristics Specified

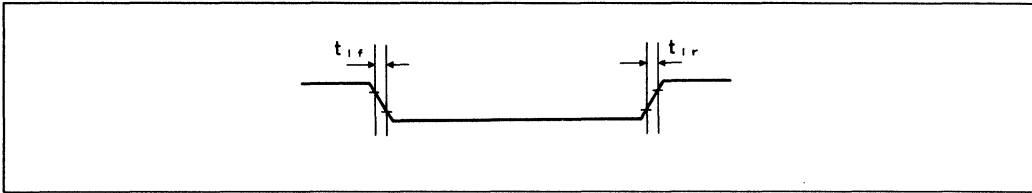


Figure 13-24. Rise and Fall Times of Input Signals with No Characteristics Specified

#### (2) Reference Levels (when not otherwise specified)

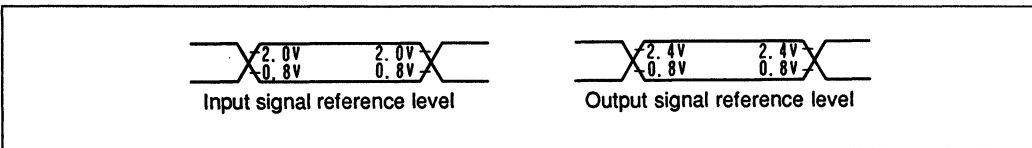


Figure 13-25. Reference Levels

#### (3) Bus Timing Load (TTL load)

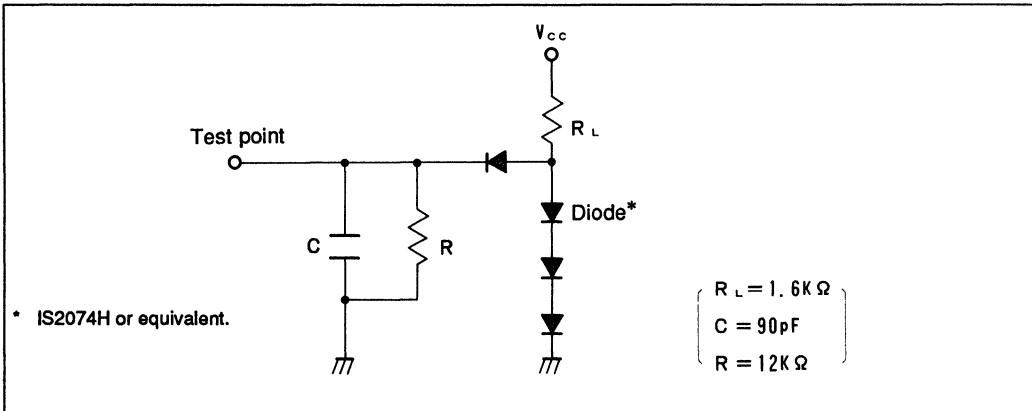
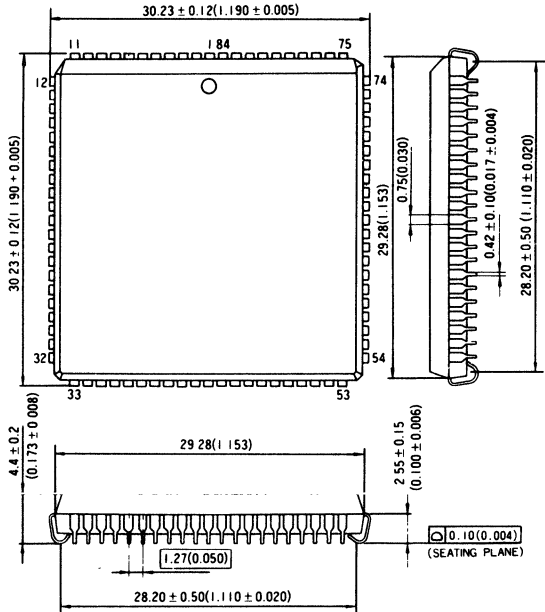


Figure 13-26. Bus Timing Load

# Section 14. Package Dimensions

## 14.1 Package Dimensions

Figures 14-1 A&B shows the external dimensions of the HD64180S.



(CP-84)

Figure 14-1A. A CP-84 Package Dimensions

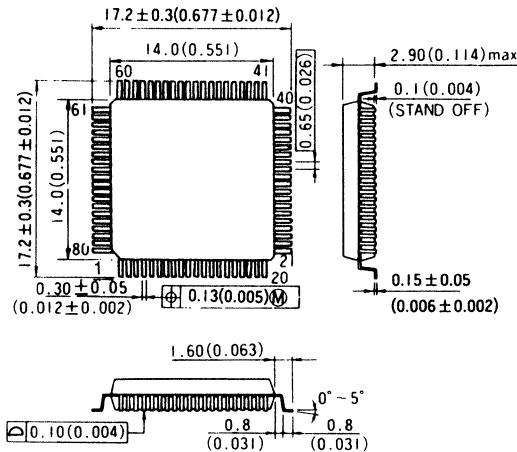
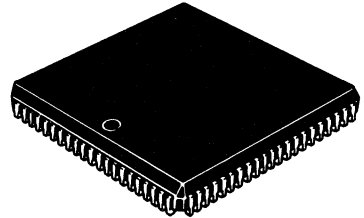
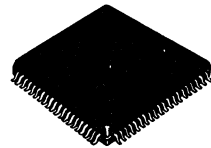


Figure 14-1B. FP-80A Package Dimensions



# Appendices



## A. Instruction Set

In the instruction set, the following conventions are used:

### (1) Register specification

g, g' represents a 8-bit register, while ww, xx, yy, or zz represents a pair of 8-bit registers. The corresponding registers are listed below.

<u>g, g' Register</u>	<u>ww Register</u>	<u>xx Register</u>	<u>yy Register</u>	<u>zz Register</u>
000 B	00 BC	00 BC	00 BC	00 BC
001 C	01 DE	01 DE	01 DE	01 DE
010 D	10 HL	10 IX	10 IY	10 HL
011 E	11 SP	11 SP	11 SP	11 AF
100 H				
101 L				
111 A				

Note: ww, xx, yy, or xx plus H or L (eg, wwH, IXL) indicates the high or low order byte of a 16-bit register.

### (2) Bit specification

'b' indicates the bit number of the bit operand in a bit manipulation instruction. The corresponding bits are listed below.

<u>B</u>	<u>Bit</u>
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7



### (3) Condition specification

'f' indicates the condition for executing an instruction, based on the arithmetic result. The corresponding conditions are listed below.

<b>f</b>	<b>Condition</b>	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign plus
111	M	sign minus

### (4) Restart address

'v' indicates the restart address of a restart instruction. The corresponding addresses are listed below.

<b>v</b>	<b>Address</b>
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

### (5) Flag

Flag changes are indicated by the following symbols:

- : The flag is not changed by the instruction.
- X: Flag change by this instruction is undefined.

- ↓: The flag is changed according to the arithmetic result of the instruction.
- S: The flag is set to 1 by the instruction.
- R: The flag is reset to 0 by the instruction.
- P: The flag is changed as a parity flag by the instruction.
- V: The flag is changed as an overflow flag by the instruction.

(6) Others

- ( )M: Indicates the memory at the address indicated in parentheses.
- ( )I: Indicates the I/O at the address indicated in parentheses.
- m or n: 8-bit value
- mn: 16-bit value
- r: Subscript r indicates a 8-bit register.
- R: Subscript R indicates a 16-bit register.
- b-( )M: Indicates the memory bit specified by b at the address indicated in parentheses.
- b-gr: Indicates the register bit specified by b in the register specified by gr.
- d or j: Signed 8-bit displacement
- S: Source addressing mode
- D: Destination addressing mode
- : AND
- +: OR
- ⊕: Exclusive OR

# 1. Data Manipulation Instructions

## (1) Arithmetic and logic instructions (8bits)

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag							
			IMMED	EXT	IND	REG	REGI	IMP	REL				S	Z	H	P/V	N	C		
ADD	ADD A,g	10 000 g				S					1	4	Ar+gr→Ar	1	1	1	1	V	R	1
	ADD A,(HL)	10 000 110					S				1	6	Ar+(HL) <sub>M</sub> →Ar	1	1	1	1	V	R	1
	ADD A,m	11 000 110 < m >	S								2	6	Ar+m→Ar	1	1	1	1	V	R	1
	ADD A,(IX+d)	11 011 101 10 000 110 < d >			S						3	14	Ar+(IX+d) <sub>M</sub> →Ar	1	1	1	1	V	R	1
	ADD A,(IY+d)	11 111 101 10 000 110 < d >			S						3	14	Ar+(IY+d) <sub>M</sub> →Ar	1	1	1	1	V	R	1
ADC	ADC A,g	10 001 g				S					1	4	Ar+gr+c→Ar	1	1	1	1	V	R	1
	ADC A,(HL)	10 001 110					S				1	6	Ar+(HL) <sub>M</sub> +c→Ar	1	1	1	1	V	R	1
	ADC A,m	11 001 110 < m >	S								2	6	Ar+m+c→Ar	1	1	1	1	V	R	1
	ADC A,(IX+d)	11 011 101 10 001 110 < d >			S						3	14	Ar+(IX+d) <sub>M</sub> +c→Ar	1	1	1	1	V	R	1
	ADC A,(IY+d)	11 111 101 10 001 110 < d >			S						3	14	Ar+(IY+d) <sub>M</sub> +c→Ar	1	1	1	1	V	R	1
AND	AND g	10 100 g				S					1	4	Ar-gr→Ar	1	1	S	P	R	R	R
	AND (HL)	10 100 110					S				1	6	Ar-(HL) <sub>M</sub> →Ar	1	1	S	P	R	R	R
	AND m	11 100 110 < m >	S								2	6	Ar-m→Ar	1	1	S	P	R	R	R
	AND (IX+d)	11 011 101 10 100 110 < d >			S						3	14	Ar-(IX+d) <sub>M</sub> →Ar	1	1	S	P	R	R	R
	AND (IY+d)	11 111 101 10 100 110 < d >			S						3	14	Ar-(IY+d) <sub>M</sub> →Ar	1	1	S	P	R	R	R
Compare	CP g	10 111 g				S					1	4	Ar-gr	1	1	1	1	V	S	1
	CP (HL)	10 111 110					S				1	6	Ar-(HL) <sub>M</sub>	1	1	1	1	V	S	1
	CP m	11 111 110 < m >	S								2	6	Ar-m	1	1	1	1	V	S	1
	CP (IX+d)	11 011 101 10 111 110 < d >			S						3	14	Ar-(IX+d) <sub>M</sub>	1	1	1	1	V	S	1
	CP (IY+d)	11 111 101 10 111 110 < d >			S						3	14	Ar-(IY+d) <sub>M</sub>	1	1	1	1	V	S	1
COMPLEMENT	CPL	00 101 111							S/D	1	3	Ar→Ar	.	.	S	.	S	.	.	
DEC	DEC g	00 g 101				S/D					1	4	gr-1→gr	1	1	1	1	V	S	.
	DEC (HL)	00 110 101					S/D				1	10	(HL) <sub>M</sub> -1→(HL) <sub>M</sub>	1	1	1	1	V	S	.
	DEC (IX+d)	11 011 101 00 110 101 < d >			S/D						3	18	(IX+d) <sub>M</sub> -1→ (IX+d) <sub>M</sub>	1	1	1	1	V	S	.
	DEC (IY+d)	11 111 101 00 110 101 < d >			S/D						3	18	(IY+d) <sub>M</sub> -1→ (IY+d) <sub>M</sub>	1	1	1	1	V	S	.
INC	INC g	00 g 100				S/D					1	4	gr+1→gr	1	1	1	1	V	R	.
	INC (HL)	00 110 100					S/D				1	10	(HL) <sub>M</sub> +1→(HL) <sub>M</sub>	1	1	1	1	V	R	.
	INC (IX+d)	11 011 101 00 110 100 < d >			S/D						3	18	(IX+d) <sub>M</sub> +1→ (IX+d) <sub>M</sub>	1	1	1	1	V	R	.
	INC (IY+d)	11 111 101 00 110 100 < d >			S/D						3	18	(IY+d) <sub>M</sub> +1→ (IY+d) <sub>M</sub>	1	1	1	1	V	R	.

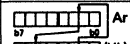
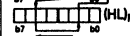
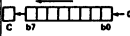
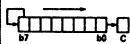
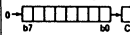
(Continued)

Operation name	MNEMONICS	OP code	Addressing								Bytes	States	Operation	Flag					
			IMMED	EXT	IND	REG	REGI	IMP	REL	7				6	4	2	1	0	
																			S
MULT	MLT ww	11 101 101 01 ww1 100				S/D					2	17	wwHxwwLr-wwx	.	.	.	.	.	.
NEGATE	NEG	11 101 101 01 000 100							S/D		2	6	0-Ar→Ar	1	1	1	V	S	1
OR	OR g	10 110 g	S			S		D		1	4	Ar-gr→Ar	1	1	R	P	R	R	
	OR (HL)	10 110 110				S	D	1	6	Ar+(HL) <sub>w</sub> →Ar	1	1	R	P	R	R			
	OR m	11 110 110 < m >					D	2	6	Ar+m→Ar	1	1	R	P	R	R			
	OR (IX+d)	11 011 101 10 110 110 < d >				S		D	3	14	Ar+(IX+d) <sub>w</sub> →Ar	1	1	R	P	R	R		
OR (IY+d)	11 111 101 10 110 110 < d >		S		D	3	14	Ar+(IY+d) <sub>w</sub> →Ar	1	1	R	P	R	R					
SUB	SUB g	10 010 g	S			S		D		1	4	Ar-gr→Ar	1	1	V	S	1		
	SUB (HL)	10 010 110				S	D	1	6	Ar-(HL) <sub>w</sub> →Ar	1	1	V	S	1				
	SUB m	11 010 110 < m >					D	2	6	Ar-m→Ar	1	1	V	S	1				
	SUB (IX+d)	11 011 101 10 010 110 < d >				S		D	3	14	Ar-(IX+d) <sub>w</sub> →Ar	1	1	V	S	1			
SUB (IY+d)	11 111 101 10 010 110 < d >		S		D	3	14	Ar-(IY+d) <sub>w</sub> →Ar	1	1	V	S	1						
SUBC	SBC A <sub>g</sub>	10 011 g	S			S		D		1	4	Ar-gr-c→Ar	1	1	V	S	1		
	SBC A <sub>(HL)</sub>	10 011 110				S	D	1	6	Ar-(HL) <sub>w</sub> -c→Ar	1	1	V	S	1				
	SBC A <sub>m</sub>	11 011 110 < m >					D	2	6	Ar-m-c→Ar	1	1	V	S	1				
	SBC A <sub>(IX+d)</sub>	11 011 101 10 011 110 < d >				S		D	3	14	Ar-(IX+d) <sub>w</sub> -c→Ar	1	1	V	S	1			
SBC A <sub>(IY+d)</sub>	11 111 101 10 011 110 < d >		S		D	3	14	Ar-(IY+d) <sub>w</sub> -c→Ar	1	1	V	S	1						
TEST	TST g	11 101 101 00 g 100	S			S				2	7	Ar-gr	1	1	S	P	R	R	
	TST (HL)	11 101 101 00 110 100				S		2	10	Ar-(HL) <sub>w</sub>	1	1	S	P	R	R			
	TST m	11 101 101 01 100 100 < m >						3	9	Ar-m	1	1	S	P	R	R			
XOR	XOR g	10 101 g	S			S		D		1	4	Ar⊕gr→Ar	1	1	R	P	R	R	
	XOR (HL)	10 101 110				S	D	1	6	Ar⊕(HL) <sub>w</sub> →Ar	1	1	R	P	R	R			
	XOR m	11 101 110 < m >					D	2	6	Ar⊕m→Ar	1	1	R	P	R	R			
	XOR (IX+d)	11 011 101 10 101 110 < d >				S		D	3	14	Ar⊕(IX+d) <sub>w</sub> →Ar	1	1	R	P	R	R		
XOR (IY+d)	11 111 101 10 101 110 < d >		S		D	3	14	Ar⊕(IY+d) <sub>w</sub> →Ar	1	1	R	P	R	R					

## (2) Rotate/shift instructions

Operation name	MNEMONICS	OP code	Addressing						Bytes	States	Operation	Flag							
			IMMED	EXT	IND	REG	REGI	IMP				REL	7	6	4	2	1	0	
													S	Z	H	P/V	N	C	
Rotate and Shift Data	RLA	00 010 111						S/D		1	3		.	.	R	.	R	R	I
	RL g	11 001 011						S/D		2	7		1	1	R	P	R	R	I
	RL (HL)	00 010 g								2	13		1	1	R	P	R	R	I
	RL (IX+d)	00 010 110								4	19		1	1	R	P	R	R	I
		11 011 101						S/D					1	1	R	P	R	R	I
		11 001 011																	
		< d >																	
		00 010 110																	
	RL (IY+d)	11 111 101						S/D		4	19		1	1	R	P	R	R	I
		11 001 011																	
		< d >																	
		00 010 110																	
	RLCA	00 000 111								1	3		.	.	R	.	R	R	I
	RLC g	11 001 011						S/D		2	7		1	1	R	P	R	R	I
		00 000 g											1	1	R	P	R	R	I
	RLC (HL)	11 001 011								2	13		1	1	R	P	R	R	I
		00 000 110											1	1	R	P	R	R	I
	RLC (IX+d)	11 011 101						S/D		4	19		1	1	R	P	R	R	I
		11 001 011											1	1	R	P	R	R	I
		< d >																	
		00 000 110											1	1	R	P	R	R	I
	RLC (IY+d)	11 111 101						S/D		4	19		1	1	R	P	R	R	I
		11 001 011																	
		< d >																	
		00 000 110																	
	RLD	00 000 110								2	16		1	1	R	P	R	R	I
		11 101 101											1	1	R	P	R	R	I
		01 101 111																	
	RRA	00 011 111								1	3		.	.	R	.	R	R	I
	RR g	11 001 011						S/D		2	7		1	1	R	P	R	R	I
		00 011 g											1	1	R	P	R	R	I
	RR (HL)	11 001 011								2	13		1	1	R	P	R	R	I
		00 011 110											1	1	R	P	R	R	I
	RR (IX+d)	11 011 101						S/D		4	19		1	1	R	P	R	R	I
		11 001 011											1	1	R	P	R	R	I
		< d >																	
		00 011 110											1	1	R	P	R	R	I
	RR (IY+d)	11 111 101						S/D		4	19		1	1	R	P	R	R	I
		11 001 011																	
		< d >																	
		00 011 110																	
	RRCA	00 001 111								1	3		.	.	R	.	R	R	I
	RRC g	11 001 011						S/D		2	7		1	1	R	P	R	R	I
		00 001 g											1	1	R	P	R	R	I
	RRC (HL)	11 001 011								2	13		1	1	R	P	R	R	I
		00 001 110											1	1	R	P	R	R	I
	RRC (IX+d)	11 011 101						S/D		4	19		1	1	R	P	R	R	I
		11 001 011											1	1	R	P	R	R	I
		< d >																	
		00 001 110											1	1	R	P	R	R	I
	RRC (IY+d)	11 111 101						S/D		4	19		1	1	R	P	R	R	I
		11 001 011																	
		< d >																	
		00 001 110																	

(Continued)

Operation name	MNEMONICS	OP code	Addressing						Bytes	States	Operation	Flag						
			IMMED	EXT	IND	REG	REGI	IMP				REL	7	6	4	2	1	0
													S	Z	H	P/V	N	C
Rotate and Shift Data	RRD	11 101 101 01 100 111						S/D		2	16		1	1	R	P	R	·
	SLA g	11 001 011 00 100 g					S/D			2	7		1	1	R	P	R	1
	SLA (HL)	11 001 011 00 100 110					S/D			2	13		1	1	R	P	R	1
	SLA (IX+d)	11 011 101 11 001 011 < d >			S/D					4	19		1	1	R	P	R	1
	SLA (IY+d)	00 100 110 11 111 101 11 001 011 < d >			S/D					4	19		1	1	R	P	R	1
	SRA g	00 100 110 11 001 011 00 101 g					S/D			2	7		1	1	R	P	R	1
	SRA (HL)	11 001 011 00 101 110					S/D			2	13		1	1	R	P	R	1
	SRA (IX+d)	11 011 101 11 001 011 < d >			S/D					4	19		1	1	R	P	R	1
	SRA (IY+d)	00 101 110 11 111 101 11 001 011 < d >			S/D					4	19		1	1	R	P	R	1
	SRL g	00 101 110 11 001 011 00 111 g					S/D			2	7		1	1	R	P	R	1
	SRL (HL)	11 001 011 00 111 110					S/D			2	3		1	1	R	P	R	1
	SRL (IX+d)	11 011 101 11 001 011 < d >			S/D					4	19		1	1	R	P	R	1
	SRL (IY+d)	00 111 110 11 111 101 11 001 011 < d >			S/D					4	19		1	1	R	P	R	1

### (3) Bit manipulation instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	IND	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
Bit Set	SET b <sub>g</sub>	11 001 011 11 b g				S/D				2	7	1→b <sub>g</sub> r	.	.	.	.	.	.
	SET b <sub>g</sub> (HL)	11 001 011 11 b 110					S/D			2	13	1→b <sub>g</sub> (HL) <sub>H</sub>	.	.	.	.	.	.
	SET b <sub>g</sub> (IX+d)	11 011 101 11 001 011 < d >			S/D					4	19	1→b <sub>g</sub> (IX+d) <sub>H</sub>	.	.	.	.	.	.
	SET b <sub>g</sub> (IY+d)	11 b 110 11 111 101 11 001 011 < d >			S/D					4	19	1→b <sub>g</sub> (IY+d) <sub>H</sub>	.	.	.	.	.	.
Bit Reset	RES b <sub>g</sub>	11 001 011 10 b g				S/D				2	7	0→b <sub>g</sub> r	.	.	.	.	.	.
	RES b <sub>g</sub> (HL)	11 001 011 10 b 110					S/D			2	13	0→b <sub>g</sub> (HL) <sub>H</sub>	.	.	.	.	.	.
	RES b <sub>g</sub> (IX+d)	11 011 101 11 001 011 < d >			S/D					4	19	0→b <sub>g</sub> (IX+d) <sub>H</sub>	.	.	.	.	.	.
	RES b <sub>g</sub> (IY+d)	10 b 110 11 111 101 11 001 011 < d >			S/D					4	19	0→b <sub>g</sub> (IY+d) <sub>H</sub>	.	.	.	.	.	.
Bit Test	BIT b <sub>g</sub>	11 001 011 01 b g				S				2	6	b <sub>g</sub> r→z	X	I	S	X	R	.
	BIT b <sub>g</sub> (HL)	11 001 011 01 b 110					S			2	9	b <sub>g</sub> (HL) <sub>H</sub> →z	X	I	S	X	R	.
	BIT b <sub>g</sub> (IX+d)	11 011 101 11 001 011 < d >			S					4	15	b <sub>g</sub> (IX+d) <sub>H</sub> →z	X	I	S	X	R	.
	BIT b <sub>g</sub> (IY+d)	01 b 110 11 111 101 11 001 011 < d >			S					4	15	b <sub>g</sub> (IY+d) <sub>H</sub> →z	X	I	S	X	R	.

(4) Arithmetic instructions (16 bits)

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	IND	REG	REGI	IMP	REL				7	6	4	2	1	0
													S	Z	H	P/V	N	C
ADD	ADD HL,ww	00 ww1 001				S		D		1	7	$HL_n + ww_n \rightarrow HL_n$	.	.	X	.	R	1
	ADD IX,xx	11 011 101 00 xx1 001				S		D		2	10	$IX_n + xx_n \rightarrow IX_n$	.	.	X	.	R	1
	ADD IY,yy	11 111 101 00 yy1 001				S		D		2	10	$IY_n + yy_n \rightarrow IY_n$	.	.	X	.	R	1
ADC	ADC HL,ww	11 101 101 01 ww1 010				S		D		2	10	$HL_n + ww_n + c \rightarrow HL_n$	1	1	X	V	R	1
DEC	DEC ww	00 ww1 011				S/D				1	4	$ww_n - 1 \rightarrow ww_n$	.	.	.	.	.	.
	DEC IX	11 011 101 00 101 011						S/D		2	7	$IX_n - 1 \rightarrow IX_n$	.	.	.	.	.	.
	DEC IY	11 111 101 00 101 011						S/D		2	7	$IY_n - 1 \rightarrow IY_n$	.	.	.	.	.	.
INC	INC ww	00 ww0 011				S/D				1	4	$ww_n + 1 \rightarrow ww_n$	.	.	.	.	.	.
	INC IX	11 011 101 00 100 011						S/D		2	7	$IX_n + 1 \rightarrow IX_n$	.	.	.	.	.	.
	INC IY	11 111 101 00 100 011						S/D		2	7	$IY_n + 1 \rightarrow IY_n$	.	.	.	.	.	.
SBC	SBC HL,ww	11 101 101 01 ww0 010				S		D		2	10	$HL_n - ww_n - c \rightarrow HL_n$	1	1	X	V	S	1



## 2. Data Transfer Instructions

### (1) 8-bit transfer instructions

Operation name	MNEMONICS	OP code	Addressing						Bytes	States	Operation	Flag								
			IMMED	EXT	IND	REG	REGI	IMP				REL	7	6	4	2	1	0		
													S	Z	H	P/V	N	C		
Load 8-bit Data	LD A,I	11 101 101 01 010 111						S/D		2	6	Ir→Ar	*1	I	I	R	IEF <sub>2</sub>	R	.	
	LD A,R	11 101 101 01 011 111						S/D		2	6	Rr→Ar	*1	I	I	R	IEF <sub>2</sub>	R	.	
	LD A,(BC)	00 001 010					S	D		1	6	(BC) <sub>m</sub> →Ar		.	.	.	.	.	.	.
	LD A,(DE)	00 011 010					S	D		1	6	(DE) <sub>m</sub> →Ar		.	.	.	.	.	.	.
	LD A,(mn)	00 111 010 < n > < m >		S				D		3	12	(mn) <sub>m</sub> →Ar		.	.	.	.	.	.	.
	LD I,A	11 101 101 01 000 111						S/D		2	6	Ar→Ir		.	.	.	.	.	.	.
	LD R,A	11 101 101 01 001 111						S/D		2	6	Ar→Rr		.	.	.	.	.	.	.
	LD (BC),A	00 000 010						D	S		1	7	Ar→(BC) <sub>m</sub>		.	.	.	.	.	.
	LD (DE),A	00 010 010						D	S		1	7	Ar→(DE) <sub>m</sub>		.	.	.	.	.	.
	LD (mn),A	00 110 010 < n > < m >		D					S		3	13	Ar→(mn) <sub>m</sub>		.	.	.	.	.	.
	LD g,g'	01 g g'					S/D				1	4	gr'→gr		.	.	.	.	.	.
	LD g,(HL)	01 g 110					D	S			1	6	(HL) <sub>m</sub> →gr		.	.	.	.	.	.
	LD g,m	00 g 110 < m >	S								2	6	m→gr		.	.	.	.	.	.
	LD g,(IX+d)	11 011 101 01 g 110 < d >			S	D					3	14	(IX+d) <sub>m</sub> →gr		.	.	.	.	.	.
	LD g,(IY+d)	11 111 101 01 g 110 < d >			S	D					3	14	(IY+d) <sub>m</sub> →gr		.	.	.	.	.	.
	LD (HL),m	00 110 110 < m >	S					D			2	9	m→(HL) <sub>m</sub>		.	.	.	.	.	.
	LD (IX+d),m	11 011 101 00 110 110 < d > < m >	S		D						4	15	m→(IX+d) <sub>m</sub>		.	.	.	.	.	.
	LD (IY+d),m	11 111 101 00 110 110 < d > < m >	S		D						4	15	m→(IY+d) <sub>m</sub>		.	.	.	.	.	.
	LD (HL),g	01 110 g				S	D				1	7	gr→(HL) <sub>m</sub>		.	.	.	.	.	.
	LD (IX+d),g	11 011 101 01 110 g < d >			D	S					3	15	gr→(IX+d) <sub>m</sub>		.	.	.	.	.	.
	LD (IY+d),g	11 111 101 01 110 g < d >			D	S					3	15	gr→(IY+d) <sub>m</sub>		.	.	.	.	.	.

\*1 No interrupts are sampled at the end of LD A,I or LD A,R instruction.

## (2) 16-bit transfer instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag							
			IMMED	EXT	IND	REG	REGI	IMP	REL				7	6	4	2	1	0		
													S	Z	H	P/V	N	C		
Load 16-bit Data	LD ww,mn	00 ww0 001 < n > < m >	S			D					3	9	mn→ww <sub>s</sub>	.	.	.	.	.	.	
	LD IX,mn	11 011 101 00 100 001 < n > < m >	S						D		4	12	mn→IX <sub>s</sub>	.	.	.	.	.	.	
	LD IY,mn	11 111 101 00 100 001 < n > < m >	S						D		4	12	mn→IY <sub>s</sub>	.	.	.	.	.	.	
	LD SP,HL	11 111 001							S/D		1	4	HL <sub>s</sub> →SP <sub>s</sub>	.	.	.	.	.	.	
	LD SP,IX	11 011 101							S/D		2	7	IX <sub>s</sub> →SP <sub>s</sub>	.	.	.	.	.	.	
	LD SP,IY	11 111 001							S/D		2	7	IY <sub>s</sub> →SP <sub>s</sub>	.	.	.	.	.	.	
	LD ww,(mn)	11 101 101 01 ww1 011 < n > < m >		S		D						4	18	(mn+1) <sub>w</sub> →wwHr (mn) <sub>w</sub> →wwLr	.	.	.	.	.	.
	LD HL,(mn)	00 101 010 < n > < m >		S					D			3	15	(mn+1) <sub>w</sub> →HLr (mn) <sub>w</sub> →Lr	.	.	.	.	.	.
	LD IX,(mn)	11 011 101 00 101 010 < n > < m >		S						D		4	18	(mn+1) <sub>w</sub> →IXHr (mn) <sub>w</sub> →IXLr	.	.	.	.	.	.
	LD IY,(mn)	11 111 101 00 101 010 < n > < m >		S						D		4	18	(mn+1) <sub>w</sub> →IYHr (mn) <sub>w</sub> →IYLr	.	.	.	.	.	.
	LD (mn),ww	11 101 101 01 ww0 011 < n > < m >			D		S					4	19	wwHr→(mn+1) <sub>w</sub> wwLr→(mn) <sub>w</sub>	.	.	.	.	.	.
	LD (mn),HL	00 100 010 < n > < m >			D					S		3	16	Hr→(mn+1) <sub>w</sub> Lr→(mn) <sub>w</sub>	.	.	.	.	.	.
	LD (mn),IX	11 011 101 00 100 010 < n > < m >			D					S		4	19	IXHr→(mn+1) <sub>w</sub> IXLr→(mn) <sub>w</sub>	.	.	.	.	.	.
	LD (mn),IY	11 111 101 00 100 010 < n > < m >			D					S		4	19	IYHr→(mn+1) <sub>w</sub> IYLr→(mn) <sub>w</sub>	.	.	.	.	.	.

### (3) Block transfer instructions

Operation name	MNEMONICS	OP code	Addressing						Bytes	States	Operation	Flag							
			IMMED	EXT	IND	REG	REGI	IMP				REL	7	6	4	2	1	0	
													S	Z	H	P/V	N	C	
Block Transfer Search Data	CPD	11 101 101 10 101 001					S	S	2	12	Ar-(HL) <sub>M</sub> BC <sub>R</sub> -1→BC <sub>R</sub> HL <sub>R</sub> +1→HL <sub>R</sub>	*3	*2	1	1	1	1	S	.
	CPDR	11 101 101 10 111 001					S	S	2	14 12	BC <sub>R</sub> ≠0 Ar≠(HL) <sub>M</sub> BC <sub>R</sub> =0 or Ar=(HL) <sub>M</sub> (Ar-(HL) <sub>M</sub> ) Q BC <sub>R</sub> -1→BC <sub>R</sub> HL <sub>R</sub> +1→HL <sub>R</sub> Repeat Q until Ar=(HL) <sub>M</sub> or BC <sub>R</sub> =0	*3	*2	1	1	1	1	S	.
	CPI	11 101 101 10 100 001					S	S	2	12	Ar-(HL) <sub>M</sub> BC <sub>R</sub> -1→BC <sub>R</sub> HL <sub>R</sub> +1→HL <sub>R</sub>	*3	*2	1	1	1	1	S	.
	CPIR	11 101 101 10 110 001					S	S	2	14 12	BC <sub>R</sub> ≠0 Ar≠(HL) <sub>M</sub> BC <sub>R</sub> =0 or Ar=(HL) <sub>M</sub> (Ar-(HL) <sub>M</sub> ) Q BC <sub>R</sub> -1→BC <sub>R</sub> HL <sub>R</sub> +1→HL <sub>R</sub> Repeat Q until Ar=(HL) <sub>M</sub> or BC <sub>R</sub> =0	*3	*2	1	1	1	1	S	.
	LDD	11 101 101 10 101 000					S/D		2	12	(HL) <sub>M</sub> →(DE) <sub>M</sub> (HL) <sub>M</sub> →(DE) <sub>M</sub> BC <sub>R</sub> -1→BC <sub>R</sub> DE <sub>R</sub> +1→DE <sub>R</sub> HL <sub>R</sub> +1→HL <sub>R</sub>		*2	.	.	R	I	R	.
	LDDR	11 101 101 10 111 000					S/D		2	14 (BC <sub>R</sub> ≠0) 12 (BC <sub>R</sub> =0)	(HL) <sub>M</sub> →(DE) <sub>M</sub> BC <sub>R</sub> -1→BC <sub>R</sub> Q DE <sub>R</sub> +1→DE <sub>R</sub> HL <sub>R</sub> +1→HL <sub>R</sub> Repeat Q until BC <sub>R</sub> =0			.	.	R	R	R	.
	LDI	11 101 101 10 100 000					S/D		2	12	(HL) <sub>M</sub> →(DE) <sub>M</sub> (HL) <sub>M</sub> →(DE) <sub>M</sub> BC <sub>R</sub> -1→BC <sub>R</sub> DE <sub>R</sub> +1→DE <sub>R</sub> HL <sub>R</sub> +1→HL <sub>R</sub>		*2	.	.	R	I	R	.
	LDIR	11 101 101 10 110 000					S/D		2	14 (BC <sub>R</sub> ≠0) 12 (BC <sub>R</sub> =0)	(HL) <sub>M</sub> →(DE) <sub>M</sub> BC <sub>R</sub> -1→BC <sub>R</sub> Q DE <sub>R</sub> +1→DE <sub>R</sub> HL <sub>R</sub> +1→HL <sub>R</sub> Repeat Q until BC <sub>R</sub> =0			.	.	R	R	R	.

- \*2 P/V = 0 : BC<sub>R</sub> - 1 = 0  
P/V = 1 : BC<sub>R</sub> - 1 ≠ 0  
\*3 Z = 1 : Ar = (HL)<sub>M</sub>  
Z = 0 : Ar ≠ (HL)<sub>M</sub>

#### (4) Stack/exchange instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	IND	REG	REGI	IMP	REL				7	6	4	2	1	0
						S		D					S	Z	H	P/V	N	C
PUSH	PUSH zz	11 zz0 101				S		D		1	11	zzLr→(SP-2) <sub>M</sub> zzHr→(SP-1) <sub>M</sub> SP <sub>R</sub> -2→SP <sub>R</sub>	.	.	.	.	.	.
	PUSH IX	11 011 101 11 100 101						S/D		2	14	IXLr→(SP-2) <sub>M</sub> IXHr→(SP-1) <sub>M</sub> SP <sub>R</sub> -2→SP <sub>R</sub>	.	.	.	.	.	.
	PUSH IY	11 111 101 11 100 101						S/D		2	14	IYLr→(SP-2) <sub>M</sub> IYHr→(SP-1) <sub>M</sub> SP <sub>R</sub> -2→SP <sub>R</sub>	.	.	.	.	.	.
POP	POP zz	11 zz0 001				D		S		1	9	(SP+1) <sub>M</sub> →zzHr **4 (SP) <sub>M</sub> →zzLr SP <sub>R</sub> +2→SP <sub>R</sub>	.	.	.	.	.	.
	POP IX	11 011 101 11 100 001						S/D		2	12	(SP+1) <sub>M</sub> →IXHr (SP) <sub>M</sub> →IXLr SP <sub>R</sub> +2→SP <sub>R</sub>	.	.	.	.	.	.
	POP IY	11 111 101 11 100 001						S/D		2	12	(SP+1) <sub>M</sub> →IYHr (SP) <sub>M</sub> →IYLr SP <sub>R</sub> +2→SP <sub>R</sub>	.	.	.	.	.	.
Exchange	EX AF,AF'	00 001 000						S/D		1	4	AF <sub>R</sub> →AF' <sub>R</sub>	.	.	.	.	.	.
	EX DE,HL	11 101 011						S/D		1	3	DE <sub>R</sub> →HL <sub>R</sub>	.	.	.	.	.	.
	EXX	11 011 001						S/D		1	3	BC <sub>R</sub> →BC' <sub>R</sub> DE <sub>R</sub> →DE' <sub>R</sub> HL <sub>R</sub> →HL' <sub>R</sub>	.	.	.	.	.	.
	EX (SP),HL	11 100 011						S/D		1	16	Hr→(SP+1) <sub>M</sub> Lr→(SP) <sub>M</sub>	.	.	.	.	.	.
	EX (SP),IX	11 011 101 11 100 011						S/D		2	19	IXHr→(SP+1) <sub>M</sub> IXLr→(SP) <sub>M</sub>	.	.	.	.	.	.
	EX (SP),IY	11 111 101 11 100 011						S/D		2	19	IYHr→(SP+1) <sub>M</sub> IYLr→(SP) <sub>M</sub>	.	.	.	.	.	.

\*4 POP AF writes the stack contents to the flag.

### 3. Program Control Instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag							
			IMMED	EXT	IND	REG	REGI	IMP	REL				7	6	4	2	1	0		
													S	Z	H	P/V	N	C		
Call	CALL mn	11 001 101 < n > < m >		D							3	16	PCHr→(SP-1) <sub>w</sub> PCLr→(SP-2) <sub>w</sub> mn→PC <sub>e</sub> SP <sub>e</sub> +2→SP <sub>e</sub>	.	.	.	.	.	.	
	CALL f,mn	11 f 100 < n > < m >		D							3	6 (f : false) 16 (f : true)	continue : f is false CALL mn : f is true	.	.	.	.	.	.	
Jump	DJNZ j	00 010 000 <j>							D		2	9 (Br≠0) 7 (Br=0)	Br-1→Br continue : Br=0 PC <sub>e</sub> +j→PC <sub>e</sub> : Br≠0	.	.	.	.	.	.	
	JP f,mn	11 f 010 < n > < m >		D							3	6 (f : false)	mn→PC <sub>e</sub> : f is true	.	.	.	.	.	.	
											3	9 (f : true)	continue : f is false	.	.	.	.	.	.	
	JP mn	11 000 011 < n > < m >		D							3	9	mn→PC <sub>e</sub>	.	.	.	.	.	.	
	JP (HL)	11 101 001						D			1	3	HL <sub>e</sub> →PC <sub>e</sub>	.	.	.	.	.	.	
	JP (IX)	11 011 101					D			2	6	IX <sub>e</sub> →PC <sub>e</sub>	.	.	.	.	.	.		
	JP (IY)	11 101 001					D			2	6	IY <sub>e</sub> →PC <sub>e</sub>	.	.	.	.	.	.		
	JR j	00 011 000 <j>							D		2	8	PC <sub>e</sub> +j→PC <sub>e</sub>	.	.	.	.	.	.	
	JR C,j	00 111 000 <j>									D	2	6	continue : C=0	.	.	.	.	.	.
												2	8	PC <sub>e</sub> +j→PC <sub>e</sub> : C=1	.	.	.	.	.	
	JR NC,j	00 110 000 <j>									D	2	6	continue : C=1	.	.	.	.	.	.
												2	8	PC <sub>e</sub> +j→PC <sub>e</sub> : C=0	.	.	.	.	.	
JR Z,j	00 101 000 <j>									D	2	6	continue : Z=0	.	.	.	.	.	.	
											2	8	PC <sub>e</sub> +j→PC <sub>e</sub> : Z=1	.	.	.	.	.		
JR NZ,j	00 100 000 <j>									D	2	6	continue : Z=1	.	.	.	.	.	.	
											2	8	PC <sub>e</sub> +j→PC <sub>e</sub> : Z=0	.	.	.	.	.		
Return	RET	11 001 001							D		1	9	(SP) <sub>w</sub> →PCLr (SP+1) <sub>w</sub> →PCHr SP <sub>e</sub> +2→SP <sub>e</sub>	.	.	.	.	.	.	
	RET f	11 f 000							D		1	5 (f : false) 10 (f : true)	continue : f is false RET : f is true	.	.	.	.	.	.	
	RETI	11 101 101 01 001 101							D		2	22	(SP) <sub>w</sub> →PCLr (SP+1) <sub>w</sub> →PCHr SP <sub>e</sub> +2→SP <sub>e</sub>	.	.	.	.	.	.	
	RETN	11 101 101 01 000 101							D		2	12	(SP) <sub>w</sub> →PCLr (SP+1) <sub>w</sub> →PCHr SP <sub>e</sub> +2→SP <sub>e</sub> IEF <sub>r</sub> →IEF <sub>i</sub>	.	.	.	.	.	.	
Restart	RST v	11 v 111							D		1	11	PCHr→(SP-1) <sub>w</sub> PCLr→(SP-2) <sub>w</sub> 0→PCHr v→PCLr SP <sub>e</sub> +2→SP <sub>e</sub>	.	.	.	.	.	.	

## 4. I/O Instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag						
			IMMED	EXT	IND	REG	REGI	IMP	I/O				7	6	4	2	1	0	
													S	Z	H	P/V	N	C	
INPUT	IN A,(m)	11 011 011 < m >							D	S	2	9	(Am) <sub>1</sub> →Ar m→A <sub>0</sub> ~A <sub>7</sub> Ar→A <sub>8</sub> ~A <sub>15</sub>	.	.	.	.	.	.
	IN g,(C)	11 101 101 01 g 000							D	S	2	9	(BC) <sub>1</sub> →gr g=110 : Only the flags will change. Cr→A <sub>0</sub> ~A <sub>7</sub> Br→A <sub>8</sub> ~A <sub>15</sub>	I	I	R	P	R	.
	IN0 g,(m)	11 101 101 00 g 000 < m >							D	S	3	12	(00m) <sub>1</sub> →gr g=110 : Only the flags will change. m→A <sub>0</sub> ~A <sub>7</sub> 00→A <sub>8</sub> ~A <sub>15</sub>	I	I	R	P	R	.
	IND	11 101 101 10 101 010							D	S	2	12	(BC) <sub>1</sub> →(HL) <sub>n</sub> HL <sub>n</sub> -1→HL <sub>n</sub> Br-1→Br Cr→A <sub>0</sub> ~A <sub>7</sub> Br→A <sub>8</sub> ~A <sub>15</sub>	X	I	X	X	I	X
	INDR	11 101 101 10 111 010							D	S	2	14(Br≠0) 12(Br=0)	(BC) <sub>1</sub> →(HL) <sub>n</sub> Q   HL <sub>n</sub> -1→HL <sub>n</sub>   Br-1→Br Repeat Q until Br=0 Cr→A <sub>0</sub> ~A <sub>7</sub> Br→A <sub>8</sub> ~A <sub>15</sub>	X	S	X	X	I	X
	INI	11 101 101 10 100 010							D	S	2	12	(BC) <sub>1</sub> →(HL) <sub>n</sub> HL <sub>n</sub> +1→HL <sub>n</sub> Br-1→Br Cr→A <sub>0</sub> ~A <sub>7</sub> Br→A <sub>8</sub> ~A <sub>15</sub>	X	I	X	X	I	X
INIR	11 101 101 10 110 010							D	S	2	14(Br≠0) 12(Br=0)	(BC) <sub>1</sub> →(HL) <sub>n</sub> Q   HL <sub>n</sub> +1→HL <sub>n</sub>   Br-1→Br Repeat Q until Br=0 Cr→A <sub>0</sub> ~A <sub>7</sub> Br→A <sub>8</sub> ~A <sub>15</sub>	X	S	X	X	I	X	

\*5 Z = 1 : Br - 1 = 0

Z = 0 : Br - 1 ≠ 0

\*6 N = 1 : MSB of Data = 1

N = 0 : MSB of Data = 0

(Continued)

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag					
			IMMED	EXT	IND	REG	REGI	IMP	I/O				7	6	4	2	1	0
													S	Z	H	P/V	N	C
OUTPUT	OUT (m),A	11 010 011 < m >						S	D	2	10	Ar→(Am), m→A <sub>6</sub> -A <sub>7</sub> Ar→A <sub>6</sub> -A <sub>15</sub>	.	.	.	.	.	.
	OUT (C),g	11 101 101 01 g 001					S		D	2	10	gr→(BC), Cr→A <sub>6</sub> -A <sub>7</sub> Br→A <sub>6</sub> -A <sub>15</sub>	.	.	.	.	.	.
	OUT0 (m),g	11 101 101 00 g 001 < m >					S		D	3	13	gr→(00m), m→A <sub>6</sub> -A <sub>7</sub> 00→A <sub>6</sub> -A <sub>15</sub>	.	.	.	.	.	.
	OTDM	11 101 101 10 001 011					S		D	2	14	(HL) <sub>w</sub> →(00C), HL <sub>n</sub> -1→HL <sub>n</sub> Cr-1→Cr Br-1→Br Cr→A <sub>6</sub> -A <sub>7</sub> 00→A <sub>6</sub> -A <sub>15</sub>	1	1	1	P	1	1
	OTDMR	11 101 101 10 011 011					S		D	2	16(Br≠0) 14(Br=0)	(HL) <sub>w</sub> →(00C), HL <sub>n</sub> -1→HL <sub>n</sub> Cr-1→Cr Br-1→Br Repeat Q until Br=0 Cr→A <sub>6</sub> -A <sub>7</sub> 00→A <sub>6</sub> -A <sub>15</sub>	R	S	R	S	1	R
	OTDR	11 101 101 10 111 011					S		D	2	14(Br≠0) 12(Br=0)	(HL) <sub>w</sub> →(BC), Q HL <sub>n</sub> -1→HL <sub>n</sub> Br-1→Br Repeat Q until Br=0 Cr→A <sub>6</sub> -A <sub>7</sub> Br→A <sub>6</sub> -A <sub>15</sub>	X	S	X	X	1	X
	OUTI	11 101 101 10 100 011					S		D	2	12	(HL) <sub>w</sub> →(BC), HL <sub>n</sub> +1→HL <sub>n</sub> Br-1→Br Cr→A <sub>6</sub> -A <sub>7</sub> Br→A <sub>6</sub> -A <sub>15</sub>	X	1	X	X	1	X
	OTIR	11 101 101 10 110 011					S		D	2	14(Br≠0) 12(Br=0)	(HL) <sub>w</sub> →(BC), Q HL <sub>n</sub> +1→HL <sub>n</sub> Br-1→Br Repeat Q until Br=0 Cr→A <sub>6</sub> -A <sub>7</sub> Br→A <sub>6</sub> -A <sub>15</sub>	X	S	X	X	1	X
	TSTIO m	11 101 101 01 110 100 < m >	S						S	3	12	(00C), m Cr→A <sub>6</sub> -A <sub>7</sub> 00→A <sub>6</sub> -A <sub>15</sub>	1	1	S	P	R	R
	OTIM	11 101 101 10 000 011					S		D	2	14	(HL) <sub>w</sub> →(00C), HL <sub>n</sub> +1→HL <sub>n</sub> Cr+1→Cr Br-1→Br Cr→A <sub>6</sub> -A <sub>7</sub> 00→A <sub>6</sub> -A <sub>15</sub>	1	1	1	P	1	1
	OTIMR	11 101 101 10 010 011					S		D	2	16(Br≠0) 14(Br=0)	(HL) <sub>w</sub> →(00C), Q HL <sub>n</sub> +1→HL <sub>n</sub> Cr+1→Cr Br-1→Br Repeat Q until Br=0 Cr→A <sub>6</sub> -A <sub>7</sub> 00→A <sub>6</sub> -A <sub>15</sub>	R	S	R	S	1	R
	OUTD	11 101 101 10 101 011					S		D	2	12	(HL) <sub>w</sub> →(BC), HL <sub>n</sub> -1→HL <sub>n</sub> Br-1→Br Cr→A <sub>6</sub> -A <sub>7</sub> Br→A <sub>6</sub> -A <sub>15</sub>	X	1	X	X	1	X

\*7 Z = 1 : Br - 1 = 0

Z = 0 : Br - 1 ≠ 0

\*8 N = 1 : MSB of Data = 1

N = 0 : MSB of Data = 0

## 5. Special Control Instructions

Operation name	MNEMONICS	OP code	Addressing							Bytes	States	Operation	Flag							
			IMMED	EXT	IND	REG	REGI	IMP	REL				7	6	4	2	1	0		
													S	Z	H	P/V	N	C		
Special Function	DAA	00 100 111							S/D		1	4	Decimal Adjust Accumulator	I	I	I	P	.	I	
Carry Control	CCF	00 111 111									1	3	$\bar{C}$ →C	.	.	R	.	R	I	
	SCF	00 110 111									1	3	1→C	.	.	R	.	R	S	
CPU Control	DI	11 110 011									1	3	0→IEF <sub>0</sub> , 0→IEF <sub>2</sub> *9	.	.	.	.	.	.	
	EI	11 111 011									1	3	1→IEF <sub>0</sub> , 1→IEF <sub>2</sub> *9	.	.	.	.	.	.	
	HALT	01 110 110									1	3	CPU halted	.	.	.	.	.	.	
	IM 0		11 101 101									2	6	Interrupt mode 0	.	.	.	.	.	.
			01 000 110												.	.	.	.	.	.
	IM 1		11 101 101									2	6	Interrupt mode 1	.	.	.	.	.	.
			01 010 110												.	.	.	.	.	.
	IM 2		11 101 101									2	6	Interrupt mode 2	.	.	.	.	.	.
			01 011 110												.	.	.	.	.	.
	NOP		00 000 000									1	3	No operation	.	.	.	.	.	.
	SLP		11 101 101									2	8	Sleep	.	.	.	.	.	.
		01 110 110												.	.	.	.	.	.	

\*9 No interrupts are sampled at the end of a DI or EI instruction.



## B. Alphabetical List of Instructions

MNEMONICS	Bytes	Machine Cycles	States
ADC A, m	2	2	6
ADC A, g	1	2	4
ADC A, (HL)	1	2	6
ADC A, (IX+d)	3	6	14
ADC A, (IY+d)	3	6	14
ADD A, m	2	2	6
ADD A, g	1	2	4
ADD A, (HL)	1	2	6
ADD A, (IX+d)	3	6	14
ADD A, (IY+d)	3	6	14
ADC HL, ww	2	6	10
ADD HL, ww	1	5	7
ADD IX, xx	2	6	10
ADD IY, yy	2	6	10
AND m	2	2	6
AND g	1	2	4
AND (HL)	1	2	6
AND (IX+d)	3	6	14
AND (IY+d)	3	6	14
BIT b, (HL)	2	3	9
BIT b, (IX+d)	4	5	15
BIT b, (IY+d)	4	5	15
BIT b, g	2	2	6
CALL f, mn	3	2	6
			(If condition is false)
	3	6	16
			(If condition is true)
CALL mn	3	6	16
CCF	1	1	3
CPD	2	6	12
CPDR	2	8	14
			(If $BC_R \neq 0$ and $Ar \neq (HL)_M$ )
	2	6	12
			(If $BC_R = 0$ or $Ar = (HL)_M$ )
CP(HL)	1	2	6
CPI	2	6	12
CPIR	2	8	14
			(If $BC_R \neq 0$ and $Ar \neq (HL)_M$ )

(Continued)

MNEMONICS	Bytes	Machine Cycles	States
CPIR	2	6	12 (If $BC_R=0$ or $Ar=(HL)_M$ )
CP (IX+d)	3	6	14
CP (IY+d)	3	6	14
CPL	1	1	3
CP m	2	2	6
CP g	1	2	4
DAA	1	2	4
DEC (HL)	1	4	10
DEC IX	2	3	7
DEC IY	2	3	7
DEC (IX+d)	3	8	18
DEC (IY+d)	3	8	18
DEC g	1	2	4
DEC ww	1	2	4
DI	1	1	3
DJNZ j	2	5	9(If $Br \neq 0$ )
	2	3	7(If $Br=0$ )
EI	1	1	3
EX AF, AF'	1	2	4
EX DE, HL	1	1	3
EX (SP), HL	1	6	16
EX (SP), IX	2	7	19
EX (SP), IY	2	7	19
EXX	1	1	3
HALT	1	1	3
IM 0	2	2	6
IM 1	2	2	6
IM 2	2	2	6
INC g	1	2	4
INC (HL)	1	4	10
INC (IX+d)	3	8	18
INC (IY+d)	3	8	18
INC ww	1	2	4
INC IX	2	3	7
INC IY	2	3	7
IN A, (m)	2	3	9
IN g, (C)	2	3	9
INI	2	4	12
INIR	2	6	14(If $Br \neq 0$ )

(Continued)

MNEMONICS	Bytes	Machine Cycles	States
INIR	2	4	12 (If Br=0)
IND	2	4	12
INDR	2	6	14 (If Br≠0)
	2	4	12 (If Br=0)
IN0 g, (m)	3	4	12
JP f, mn	3	2	6
			(If f is false)
	3	3	9
			(If f is true)
JP (HL)	1	1	3
JP (IX)	2	2	6
JP (IY)	2	2	6
JP mn	3	3	9
JR j	2	4	8
JR C, j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
JR NC, j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
JR Z, j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
JR NZ, j	2	2	6
			(If condition is false)
	2	4	8
			(If condition is true)
LD A, (BC)	1	2	6
LD A, (DE)	1	2	6
LD A, I	2	2	6
LD A, (mn)	3	4	12
LD A, R	2	2	6
LD (BC), A	1	3	7
LDD	2	4	12
LD (DE), A	1	3	7
LD ww, mn	3	3	9
LD ww, (mn)	4	6	18

(Continued)

MNEMONICS	Bytes	Machine Cycles	States
LDDR	2	6	14 (If $BC_R \neq 0$ )
	2	4	12 (If $BC_R = 0$ )
LD (HL), m	2	3	9
LD HL, (mn)	3	5	15
LD (HL), g	1	3	7
LDI	2	4	12
LD I, A	2	2	6
LDIR	2	6	14 (If $BC_R \neq 0$ )
	2	4	12 (If $BC_R = 0$ )
LD IX, mn	4	4	12
LD IX, (mn)	4	6	18
LD (IX+d), m	4	5	15
LD (IX+d), g	3	7	15
LD IY, mn	4	4	12
LD IY, (mn)	4	6	18
LD (IY+d), m	4	5	15
LD (IY+d), g	3	7	15
LD (mn), A	3	5	13
LD (mn), ww	4	7	19
LD (mn), HL	3	6	16
LD (mn), IX	4	7	19
LD (mn), IY	4	7	19
LD R, A	2	2	6
LD g, (HL)	1	2	6
LD g, (IX+d)	3	6	14
LD g, (IY+d)	3	6	14
LD g, m	2	2	6
LD g, g'	1	2	4
LD SP, HL	1	2	4
LD SP, IX	2	3	7
LD SP, IY	2	3	7
MLT ww	2	13	17
NEG	2	2	6
NOP	1	1	3
OR (HL)	1	2	6
OR (IX+d)	3	6	14
OR (IY+d)	3	6	14
OR m	2	2	6
OR g	1	2	4
OTDM	2	6	14

(Continued)

MNEMONICS	Bytes	Machine Cycles	States
OTDMR	2	8	16 (If B <sub>r</sub> ≠ 0)
	2	6	14 (If B <sub>r</sub> = 0)
OTDR	2	6	14 (If B <sub>r</sub> ≠ 0)
	2	4	12 (If B <sub>r</sub> = 0)
OTIM	2	6	14
OTIMR	2	8	16 (If B <sub>r</sub> ≠ 0)
	2	6	14 (If B <sub>r</sub> = 0)
OTIR	2	6	14 (If B <sub>r</sub> ≠ 0)
	2	4	12 (If B <sub>r</sub> = 0)
OUTD	2	4	12
OUTI	2	4	12
OUT (m), A	2	4	10
OUT (C), g	2	4	10
OUT0 (m), g	3	5	13
POP IX	2	4	12
POP IY	2	4	12
POP zz	1	3	9
PUSH IX	2	6	14
PUSH IY	2	6	14
PUSH zz	1	5	11
RES b, (HL)	2	5	13
RES b, (IX+d)	4	7	19
RES b, (IY+d)	4	7	19
RES b, g	2	3	7
RET	1	3	9
RET f	1	3	5
			(If condition is false)
	1	4	10
			(If condition is true)
RETI	2	10	22
RETN	2	4	12
RLA	1	1	3
RLCA	1	1	3
RLC (HL)	2	5	13
RLC (IX+d)	4	7	19
RLC (IY+d)	4	7	19
RLC g	2	3	7
RLD	2	8	16
RL (HL)	2	5	13

(Continued)

MNEMONICS	Bytes	Machine Cycles	States
RL (IX+d)	4	7	19
RL (IY+d)	4	7	19
RL g	2	3	7
RRA	1	1	3
RRCA	1	1	3
RRC (HL)	2	5	13
RRC (IX+d)	4	7	19
RRC (IY+d)	4	7	19
RRC g	2	3	7
RRD	2	8	16
RR (HL)	2	5	13
RR (IX+d)	4	7	19
RR (IY+d)	4	7	19
RR g	2	3	7
RST v	1	5	11
SBC A, (HL)	1	2	6
SBC A, (IX+d)	3	6	14
SBC A, (IY+d)	3	6	14
SBC A, m	2	2	6
SBC A, g	1	2	4
SBC HL, ww	2	6	10
SCF	1	1	3
SET b, (HL)	2	5	13
SET b, (IX+d)	4	7	19
SET b, (IY+d)	4	7	19
SET b, g	2	3	7
SLA (HL)	2	5	13
SLA (IX+d)	4	7	19
SLA (IY+d)	4	7	19
SLA g	2	3	7
SLP	2	2	8
SRA (HL)	2	5	13
SRA (IX+d)	4	7	19
SRA (IY+d)	4	7	19
SRA g	2	3	7
SRL (HL)	2	5	13
SRL (IX+d)	4	7	19
SRL (IY+d)	4	7	19
SRL g	2	3	7
SUB (HL)	1	2	6

(Continued)

MNEMONICS	Bytes	Machine Cycles	States
SUB (IX+d)	3	6	14
SUB (IY+d)	3	6	14
SUB m	2	2	6
SUB g	1	2	4
TSTIO m	3	4	12
TST g	2	3	7
TST m	3	3	9
TST (HL)	2	4	10
XOR (HL)	1	2	6
XOR (IX+d)	3	6	14
XOR (IY+d)	3	6	14
XOR m	2	2	6
XOR g	1	2	4

# C. Opcode Maps

Table 1. Opcode Map (1)

First opcode

Instruction format: XX

S (HI=ALL)		LO		ww (LO=ALL)				g (LO=0~7)								LO=0~7								
				BC	DE	HL	SP	B	D	H	(HL)	B	D	H	(HL)	1000	1001	1010	1011	BC	DE	HL	AF	zz
				0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	00H	10H	20H	30H	v
				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F					
B	0000	0	NOP	DJNZ j	JR NZ,j	JR NC,j															RET f	0		
C	0001	1	LD ww, mn																			POP zz	1	
D	0010	2	LD(ww), A	LD(mn), HL	LD(mn), A																JP mn	3		
E	0011	3	INC ww																			CALL f, mn	4	
H	0100	4	INC g																			PUSH zz	5	
L	0101	5	DEC g																			ADD A, m	6	
(HL)	0110	6	LD g, m																			SUB m	6	
A	0111	7	RLCA	RLA	DAA	SCF																RST v	7	
B	1000	8	EXAF, AF	JR j	JR Z, j	JR C, j																RET f	8	
C	1001	9	ADD HL, ww																			RET	9	
D	1010	A	LD A, (ww)	LD HL, (mn)	LD A, (mn)																	LD SP, HL	9	
E	1011	B	DEC ww																			JP f, mn	A	
H	1100	C	INC g																			Table 2	B	
L	1101	D	DEC g																			CALL f, mn	C	
(HL)	1110	E	LD g, m																			CALL mn	D	
A	1111	F	RRCA	RRA	CPL	COF																RST v	F	
			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F						
			C	E	L	A	C	E	L	A					Z	C	PE	M				f		
			g (LO=8~F)													08H	18H	28H	38H	v				
																LO=8~F								



\*1 g is replaced by (HL).

\*2 s is replaced by (HL).

\*3 If DD is added to the beginning of an opcode (DD XX), only the instructions having HL, (HL) as an operand are replaced with

$$\left\{ \begin{array}{l} \text{HL} \rightarrow \text{IX} \\ (\text{HL}) \rightarrow (\text{IX} + \text{d}) \end{array} \right\}$$

to perform the same operation.

(Example)

22H; LD (mn), HL  
↓  
DDH 22H; LD (mn), IX

If FD is added to the beginning of the opcode (FD XX), it is replaced by

$$\left\{ \begin{array}{l} \text{HL} \rightarrow \text{IY} \\ (\text{HL}) \rightarrow (\text{IY} + \text{d}) \end{array} \right\}$$

to perform the same operation.

(Example)

34H; INC (HL)  
↓  
FDH 34H; INC (IY + d)

As an exception, when DDH, FDH is added to the beginning of JP (HL) of E9H, (HL) is replaced by (IX), (IY).

If DDH, FDH is added to the beginning of EX DE, HL of EBH, HL is not replaced. It becomes an undefined instruction.

**Table 2. Opcode Map (2)**

Second opcode

Instruction format: CB XX

		<table border="1"> <tr> <th>HI</th> <th>LO</th> </tr> <tr> <td>0000</td> <td>0</td> </tr> <tr> <td>0001</td> <td>1</td> </tr> <tr> <td>0010</td> <td>2</td> </tr> <tr> <td>0011</td> <td>3</td> </tr> <tr> <td>0100</td> <td>4</td> </tr> <tr> <td>0101</td> <td>5</td> </tr> <tr> <td>0110</td> <td>6</td> </tr> <tr> <td>0111</td> <td>7</td> </tr> <tr> <td>1000</td> <td>8</td> </tr> <tr> <td>1001</td> <td>9</td> </tr> <tr> <td>1010</td> <td>A</td> </tr> <tr> <td>1011</td> <td>B</td> </tr> <tr> <td>1100</td> <td>C</td> </tr> <tr> <td>1101</td> <td>D</td> </tr> <tr> <td>1110</td> <td>E</td> </tr> <tr> <td>1111</td> <td>F</td> </tr> </table>		HI	LO	0000	0	0001	1	0010	2	0011	3	0100	4	0101	5	0110	6	0111	7	1000	8	1001	9	1010	A	1011	B	1100	C	1101	D	1110	E	1111	F	b (LO=0~7)																
				HI	LO																																																	
				0000	0																																																	
0001	1																																																					
0010	2																																																					
0011	3																																																					
0100	4																																																					
0101	5																																																					
0110	6																																																					
0111	7																																																					
1000	8																																																					
1001	9																																																					
1010	A																																																					
1011	B																																																					
1100	C																																																					
1101	D																																																					
1110	E																																																					
1111	F																																																					
				0	2	4	6	0	2	4	6	0	2	4	6																																							
				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F																																			
g	(HI=ALL)	B	0000	0					BIT b,g				RES b,g				SET b,g				0																																	
		C	0001	1					BIT b,g				RES b,g				SET b,g				1																																	
		D	0010	2					BIT b,g				RES b,g				SET b,g				2																																	
		E	0011	3					BIT b,g				RES b,g				SET b,g				3																																	
		H	0100	4	RLC g	RL g	SLA g		BIT b,g				RES b,g				SET b,g				4																																	
		L	0101	5					BIT b,g				RES b,g				SET b,g				5																																	
		(HL)	0110	6	*	*	*		BIT b,g				RES b,g				SET b,g				6																																	
		A	0111	7					BIT b,g				RES b,g				SET b,g				7																																	
	B	1000	8					BIT b,g				RES b,g				SET b,g				8																																		
	C	1001	9					BIT b,g				RES b,g				SET b,g				9																																		
	D	1010	A					BIT b,g				RES b,g				SET b,g				A																																		
	E	1011	B					BIT b,g				RES b,g				SET b,g				B																																		
	H	1100	C	RRC g	RR g	SRA g	SRL g		BIT b,g				RES b,g				SET b,g				C																																	
	L	1101	D					BIT b,g				RES b,g				SET b,g				D																																		
	(HL)	1110	E	*	*	*	*	BIT b,g				RES b,g				SET b,g				E																																		
	A	1111	F					BIT b,g				RES b,g				SET b,g				F																																		
				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F																																			
								1	3	5	7	1	3	5	7	1	3	5	7																																			
				b (LO=8~F)																																																		

\* In the instruction to be executed, DDH can be added to the beginning of the opcode and (HL) is replaced by (IX + d) in opcode DD CB d XX. In the same way, FDH can be added to the beginning of the opcode. In the instruction to be executed, (HL) is replaced by (IY + d) in opcode FD CB d XX.

Table 3. Opcode Map (3)

Second opcode

Instruction format: ED XX

LO \ HI		ww (LO=ALL)								g (LO=0~7)								BC		DE		HL		SP	
		g (LO=0~7)				g (LO=0~7)																			
		B	D	H		B	D	H																	
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	8	9	A	B	C	D	E	F		
0000	0	IN0 g, (m)				IN g, (C)						LDI	LDIR							0					
0001	1	OUT0 (m),g				OUT (C),g						CPI	CPIR							1					
0010	2					SBC HL, ww						INI	INIR							2					
0011	3					LD (mn), ww				OTIM	OTIMR	OUTI	OTIR							3					
0100	4	TST g		TST (HL)	NEG			TST m	TST0m									4							
0101	5					RETN												5							
0110	6					IM 0	IM 1			SLP									6						
0111	7					LD I,A	LD A,I	RRD										7							
1000	8	IN0 g, (m)				IN g, (C)						LDD	LDDR							8					
1001	9	OUT0 (m),g				OUT (C),g						CPD	CPDR							9					
1010	A					ADC HL, ww						IND	INDR							A					
1011	B					LD ww, (mn)				OTDM	OTDMR	OUTD	OTDR							B					
1100	C	TST g				MLT ww												C							
1101	D					RETI												D							
1110	E									IM 2									E						
1111	F					LD R,A	LD A,R	RLD										F							
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F								
		C	E	L	A	C	E	L	A																
		g (LO=8~F)																							

## D. Bus Cycle States

'\*' in the ADDRESS column indicates that the address output is undefined and 'Z' in the DATA column indicates that the data pin is in the high-impedance state. The  $\overline{\text{LIR}}$  pin output value is obtained when the LIRE bit in the operation mode control register is 1.

Instruction	Machine Cycle	St <sub>1</sub> St <sub>2</sub> n	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{MF}}$	$\overline{\text{IOF}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
ADD HL,ww	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub> ~MC <sub>6</sub>	T <sub>1</sub> T <sub>1</sub> T <sub>1</sub> T <sub>1</sub>	*	Z	1	1	1	1	1	1	1
ADD IX,xx ADD IY,yy	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub> ~MC <sub>6</sub>	T <sub>1</sub> T <sub>1</sub> T <sub>1</sub> T <sub>1</sub>	*	Z	1	1	1	1	1	1	1
ADC HL,ww SBC HL,ww	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub> ~MC <sub>6</sub>	T <sub>1</sub> T <sub>1</sub> T <sub>1</sub> T <sub>1</sub>	*	Z	1	1	1	1	1	1	1
ADD A,g ADC A,g SUB g SBC A,g AND g OR g XOR g CP g	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
ADD A,m ADC A,m SUB m SBC A,m AND m OR m XOR m CP m	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	m	0	1	0	1	1	1	1
ADD A, (HL) ADC A, (HL) SUB (HL) SBC A, (HL) AND (HL) OR (HL) XOR (HL) CP (HL)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
ADD A, (IX+d) ADD A, (IY+d) ADC A, (IX+d) ADC A, (IY+d) SUB (IX+d) SUB (IY+d) SBC A, (IX+d)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
SBC A, (Y+d) AND (X+d) AND (Y+d) OR (X+d) OR (Y+d) XOR (X+d) XOR (Y+d) CP (X+d) CP (Y+d)	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	d	0	1	0	1	1	1	1
	MC <sub>4</sub> ~MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	IX+d Y+d	DATA	0	1	0	1	1	1	1
BIT b,g	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
BIT b, (HL)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
BIT b, (IX+d) BIT b, (Y+d)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	d	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	3rd op-code Address	3rd op-code	0	1	0	1	0	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	IX+d Y+d	DATA	0	1	0	1	1	1	1
CALL mn	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-1	PCH	1	0	0	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-2	PCL	1	0	0	1	1	1	1
CALL f,mn (if condition is false)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
CALL f,mn (If condition is true)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-1	PCH	1	0	0	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-2	PCL	1	0	0	1	1	1	1
CCF	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
CPI CPD	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub> ~MC <sub>6</sub>	T <sub>i</sub> T <sub>i</sub> T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
CPIR CPDR (If BC <sub>r</sub> ≠0 and Ar≠(HL) <sub>M</sub> )	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub> ~MC <sub>6</sub>	T <sub>i</sub> T <sub>i</sub> T <sub>i</sub> T <sub>i</sub> T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
CPIR CPDR (If BC <sub>r</sub> =0 or Ar=(HL) <sub>M</sub> )	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub> ~MC <sub>6</sub>	T <sub>i</sub> T <sub>i</sub> T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
CPL	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
DAA	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
DI *1	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0

\*1 No interrupts are sampled at the end of a DI instruction.

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
DJNZ j (if Br≠0)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>i</sub> * <sup>2</sup>	*	Z	1	1	1	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	j-2	0	1	0	1	1	1	1
	MC <sub>4</sub> ~MC <sub>5</sub>	T <sub>i</sub> T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
DJNZ j (if Br=0)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>i</sub> * <sup>2</sup>	*	Z	1	1	1	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	j-2	0	1	0	1	1	1	1
EI * <sup>3</sup>	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
EX DE, HL EXX	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
EX AF, AF'	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
EX (SP), HL	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP	DATA	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP+1	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP+1	H	1	0	0	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP	L	1	0	0	1	1	1	1
EX (SP),IX EX (SP),IY	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP+1	DATA	0	1	0	1	1	1	1
	MC <sub>5</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1

(Continued)

\*2 DMA, refresh, and bus release cannot be executed immediately after this state (their requests are ignored).

\*3 No interrupts are sampled at the end of an EI instruction.

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
EX (SP), IX EX (SP), IY	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP+1	IXH IYH	1	0	0	1	1	1	1
	MC <sub>7</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP	IXL IYL	1	0	0	1	1	1	1
HALT	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	—	—	Next op-code Address	Next op-code	0	1	0	1	0	0	0
IM 0 IM 1 IM 2	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
INC g DEC g	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
INC (HL) DEC (HL)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	1	0	0	1	1	1	1
INC (IX+d) INC (IY+d) DEC (IX+d) DEC (IY+d)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	d	0	1	0	1	1	1	1
	MC <sub>4</sub> ~MC <sub>5</sub>	T <sub>i</sub> T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	IX+d IY+d	DATA	0	1	0	1	1	1	1
	MC <sub>7</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>8</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	IX+d IY+d	DATA	1	0	0	1	1	1	1
INC ww DEC ww	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
INC IX INC IY DEC IX DEC IY	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1

(Continued)



Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
IN A,(m)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	m	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	m to A <sub>0</sub> ~A <sub>7</sub> A to A <sub>8</sub> ~A <sub>15</sub>	DATA	0	1	1	0	1	1	1
IN g,(C)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	BC	DATA	0	1	1	0	1	1	1
INO g,(m)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	m	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	m to A <sub>0</sub> ~A <sub>7</sub> 00H to A <sub>8</sub> ~A <sub>15</sub>	DATA	0	1	1	0	1	1	1
INI IND	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	BC	DATA	0	1	1	0	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	1	0	0	1	1	1	1
INIR INDR (if Br≠0)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	BC	DATA	0	1	1	0	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	1	0	0	1	1	1	1
	MC <sub>5</sub> ~MC <sub>8</sub>	TITI	*	Z	1	1	1	1	1	1	1
INIR INDR (if Br=0)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	BC	DATA	0	1	1	0	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	1	0	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{LIR}}$	$\overline{\text{HALT}}$	ST
JP mn	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
JP f,mn (If f is false)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
JP f,mn (If f is true)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
JP (HL)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
JP (IX) JP (IY)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
JR j	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	j-2	0	1	0	1	1	1	1
	MC <sub>3</sub> ~MC <sub>4</sub>	TiTi	*	Z	1	1	1	1	1	1	1
JR C,j JR NC,j JR Z,j JR NZ,j (If condition is false)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	j-2	0	1	0	1	1	1	1
JR C,j JR NC,j JR Z,j JR NZ,j (If condition is true)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	j-2	0	1	0	1	1	1	1
	MC <sub>3</sub> ~MC <sub>4</sub>	TiTi	*	Z	1	1	1	1	1	1	1
LD g,g'	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	Ti	*	Z	1	1	1	1	1	1	1
LD g,m	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	m	0	1	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{OE}$	$\overline{UR}$	$\overline{HALT}$	ST
LD g, (HL)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
LD g, (IX+d) LD g, (IY+d)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	d	0	1	0	1	1	1	1
	MC <sub>4</sub> ~MC <sub>5</sub>	TiTi	*	Z	1	1	1	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	IX+d IY+d	DATA	0	1	0	1	1	1	1
LD (HL),g	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	Ti	*	Z	1	1	1	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	g	1	0	0	1	1	1	1
LD (IX+d),g LD (IY+d),g	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	d	0	1	0	1	1	1	1
	MC <sub>4</sub> ~MC <sub>6</sub>	TiTiTi	*	Z	1	1	1	1	1	1	1
	MC <sub>7</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	IX+d IY+d	g	1	0	0	1	1	1	1
LD (HL),m	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	m	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	1	0	0	1	1	1	1
LD (IX+d),m LD (IY+d),m	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	d	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	IX+d IY+d	DATA	1	0	0	1	1	1	1
LD A, (BC) LD A, (DE)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	HALT	ST
LD A, (BC) LD A, (DE)	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	BC DE	DATA	0	1	0	1	1	1	1
LD A, (mn)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn	DATA	0	1	0	1	1	1	1
LD (BC),A LD (DE),A	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	BC DE	A	1	0	0	1	1	1	1
LD (mn),A	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn	A	1	0	0	1	1	1	1
LD A, I *4 LD A, R *4 LD I, A LD R, A	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
LD ww, mn	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
LD IX, mn LD IY, mn	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
LD HL, (mn)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1

\*4 No interrupts are sampled at the end of a LD A, I or LD A, R instruction.

(Continued)

Instruction	Machine Cycle	Status	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LD HL, (mn)	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn	DATA	0	1	0	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn+1	DATA	0	1	0	1	1	1	1
LD ww,(mn)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn	DATA	0	1	0	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn+1	DATA	0	1	0	1	1	1	1
LD IX,(mn) LD IY,(mn)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn	DATA	0	1	0	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn+1	DATA	0	1	0	1	1	1	1
LD (mn),HL	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn	L	1	0	0	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn+1	H	1	0	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LR}$	$\overline{HALT}$	ST
LD (mn),vw	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
	MC <sub>5</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn	wwL	1	0	0	1	1	1	1
	MC <sub>7</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn+1	wwH	1	0	0	1	1	1	1
LD (mn),IX LD (mn),IY	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	n	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd operand Address	m	0	1	0	1	1	1	1
	MC <sub>5</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn	IXL IYL	1	0	0	1	1	1	1
	MC <sub>7</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	mn+1	IXH IYH	1	0	0	1	1	1	1
LD SP, HL	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
LD SP,IX LD SP,IY	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
LDI LDD	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	DE	DATA	1	0	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
LDIR LDDR (if $BC_R \neq 0$ )	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	DE	DATA	1	0	0	1	1	1	1
	MC <sub>5</sub> ~MC <sub>6</sub>	TiTi	*	Z	1	1	1	1	1	1	1
LDIR LDDR (if $BC_R = 0$ )	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	DE	DATA	1	0	0	1	1	1	1
MLT ww	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub> ~MC <sub>13</sub>	TiTiTiTi TiTiTiTi TiTiTi	*	Z	1	1	1	1	1	1	1
NEG	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
NOP	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
OUT (m),A	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	m	0	1	0	1	1	1	1
	MC <sub>3</sub>	Ti	*	Z	1	1	1	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	m to A <sub>0</sub> ~A <sub>7</sub> A to A <sub>8</sub> ~A <sub>15</sub>	A	1	0	1	0	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{UR}$	$\overline{HALT}$	ST
OUT (C),g	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	BC	g	1	0	1	0	1	1	1
OUTO (m),g	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	m	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	m to A <sub>0</sub> ~A <sub>7</sub> 00H to A <sub>8</sub> ~A <sub>15</sub>	g	1	0	1	0	1	1	1
OTIM OTDM	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	C to A <sub>0</sub> ~A <sub>7</sub> 00H to A <sub>8</sub> ~A <sub>15</sub>	DATA	1	0	1	0	1	1	1
	MC <sub>6</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
OTIMR OTDMR (if Br≠0)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	C to A <sub>0</sub> ~A <sub>7</sub> 00H to A <sub>8</sub> ~A <sub>15</sub>	DATA	1	0	1	0	1	1	1
	MC <sub>6</sub> ~MC <sub>8</sub>	T <sub>i</sub> T <sub>i</sub> T <sub>i</sub>	*	Z	1	1	1	1	1	1	1

(Continued)



Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	ME	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
OTIMR OTDMR (If Br=0)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	C to A <sub>0</sub> ~A <sub>7</sub> OOH to A <sub>8</sub> ~A <sub>15</sub>	DATA	1	0	1	0	1	1	1
	MC <sub>6</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
OUTI OUTD	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	BC	DATA	1	0	1	0	1	1	1
OTIR OTDR (If Br≠0)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	BC	DATA	1	0	1	0	1	1	1
	MC <sub>5</sub> ~MC <sub>6</sub>	T <sub>i</sub> T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
OTIR OTDR (If Br=0)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	BC	DATA	1	0	1	0	1	1	1
POP zz	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP	DATA	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP+1	DATA	0	1	0	1	1	1	1
POP IX POP IY	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LJR}$	$\overline{HALT}$	ST
POP IX POP IY	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP+1	DATA	0	1	0	1	1	1	1
PUSH zz	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub> ~MC <sub>3</sub>	TiTi	*	Z	1	1	1	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-1	zzH	1	0	0	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-2	zzL	1	0	0	1	1	1	1
PUSH IX PUSH IY	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub> ~MC <sub>4</sub>	TiTi	*	Z	1	1	1	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-1	IXH IYH	1	0	0	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-2	IXL IYL	1	0	0	1	1	1	1
RET	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP	DATA	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP+1	DATA	0	1	0	1	1	1	1
RET f (If condition is false)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub> ~MC <sub>3</sub>	TiTi	*	Z	1	1	1	1	1	1	1
RET f (If condition is true)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	Ti	*	Z	1	1	1	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP+1	DATA	0	1	0	1	1	1	1
RETN	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP+1	DATA	0	1	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
RETI	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0* <sup>5</sup> 1	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0* <sup>5</sup> 1	1	1
	MC <sub>3</sub> ~MC <sub>5</sub>	T <sub>1</sub> T <sub>1</sub> T <sub>1</sub>	*	Z	1	1	1	1	1* <sup>5</sup> 1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0* <sup>5</sup> 0	1	1
	MC <sub>7</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1* <sup>5</sup> 1	1	1
	MC <sub>8</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0* <sup>5</sup> 0	1	1
	MC <sub>9</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP	data	0	1	0	1	1* <sup>5</sup> 1	1	1
	MC <sub>10</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP+1	data	0	1	0	1	1* <sup>5</sup> 1	1	1
RLCA RLA RRCA RRA	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
RLC g RL g RRC g RR g SLA g SRA g SRL g	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
RLC (HL) RL (HL) RRC (HL) RR (HL) SLA (HL) SRA (HL) SRL (HL)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	1	0	0	1	1	1	1

(Continued)

\*5 The upper column indicates the  $\overline{LIR}$  pin value when the LIRE bit in the operation mode control register is 1, and the lower column indicates that when the same bit is 0.

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
RLC (IX+d) RLC (IY+d) RL (IX+d) RL (IY+d) RRC (IX+d) RRC (IY+d) RR (IX+d) RR (IY+d) SLA (IX+d) SLA (IY+d) SRA (IX+d) SRA (IY+d) SRL (IX+d) SRL (IY+d)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	d	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	3rd op-code Address	3rd op-code	0	1	0	1	0	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	IX+d IY+d	DATA	0	1	0	1	1	1	1
	MC <sub>6</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>7</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	IX+d IY+d	DATA	1	0	0	1	1	1	1
RLD RRD	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub> ~MC <sub>7</sub>	T <sub>i</sub> T <sub>i</sub> T <sub>i</sub> T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>8</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	1	0	0	1	1	1	1
RST v	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub> ~MC <sub>3</sub>	T <sub>i</sub> T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-1	PCH	1	0	0	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-2	PCL	1	0	0	1	1	1	1
SCF	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
SET b,g RES b,g	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
SET b, (HL) RES b, (HL)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	1	0	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
SET b, (IX+d) SET b, (IY+d) RES b, (IX+d) RES b, (IY+d)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	d	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	3rd op-code Address	3rd op-code	0	1	0	1	0	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	IX+d IY+d	DATA	0	1	0	1	1	1	1
	MC <sub>6</sub>	Ti	*	Z	1	1	1	1	1	1	1
	MC <sub>7</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	IX+d IY+d	DATA	1	0	0	1	1	1	1
SLP	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	—	—	FFFFFH	Z	1	1	1	1	1	0	1
TSTIO m	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	m	0	1	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	C to A <sub>0</sub> ~A <sub>7</sub> OOH to A <sub>8</sub> ~A <sub>15</sub>	DATA	0	1	1	0	1	1	1
TST g	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	Ti	*	Z	1	1	1	1	1	1	1
TST m	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st operand Address	m	0	1	0	1	1	1	1
TST (HL)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	1st op-code Address	1st op-code	0	1	0	1	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	2nd op-code Address	2nd op-code	0	1	0	1	0	1	1
	MC <sub>3</sub>	Ti	*	Z	1	1	1	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	HL	DATA	0	1	0	1	1	1	1

(Continued)

## Interrupts

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{IOE}$	$\overline{LIR}$	$\overline{HALT}$	ST
$\overline{NMI}$	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	Next op-code Address (PC)		0	1	0	1	0	1	0
	MC <sub>2</sub> ~MC <sub>3</sub>	TiTi	*	Z	1	1	1	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-1	PCH	1	0	0	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-2	PCL	1	0	0	1	1	1	1
	$\overline{INT_0}$ MODE 0 (RST INSERTED)	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>w</sub> T <sub>w</sub> T <sub>3</sub>	Next op-code Address (PC)	1st op-code	1	1	1	0	0	1
MC <sub>2</sub> ~MC <sub>3</sub>		TiTi	*	Z	1	1	1	1	1	1	1
MC <sub>4</sub>		T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-1	PCH	1	0	0	1	1	1	1
MC <sub>5</sub>		T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-2	PCL	1	0	0	1	1	1	1
$\overline{INT_0}$ MODE 0 (CALL INSERTED)		MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>w</sub> T <sub>w</sub> T <sub>3</sub>	Next op-code Address (PC)	1st op-code	1	1	1	0	0	1
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	PC	n	0	1	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	PC+1	m	0	1	0	1	1	1	1
	MC <sub>4</sub>	Ti	*	Z	1	1	1	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-1	PC+2(H)	1	0	0	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-2	PC+2(L)	1	0	0	1	1	1	1
$\overline{INT_0}$ MODE 1	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>w</sub> T <sub>w</sub> T <sub>3</sub>	Next op-code Address (PC)		1	1	1	0	0	1	0
	MC <sub>2</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-1	PCH	1	0	0	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-2	PCL	1	0	0	1	1	1	1
$\overline{INT_0}$ MODE 2	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>w</sub> T <sub>w</sub> T <sub>3</sub>	Next op-code Address (PC)	Vector	1	1	1	0	0	1	0
	MC <sub>2</sub>	Ti	*	Z	1	1	1	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-1	PCH	1	0	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-2	PCL	1	0	0	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	I, vector	DATA	0	1	0	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	I, vector + 1	DATA	0	1	0	1	1	1	1

(Continued)

Instruction	Machine Cycle	States	ADDRESS	DATA	$\overline{RD}$	$\overline{WR}$	$\overline{ME}$	$\overline{OE}$	$\overline{LIR}$	$\overline{HALT}$	ST
$\overline{INT_1}$ $\overline{INT_2}$ Internal interrupts	MC <sub>1</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>W</sub> T <sub>W</sub> T <sub>3</sub>	Next op-code Address (PC)		1	1	1	1	1	1	0
	MC <sub>2</sub>	T <sub>i</sub>	*	Z	1	1	1	1	1	1	1
	MC <sub>3</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-1	PCH	1	0	0	1	1	1	1
	MC <sub>4</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	SP-2	PCL	1	0	0	1	1	1	1
	MC <sub>5</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	I, vector	DATA	0	1	0	1	1	1	1
	MC <sub>6</sub>	T <sub>1</sub> T <sub>2</sub> T <sub>3</sub>	I, vector + 1	DATA	0	1	0	1	1	1	1

## E-1. Requests in Each Operating Mode

Current Status		Operation Requests		
Chip Operation Mode	Operation Cycle	Interrupt Requests		
		WAIT	NMI	INT0-INT2, or Internal Interrupt
Normal operation mode	CPU	Accepted	Accepted at end of instruction	Accepted at end of instruction
	Interrupt acknowledge cycle	Accepted	Not accepted	Not accepted
	DMA	Accepted	Accepted; DMA cycle aborted	Accepted
	Refresh	Accepted *1	Accepted *3	Accepted *3
	Bus release mode	Not accepted	Accepted *2	Accepted *2
Halt mode	DMA	Accepted	Accepted; DMA cycle aborted and halt mode released	Accepted
	Refresh	Accepted *1	Accepted; halt mode released after completion of refresh cycle*3	Accepted; halt mode released after completion of refresh cycle*3
	Bus release mode	Not accepted	Accepted; halt mode released after completion of bus release mode*2	Accepted; halt mode released after completion of bus release mode*2
	Other halt mode	Accepted	Accepted; halt mode released	Accepted; halt mode released
Sleep mode	DMA	Accepted	Accepted; DMA cycle aborted and sleep mode released	Accepted
	Refresh	Accepted *1	Accepted; sleep mode released after completion of refresh cycle*3	Accepted; sleep mode released after completion of refresh cycle*3
	Bus release mode	Not accepted	Accepted; sleep mode released after completion of bus release mode*2	Accepted; sleep mode released after completion of bus release mode*2
	Other sleep mode	Not accepted	Accepted; sleep mode released	Accepted; sleep mode released



## E-1. Requests in Each Operating Mode (cont.)

Current Status		Operation Requests		
Chip Operation Mode	Operation Cycle	Interrupt Requests		
		WAIT	NMI	INT0-INT2, or Internal Interrupt
System stop mode	Bus release mode	Not accepted	Accepted; system stop mode released after completion of bus release mode*2	Accepted; system stop mode released after completion of bus release mode*2
	Other system stop mode	Not accepted	Accepted; system stop mode released	Accepted; system stop mode released
Reset mode	-----	Not accepted	Not accepted	Not accepted

## E-1. Requests in Each Operating Mode (cont.)

Current Status		Operation Requests		
Chip Operation	Operation Cycle	Bus Requests		
Mode		BUSREQ	Refresh Request	DMA Request from <u>DREQ0</u> , <u>DREQ1</u> , or MSCI
Normal operation mode	CPU	Accepted; enters bus release mode at end of machine cycle	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle
	Interrupt acknowledge cycle	Accepted; enters bus release mode at end of machine cycle	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle
	DMA	Accepted; enters bus release mode at end of machine cycle	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle
	Refresh	Accepted; enters bus release mode at end of machine cycle *3	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle *3
	Bus release mode	Bus release mode continues	Accepted; refresh cycle executed after completion of bus release mode*2	Accepted; DMA cycle executed after completion of bus release mode *2
Halt mode	DMA	Accepted; bus release mode entered at end of machine cycle	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle
	Refresh	Accepted; bus release mode entered at end of machine cycle *3	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle *3
	Bus release mode	Bus release mode continues	Accepted; refresh cycle executed after completion of bus release mode*2	Accepted; DMA cycle executed after completion of bus release mode *2
	Other halt mode	Accepted; bus release mode entered at end of machine cycle	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle

## E-1. Requests in Each Operating Mode (cont.)

Current Status		Operation Requests		
Chip Operation Mode	Operation Cycle	Bus Requests		
		<u>BUSREQ</u>	Refresh Request	DMA Request from <u>DREQ0, DREQ1, or MSCI</u>
	DMA	Accepted; bus release mode entered at end of machine cycle	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle
	Refresh	Accepted; bus release mode entered at end of machine cycle *3	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle *3
Sleep mode	Bus release mode	Bus release mode continues	Accepted; refresh cycle executed after completion of bus release mode*2	Accepted; DMA cycle executed after completion of bus release mode *2
	Other sleep mode	Accepted; enters bus release mode	Accepted; refresh cycle executed at end of machine cycle	Accepted; DMA cycle executed at end of machine cycle
System stop mode	Bus release mode	Bus release mode continues	Not accepted	Not accepted
	Other system stop mode	Accepted; enters bus release mode	Not accepted	Not accepted
Reset mode	-----	Not accepted	Not accepted	Not accepted

\*1 Not accepted when the number of programmable wait states is 0.

\*2 Requests are held until the bus release mode completes.

\*3 Requests are held until the refresh cycle completes.

## E-2. Request Priorities

Requests to the HD64180S are categorized into three types:

- ① Requests accepted and executed in each state .....  $\overline{\text{WAIT}}$
- ② Requests accepted and executed in each machine cycle ..... Refresh request  
DMA request  
 $\overline{\text{BUSREQ}}$  request
- ③ Requests accepted and executed in each instruction ..... Interrupts

In principle, request priorities are as follows:

(High) ① > ② > ③ (Low)

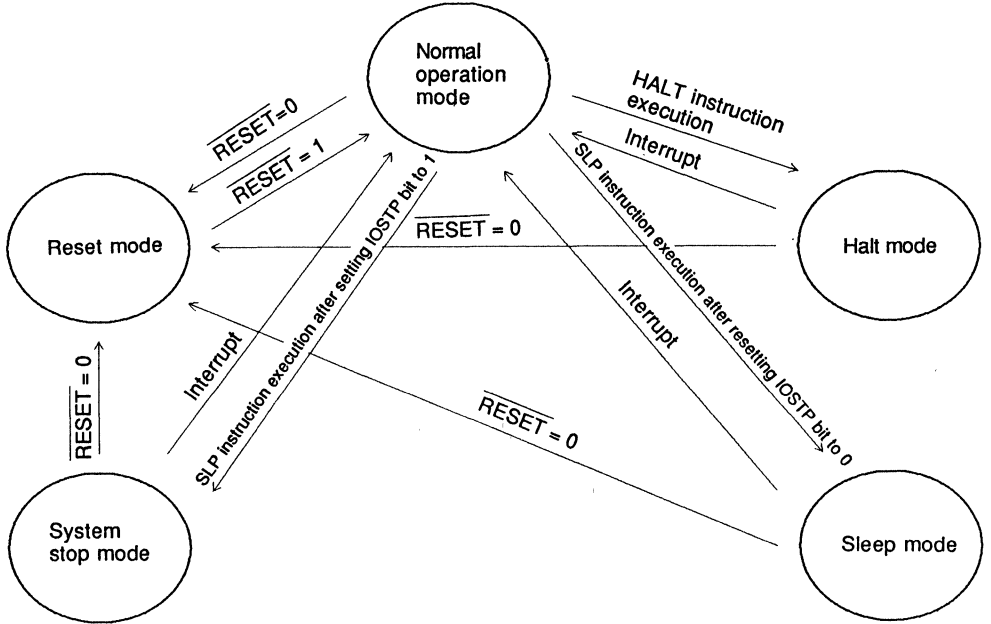
Type ② requests are prioritized as follows:

(High)  $\overline{\text{BUSREQ}}$  > Refresh request > DMA request (Low)

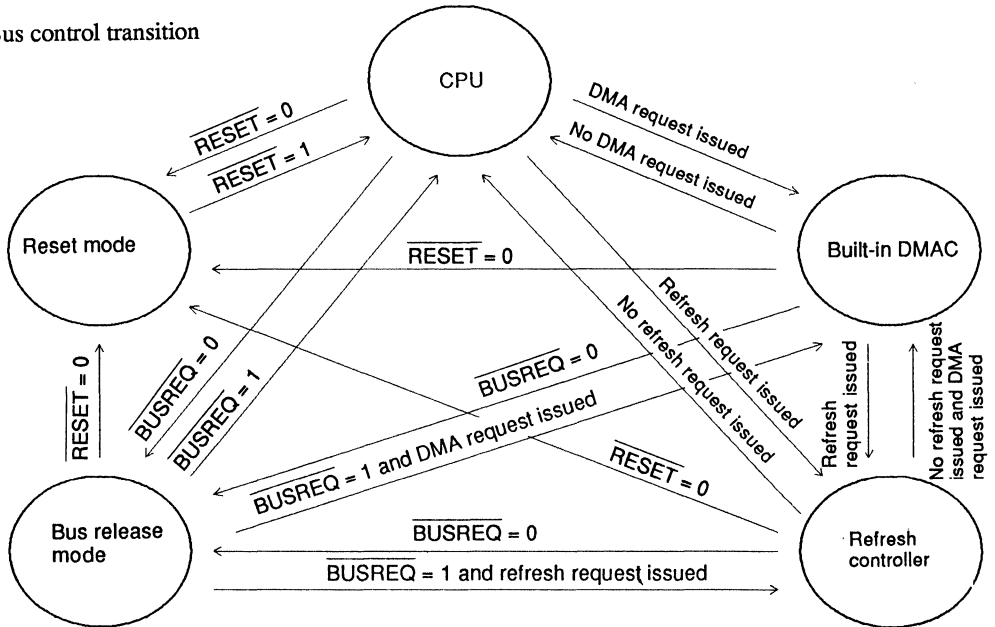
For the priority of type ③ requests, see section 3.6 "Interrupts"

### E-3. State Transition Diagrams

(1) Chip operation mode transition diagram



(2) Bus control transition



## F-1. Status Signals

Status signals are listed below.

Chip operation mode	Operation cycle	$\overline{\text{LTR}}$	$\overline{\text{ME}}$	$\overline{\text{TOE}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{REF}}$	$\overline{\text{HALT}}$	$\overline{\text{BUSACK}}$	ST	$\overline{\text{CS}}_{0\sim 2}$	$\text{A}_0\sim\text{A}_9$	$\text{D}_0\sim\text{D}_7$	
Normal operation mode	CPU	First opcode fetch	0	0	1	0	1	1	1	1	0	OUT (A)	OUT (A)	IN
		Second and third opcode fetch	0	0	1	0	1	1	1	1	1	OUT (A)	OUT (A)	IN
		Memory read	1	0	1	0	1	1	1	1	1	OUT (A)	OUT (A)	IN
		Memory write	1	0	1	1	0	1	1	1	1	OUT (A)	OUT (A)	OUT (A)
		I/O read	1	1	0	0	1	1	1	1	1	1	OUT (A)	IN
		I/O write	1	1	0	1	0	1	1	1	1	1	OUT (A)	OUT (A)
		Internal operation	1	1	1	1	1	1	1	1	1	1	OUT (A)	Z
	Interrupt acknowledge (first machine cycle)	$\overline{\text{NMI}}$	0	0	1	0	1	1	1	1	0	OUT (A)	OUT (A)	IN
		$\overline{\text{INT}}_0$	0	1	0	1	1	1	1	1	0	1	OUT (A)	IN
		$\overline{\text{INT}}_1, \overline{\text{INT}}_2$ , and internal interrupts	1	1	1	1	1	1	1	1	0	1	OUT (A)	Z
	Internal DMA	Memory read	1	0	1	0	1	1	1	1	0	OUT (A)	OUT (A)	IN
		Memory write	1	0	1	1	0	1	1	1	0	OUT (A)	OUT (A)	OUT (A)
		I/O read	1	1	0	0	1	1	1	1	0	1	OUT (A)	IN
		I/O write	1	1	0	1	0	1	1	1	0	1	OUT (A)	OUT (A)
		Internal operation	1	1	1	1	1	1	1	1	0	1	OUT (A)	Z
		Refresh	1	0	1	1	1	0	1	1	1	1	OUT (A)	Z
		Bus release mode	1	Z	Z	Z	Z	1	1	0	1	1	Z	Z

1: High level output  
 0: Low level output  
 OUT (A): Any output  
 IN: Input  
 Z: High impedance

F-1. Status Signals (cont.)

Chip operation mode	Operation cycle		$\overline{\text{LTR}}$	$\overline{\text{ME}}$	$\overline{\text{IOE}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{REF}}$	$\overline{\text{HALT}}$	$\overline{\text{BUSACK}}$	ST	$\overline{\text{CS}}_{0\sim 2}$	$\text{A}_0\sim\text{A}_{19}$	$\text{D}_0\sim\text{D}_7$
Halt mode	Internal DMA	Memory read	1	0	1	0	1	1	0	1	0	OUT (A)	OUT (A)	IN
		Memory write	1	0	1	1	0	1	0	1	0	OUT (A)	OUT (A)	OUT (A)
		I/O read	1	1	0	0	1	1	0	1	0	1	OUT (A)	IN
		I/O write	1	1	0	1	0	1	0	1	0	1	OUT (A)	OUT (A)
		Internal operation	1	1	1	1	1	1	0	1	0	1	OUT (A)	Z
	Refresh	1	0	1	1	1	0	0	1	1	1	OUT (A)	Z	
	Bus release mode	1	Z	Z	Z	Z	1	0	0	1	1	Z	Z	
	Halt mode other than above	0	0	1	0	1	1	0	1	0	OUT (A)	OUT (A)	IN	
Sleep mode	Internal DMA	Memory read	1	0	1	0	1	1	0	1	0	OUT (A)	OUT (A)	IN
		Memory write	1	0	1	1	0	1	0	1	0	OUT (A)	OUT (A)	OUT (A)
		I/O read	1	1	0	0	1	1	0	1	0	1	OUT (A)	IN
		I/O write	1	1	0	1	0	1	0	1	0	1	OUT (A)	OUT (A)
		Internal operation	1	1	1	1	1	1	0	1	0	1	OUT (A)	Z
	Refresh	1	0	1	1	1	0	0	1	1	1	OUT (A)	Z	
	Bus release mode	1	Z	Z	Z	Z	1	0	0	1	1	Z	Z	
	Sleep mode other than above	1	1	1	1	1	1	0	1	1	1	1	Z	
System stop mode	Bus release mode		1	Z	Z	Z	Z	1	0	0	1	1	Z	Z
	System stop mode other than above		1	1	1	1	1	1	0	1	1	1	1	Z
Reset mode	—————		1	1	1	1	1	1	1	1	1	1	Z	Z

1: High level output  
0: Low level output  
OUT (A): Any output  
IN: Input  
Z: High impedance

## F-2. Pin States in Reset and Low Power Dissipation Modes

Pin name	Pin state		
	Reset mode	Sleep mode	System stop mode
$\overline{TIN}_0, \overline{TIN}_1$	IN (N)	IN (A)	IN (N)
$\overline{TOUT}_0, \overline{TOUT}_1$	OUT (L)	OUT (A)	HOLD
$\overline{CS}_0, \overline{CS}_1, \overline{CS}_2$	OUT (H)	OUT (A)	OUT (H)
$\overline{WAIT}$	IN (N)	IN (A)	IN (N)
$\overline{NMI}$	IN (N)	IN (A)	IN (A)
$\overline{INT}_0, \overline{INT}_1, \overline{INT}_2$	IN (N)	IN (A)	IN (A)
$\overline{RESET}$	IN (A)	IN (A)	IN (A)
$\overline{BUSREQ}$	IN (N)	IN (A)	IN (A)
$\overline{BUSACK}$	OUT (H)	OUT (A)	OUT (A)
ST	OUT (H)	OUT (A)	OUT (H)
$\overline{LIR}$	OUT (H)	OUT (H)	OUT (H)
$\overline{REF}$	OUT (H)	OUT (A)	OUT (H)
$\overline{HALT}$	OUT (H)	OUT (L)	OUT (L)
$\overline{RD}$	OUT (H)	OUT (A)	OUT (H)
$\overline{WR}$	OUT (H)	OUT (A)	OUT (H)
$\overline{ME}$	OUT (H)	OUT (A)	OUT (H)
$\overline{IOE}$	OUT (H)	OUT (A)	OUT (H)
$A_0 \sim A_{19}$	Z	OUT (A)	OUT (H)
$D_0 \sim D_7$	Z	IN (A), OUT (A), Z	Z
$\overline{SYNC}$	Input selected	IN (N)	IN (N)
	Output selected	—	HOLD
RTSM	OUT (H)	OUT (A)	HOLD
DCDM	IN (N)	IN (A)	IN (N)
CTSM	IN (N)	IN (A)	IN (N)
RXDM	IN (N)	IN (A)	IN (N)

IN (A): Input (active)  
 IN (N): Input (inactive)  
 OUT (H): Output (fixed to high level)  
 OUT (L): Output (fixed to low level)  
 OUT (A): Output (active) - High or low level output  
 Z: High impedance  
 HOLD: Holding the previous state



## F-2. Pin States in Reset and Low Power Dissipation Modes (cont.)

Pin name		Pin state		
		Reset mode	Sleep mode	System stop mode
RXCM	Input selected	I N (N)	I N (Λ)	I N (N)
	Output selected	————	O U T (A)	O U T (A)
TXCM	Input selected	I N (N)	I N (A)	I N (N)
	Output selected	————	O U T (A)	O U T (A)
TXDM		O U T (H)	O U T (A)	O U T (H)
$\overline{\text{RTSA}}$		O U T (H)	O U T (A)	H O L D
$\overline{\text{DCDA}}$		I N (N)	I N (A)	I N (N)
$\overline{\text{CTSA}}$		I N (N)	I N (Λ)	I N (N)
RXDA		I N (N)	I N (A)	I N (N)
RXCA	Input selected	I N (N)	I N (A)	I N (N)
	Output selected	————	O U T (A)	O U T (H)
TXCA	Input selected	I N (N)	I N (A)	I N (N)
	Output selected	————	O U T (A)	O U T (H)
TXDA		O U T (H)	O U T (A)	H O L D
$\overline{\text{DREQ}}_0, \overline{\text{DREQ}}_1$		I N (N)	I N (A)	I N (N)
$\overline{\text{TEND}}_0, \overline{\text{TEND}}_1$		O U T (H)	O U T (A)	O U T (H)
$\phi$		$\phi$ clock output	$\phi$ clock output	$\phi$ clock output

IN (A): Input (active)  
 IN (N): Input (inactive)  
 OUT (H): Output (fixed to high level)  
 OUT (L): Output (fixed to low level)  
 OUT (A): Output (active) - High or low level output  
 Z: High impedance  
 HOLD: Holding the previous state

# G. Built-in Registers

## CPU

Register	Address	Remarks																																				
Interrupt control register (ICR)	0000H	<table border="1"> <thead> <tr> <th>Bit Name</th> <th>7</th> <th>6</th> <th>5</th> <th>4</th> <th>3</th> <th>2</th> <th>1</th> <th>0</th> </tr> </thead> <tbody> <tr> <td></td> <td>TRAP</td> <td>UFO</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>ITE0</td> </tr> <tr> <td>Read/Write</td> <td>RW</td> <td>R</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>RW</td> </tr> <tr> <td>Initial Value</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> </tr> </tbody> </table> <p> <b>TRAP Status</b>            0: TRAP interrupt not generated            1: TRAP interrupt generated         </p> <p> <b>Undefined Fetch Object Code</b>            0: Second byte of opcode undefined            1: Third byte of opcode undefined         </p> <p> <b>INT0 Enable</b>            0: INT0 disabled            1: INT0 enabled         </p>	Bit Name	7	6	5	4	3	2	1	0		TRAP	UFO	-	-	-	-	-	ITE0	Read/Write	RW	R	-	-	-	-	-	RW	Initial Value	0	0	0	0	0	0	0	1
Bit Name	7	6	5	4	3	2	1	0																														
	TRAP	UFO	-	-	-	-	-	ITE0																														
Read/Write	RW	R	-	-	-	-	-	RW																														
Initial Value	0	0	0	0	0	0	0	1																														
MMU common base register (CBR)	0001H	<table border="1"> <thead> <tr> <th>Bit Name</th> <th>7</th> <th>6</th> <th>5</th> <th>4</th> <th>3</th> <th>2</th> <th>1</th> <th>0</th> </tr> </thead> <tbody> <tr> <td></td> <td>CB7</td> <td>CB6</td> <td>CB5</td> <td>CB4</td> <td>CB3</td> <td>CB2</td> <td>CB1</td> <td>CB0</td> </tr> <tr> <td>Read/Write</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> </tr> <tr> <td>Initial Value</td> <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> </tbody> </table>	Bit Name	7	6	5	4	3	2	1	0		CB7	CB6	CB5	CB4	CB3	CB2	CB1	CB0	Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	Initial Value	0	0	0	0	0	0	0	0
Bit Name	7	6	5	4	3	2	1	0																														
	CB7	CB6	CB5	CB4	CB3	CB2	CB1	CB0																														
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW																														
Initial Value	0	0	0	0	0	0	0	0																														
MMU bank base register (BBR)	0002H	<table border="1"> <thead> <tr> <th>Bit Name</th> <th>7</th> <th>6</th> <th>5</th> <th>4</th> <th>3</th> <th>2</th> <th>1</th> <th>0</th> </tr> </thead> <tbody> <tr> <td></td> <td>BB7</td> <td>BB6</td> <td>BB5</td> <td>BB4</td> <td>BB3</td> <td>BB2</td> <td>BB1</td> <td>BB0</td> </tr> <tr> <td>Read/Write</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> </tr> <tr> <td>Initial Value</td> <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> </tbody> </table>	Bit Name	7	6	5	4	3	2	1	0		BB7	BB6	BB5	BB4	BB3	BB2	BB1	BB0	Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	Initial Value	0	0	0	0	0	0	0	0
Bit Name	7	6	5	4	3	2	1	0																														
	BB7	BB6	BB5	BB4	BB3	BB2	BB1	BB0																														
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW																														
Initial Value	0	0	0	0	0	0	0	0																														
MMU common/bank area register (CBAR)	0003H	<table border="1"> <thead> <tr> <th>Bit Name</th> <th>7</th> <th>6</th> <th>5</th> <th>4</th> <th>3</th> <th>2</th> <th>1</th> <th>0</th> </tr> </thead> <tbody> <tr> <td></td> <td>CA3</td> <td>CA2</td> <td>CA1</td> <td>CA0</td> <td>BA3</td> <td>BA2</td> <td>BA1</td> <td>BA0</td> </tr> <tr> <td>Read/Write</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> </tr> <tr> <td>Initial Value</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> </tbody> </table> <p>           Four high order bits of the lower address limit for common area 1         </p> <p>           Four high order bits of the lower address limit for the bank area         </p>	Bit Name	7	6	5	4	3	2	1	0		CA3	CA2	CA1	CA0	BA3	BA2	BA1	BA0	Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	Initial Value	1	1	1	1	0	0	0	0
Bit Name	7	6	5	4	3	2	1	0																														
	CA3	CA2	CA1	CA0	BA3	BA2	BA1	BA0																														
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW																														
Initial Value	1	1	1	1	0	0	0	0																														
Operation mode control register (OMCR)	0004H	<table border="1"> <thead> <tr> <th>Bit Name</th> <th>7</th> <th>6</th> <th>5</th> <th>4</th> <th>3</th> <th>2</th> <th>1</th> <th>0</th> </tr> </thead> <tbody> <tr> <td></td> <td>LIRE</td> <td>LIRTE</td> <td>IOC</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> </tr> <tr> <td>Read/Write</td> <td>RW</td> <td>W</td> <td>RW</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> </tr> <tr> <td>Initial Value</td> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table> <p> <b>LIR Enable</b>            0: The LIR output is low only during the opcode fetch cycle 2 of the RETI instruction and the first machine cycle of the INT0 interrupt acknowledge cycle.            1: Normal operation         </p> <p> <b>LIR Temporary Enable</b>            0: When the LIRE bit is 0, the LIR output is low only for the opcode fetch cycle immediately after 0 is written to the LIRTE bit.            1: Normal operation         </p> <p> <b>IO Compatibility</b>            0: Output of the IOC and RD lines is compatible with that of the Z80-based peripheral LSIs.            1: Normal operation         </p>	Bit Name	7	6	5	4	3	2	1	0		LIRE	LIRTE	IOC	-	-	-	-	-	Read/Write	RW	W	RW	-	-	-	-	-	Initial Value	1	1	1	0	0	0	0	0
Bit Name	7	6	5	4	3	2	1	0																														
	LIRE	LIRTE	IOC	-	-	-	-	-																														
Read/Write	RW	W	RW	-	-	-	-	-																														
Initial Value	1	1	1	0	0	0	0	0																														

## G. Built-in Registers (cont.)

### CPU

Register	Address	Remarks																																				
I/O control register (IOCR)	0005H	<table border="1"> <tr> <td></td> <td>7</td> <td>6</td> <td>5</td> <td>4</td> <td>3</td> <td>2</td> <td>1</td> <td>0</td> </tr> <tr> <td>Bit Name</td> <td>IOSTP</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> </tr> <tr> <td>Read/Write</td> <td>RW</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> </tr> <tr> <td>Initial Value</td> <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> <p>IO Stop 0: Sleep mode (SLP instruction execution) 1: System stop mode (SLP instruction execution)</p>		7	6	5	4	3	2	1	0	Bit Name	IOSTP	-	-	-	-	-	-	-	Read/Write	RW	-	-	-	-	-	-	-	Initial Value	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0																														
Bit Name	IOSTP	-	-	-	-	-	-	-																														
Read/Write	RW	-	-	-	-	-	-	-																														
Initial Value	0	0	0	0	0	0	0	0																														
Unused	0006H																																					
Unused	0007H																																					

### Wait Control

Register	Address	Remarks																																																																								
Physical address boundary register 0 (PABR0)	0008H	<table border="1"> <tr> <td></td> <td>7</td> <td>6</td> <td>5</td> <td>4</td> <td>3</td> <td>2</td> <td>1</td> <td>0</td> </tr> <tr> <td>Bit Name</td> <td>PB07</td> <td>PB06</td> <td>PB05</td> <td>PB04</td> <td>PB03</td> <td>PB02</td> <td>PB01</td> <td>PB00</td> </tr> <tr> <td>Read/Write</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> </tr> <tr> <td>Initial Value</td> <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> <p>PAL/PAM Boundary Address (8 high-order bits)</p>		7	6	5	4	3	2	1	0	Bit Name	PB07	PB06	PB05	PB04	PB03	PB02	PB01	PB00	Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	Initial Value	0	0	0	0	0	0	0	0																																				
	7	6	5	4	3	2	1	0																																																																		
Bit Name	PB07	PB06	PB05	PB04	PB03	PB02	PB01	PB00																																																																		
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW																																																																		
Initial Value	0	0	0	0	0	0	0	0																																																																		
Physical address boundary register 1 (PABR1)	0009H	<table border="1"> <tr> <td></td> <td>7</td> <td>6</td> <td>5</td> <td>4</td> <td>3</td> <td>2</td> <td>1</td> <td>0</td> </tr> <tr> <td>Bit Name</td> <td>PB17</td> <td>PB16</td> <td>PB15</td> <td>PB14</td> <td>PB13</td> <td>PB12</td> <td>PB11</td> <td>PB10</td> </tr> <tr> <td>Read/Write</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> </tr> <tr> <td>Initial Value</td> <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> <p>PAMPAM Boundary Address (8 high-order bits)</p>		7	6	5	4	3	2	1	0	Bit Name	PB17	PB16	PB15	PB14	PB13	PB12	PB11	PB10	Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	Initial Value	0	0	0	0	0	0	0	0																																				
	7	6	5	4	3	2	1	0																																																																		
Bit Name	PB17	PB16	PB15	PB14	PB13	PB12	PB11	PB10																																																																		
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW																																																																		
Initial Value	0	0	0	0	0	0	0	0																																																																		
Wait control register L (WCRL)	000AH	<table border="1"> <tr> <td></td> <td>7</td> <td>6</td> <td>5</td> <td>4</td> <td>3</td> <td>2</td> <td>1</td> <td>0</td> </tr> <tr> <td>Bit Name</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>PALW2</td> <td>PALW1</td> <td>PALW0</td> </tr> <tr> <td>Read/Write</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>RW</td> <td>RW</td> <td>RW</td> </tr> <tr> <td>Initial Value</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> </tr> </table> <p>PAL Area Wait</p> <table border="1"> <thead> <tr> <th>PALW2</th> <th>PALW1</th> <th>PALW0</th> <th>Number of Wait States</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>2</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>3</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>4</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>5</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>6</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>7</td> </tr> </tbody> </table>		7	6	5	4	3	2	1	0	Bit Name	-	-	-	-	-	PALW2	PALW1	PALW0	Read/Write	-	-	-	-	-	RW	RW	RW	Initial Value	0	0	0	0	0	1	1	1	PALW2	PALW1	PALW0	Number of Wait States	0	0	0	0	0	0	1	1	0	1	0	2	0	1	1	3	1	0	0	4	1	0	1	5	1	1	0	6	1	1	1	7
	7	6	5	4	3	2	1	0																																																																		
Bit Name	-	-	-	-	-	PALW2	PALW1	PALW0																																																																		
Read/Write	-	-	-	-	-	RW	RW	RW																																																																		
Initial Value	0	0	0	0	0	1	1	1																																																																		
PALW2	PALW1	PALW0	Number of Wait States																																																																							
0	0	0	0																																																																							
0	0	1	1																																																																							
0	1	0	2																																																																							
0	1	1	3																																																																							
1	0	0	4																																																																							
1	0	1	5																																																																							
1	1	0	6																																																																							
1	1	1	7																																																																							

## G. Built-in Registers (cont.)

### Wait Control

Register	Address	Remarks
----------	---------	---------

Wait control register M (WCRM)000BH

Bit Name	7	6	5	4	3	2	1	0
	-	-	-	-	-	PAMW2	PAMW1	PAMW0
Read/Write	-	-	-	-	-	RW	RW	RW
Initial Value	0	0	0	0	0	1	1	1

PAM Area Wait

PAMW2	PAMW1	PAMW0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Wait control register H (WCRH) 000CH

Bit Name	7	6	5	4	3	2	1	0
	-	-	-	-	-	PAHW2	PAHW1	PAHW0
Read/Write	-	-	-	-	-	RW	RW	RW
Initial Value	0	0	0	0	0	1	1	1

PAH Area Wait

PAHW2	PAHW1	PAHW0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

I/O wait control register  
(IOWCR)

000DH

Bit Name	7	6	5	4	3	2	1	0
	-	IOH2	IOH1	IOH0	-	IOL2	IOL1	IOL0
Read/Write	-	RW	RW	RW	-	RW	RW	RW
Initial Value	0	1	1	1	0	1	1	1

IO High

IO Low

IOH2	IOH1	IOH0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

IOL2	IOL1	IOL0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

## G. Built-in Registers (cont.)

### Wait Control

Register	Address	Remarks
----------	---------	---------

Interrupt wait control register (INTWR) 000EH

Bit Name	7	6	5	4	3	2	1	0
Read/Write	-	-	-	-	-	RW	RW	RW
Initial Value	0	0	0	0	0	1	1	1

Interrupt Wait

INTW2	INTW1	INTW0	Number of Wait States
0	0	0	2
0	0	1	3
0	1	0	4
0	1	1	5
1	0	0	6
1	0	1	7
1	1	0	8
1	1	1	9

Refresh wait control register (RWCR) 000FH

Bit Name	7	6	5	4	3	2	1	0
Read/Write	-	-	-	-	-	RW	RW	RW
Initial Value	0	0	0	0	0	1	1	1

Refresh Wait

REFW2	REFW1	REFW0	Number of Wait States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

### Interrupt Control

Register	Address	Remarks
----------	---------	---------

Interrupt status register 0 (ISR0) 0010H

Bit Name	7	6	5	4	3	2	1	0
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	X	X

ASCI0SIO.DRDY

0: Request not issued  
1: Request issued

MSCI0.INT

0: Request not issued  
1: Request issued

MSCI0.DRDY

0: Request not issued  
1: Request issued

External Interrupt INT2

0: Request not issued  
1: Request issued

ASCI0SIO.RDY

0: Request not issued  
1: Request issued

MSCI0.RINT

0: Request not issued  
1: Request issued

MSCI0.RRDY

0: Request not issued  
1: Request issued

External Interrupt INT1

0: Request not issued  
1: Request issued

## G. Built-in Registers (cont.)

### Interrupt Control

**Register** **Address** **Remarks**

Interrupt status register 1 (ISR1) 0011H

Bit Name	7	6	5	4	3	2	1	0
	T1IRQ	TOIRQ	DMIB1	DMIA1	DMIB0	DMIA0	TXINT1	RXINT1
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

Timer Channel 1 Interrupt Request  
 0: Request not issued  
 1: Request issued

DMA Interrupt B Channel 1  
 0: Request not issued  
 1: Request issued

DMA Interrupt B Channel 0  
 0: Request not issued  
 1: Request issued

ASCI/CSIO TXINT  
 0: Request not issued  
 1: Request issued

Timer Channel 0 Interrupt Request  
 0: Request not issued  
 1: Request issued

DMA Interrupt A Channel 1  
 0: Request not issued  
 1: Request issued

DMA Interrupt A Channel 0  
 0: Request not issued  
 1: Request issued

ASCI/CSIO RXINT  
 0: Request not issued  
 1: Request issued

Interrupt enable register 0 (IER0)

0012H

Bit Name	7	6	5	4	3	2	1	0
	TXRDYIE	RXRDYIE	TXINTIE	RXINTIE	TXRDYOE	RXRDYOE	INT2E	INT1E
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

ASCI/CSIO TXRDY Enable  
 0: Disabled  
 1: Enabled

MSCI TXINT Enable  
 0: Disabled  
 1: Enabled

MSCI TXRDY Enable  
 0: Disabled  
 1: Enabled

INT2 External Interrupt Enable  
 0: Disabled  
 1: Enabled

ASCI/CSIO RXRDY Enable  
 0: Disabled  
 1: Enabled

MSCI RXINT Enable  
 0: Disabled  
 1: Enabled

MSCI RXRDY Enable  
 0: Disabled  
 1: Enabled

INT1 External Interrupt Enable  
 0: Disabled  
 1: Enabled

Interrupt enable register 1 (IER1)

0013H

Bit Name	7	6	5	4	3	2	1	0
	T1IRQIE	TOIRQIE	DMIB1E	DMIA1E	DMIB0E	DMIA0E	TXINT1E	RXINT1E
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Timer Channel 1 Interrupt Request Enable  
 0: Disabled  
 1: Enabled

DMA Interrupt B Channel 1 Enable  
 0: Disabled  
 1: Enabled

DMA Interrupt B Channel 0 Enable  
 0: Disabled  
 1: Enabled

DMA Interrupt A Channel 1 Enable  
 0: Disabled  
 1: Enabled

DMA Interrupt A Channel 0 Enable  
 0: Disabled  
 1: Enabled

Timer Channel 0 Interrupt Request Enable  
 0: Disabled  
 1: Enabled

ASCI/CSIO TXINT Enable  
 0: Disabled  
 1: Enabled

Interrupt vector low register (IL) 0014H

Bit Name	7	6	5	4	3	2	1	0
	IL7	IL6	-	-	-	-	-	-
Read/Write	R/W	R/W	-	-	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0

Fixed code

Low order byte of vector address

## G. Built-in Registers (cont.)

### Interrupt Control

Register	Address	Remarks
Unused	0015H	
Unused	0016H	
Unused	0017H	

### Refresh Control

Register	Address	Remarks																																				
Refresh control register (RCR)	0018H	<table border="1"> <thead> <tr> <th>Bit Name</th> <th>7</th> <th>6</th> <th>5</th> <th>4</th> <th>3</th> <th>2</th> <th>1</th> <th>0</th> </tr> </thead> <tbody> <tr> <td>REFE</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>CYC2</td> <td>CYC1</td> <td>CYC0</td> </tr> <tr> <td>Read/Write</td> <td>RW</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>RW</td> <td>RW</td> <td>RW</td> </tr> <tr> <td>Initial Value</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table> <p>Refresh Enable 0: Refresh cycles not inserted 1: Refresh cycles inserted</p> <p>Cycle Select - Insertion Interval 000: 32 states 001: 64 states 010: 96 states 011: 128 states 100: 160 states 101: 192 states 110: 224 states 111: 256 states</p>	Bit Name	7	6	5	4	3	2	1	0	REFE	-	-	-	-	-	CYC2	CYC1	CYC0	Read/Write	RW	-	-	-	-	RW	RW	RW	Initial Value	1	0	0	0	0	0	0	0
Bit Name	7	6	5	4	3	2	1	0																														
REFE	-	-	-	-	-	CYC2	CYC1	CYC0																														
Read/Write	RW	-	-	-	-	RW	RW	RW																														
Initial Value	1	0	0	0	0	0	0	0																														
Unused	0019H																																					
Unused	001AH																																					
Unused	001BH																																					

### Bus Control

Register	Address	Remarks																																																						
DMA priority control register (PCR)	001CH	<table border="1"> <thead> <tr> <th></th> <th>7</th> <th>6</th> <th>5</th> <th>4</th> <th>3</th> <th>2</th> <th>1</th> <th>0</th> </tr> </thead> <tbody> <tr> <td>Single-block Transfer Mode (dual address)</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Single-block Transfer Mode (single address)</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>PR0</td> </tr> <tr> <td>Chained-block Transfer Mode</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Read/Write</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>RW</td> </tr> <tr> <td>Initial Value</td> <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> </tbody> </table> <p>Channel Priority 0: Channel 0 has priority over channel 1 1: Channel 1 has priority over channel 0</p>		7	6	5	4	3	2	1	0	Single-block Transfer Mode (dual address)									Single-block Transfer Mode (single address)	-	-	-	-	-	-	-	PR0	Chained-block Transfer Mode									Read/Write	-	-	-	-	-	-	-	RW	Initial Value	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0																																																
Single-block Transfer Mode (dual address)																																																								
Single-block Transfer Mode (single address)	-	-	-	-	-	-	-	PR0																																																
Chained-block Transfer Mode																																																								
Read/Write	-	-	-	-	-	-	-	RW																																																
Initial Value	0	0	0	0	0	0	0	0																																																

DMA master enable register (DMER) 001DH

	7	6	5	4	3	2	1	0
Single-block Transfer Mode (dual address)								
Single-block Transfer Mode (single address)	DME	-	-	-	-	-	-	-
Chained-block Transfer Mode								
Read/Write	RW	-	-	-	-	-	-	-
Initial Value	1	0	0	0	0	0	0	0

DMA Master-Enable  
0: Disable  
1: Enable

## G. Built-in Registers (cont.)

### Bus Control

Register	Address	Remarks
Unused	001EH	
Unused	001FH	

### MSCI

Register	Address	Remarks
MSCI TX/RX buffer register (MTRB)	0020H	

Async	7	6	5	4	3	2	1	0
Byte Sync	TRB7	TRB6	TRB5	TRB4	TRB3	TRB2	TRB1	TRB0
Bit Sync								
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	X	X	X	X	X	X	X	X

Value written to, or read from, the transmit/receive buffer

MSCI status register 0 (MST0) 0021H

Async	7	6	5	4	3	2	1	0
Byte Sync	TXINT	RXINT	-	-	-	-	TXRDY	RXRDY
Bit Sync								
Read/Write	R	R	-	-	-	-	R	R
Initial Value	0	0	0	0	0	0	0	0

**TXINT interrupt**  
 0: No interrupt  
 1: Interrupt

**RXINT interrupt**  
 0: No interrupt  
 1: Interrupt

**TX ready**  
 0: Transmit buffer full  
 1: Transmit buffer empty/not full

**RX ready**  
 0: No receive data  
 1: Receive data

MSCI status register 1 (MST1) 0022H

Async	7	6	5	4	3	2	1	0
Byte Sync	UDRN	IDL	-	SYNCD	CCTS	CCDD	BRKD	BRKE
MDLC				FLGD			ABTD/	IDLD
Loop							GAPO	
Read/Write	RW	R	-	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0

**Underrun Error**  
 - Byte/Bit synchronous mode  
 0: No underrun detected  
 1: Underrun detected

**SYN Pattern Detection**  
 - Byte synchronous mode  
 0: No pattern detected  
 1: Pattern detected

**Flag Detection**  
 - Bit synchronous mode  
 0: No flag detected  
 1: Flag detected

**Transmitter Idle Status**  
 0: Not idle  
 1: idle

**CCDD Line Level Change**  
 0: Not changed  
 1: Changed

**CCSM Line Level Change**  
 0: Not changed  
 1: Changed

**Break End**  
 - Asynchronous mode  
 0: Break sequence end not detected  
 1: Break sequence end detected

**Idle Start Detection**  
 - Bit synchronous mode  
 0: Idle sequence start not detected  
 1: Idle sequence start detected

**Break Detection**  
 - Asynchronous mode  
 0: Break sequence starts not detected  
 1: Break sequence starts detected

**Abort Detection/GA Pattern Detection**  
 - Bit synchronous mode  
 0: Abort sequence start/GA pattern start not detected  
 1: Abort sequence start/GA pattern start detected

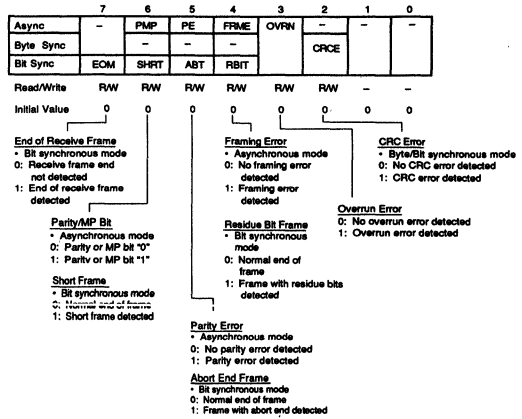


## G. Built-in Registers (cont.)

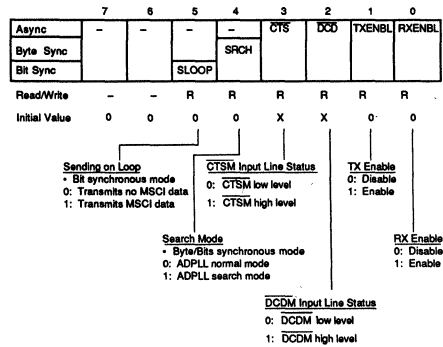
### MSCI

**Register** **Address** **Remarks**

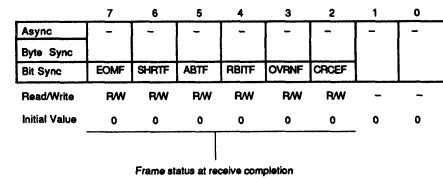
MSCI status register 2 (MST2) 0023H



MSCI status register 3 (MST3) 0024H



MSCI frame status register (MFST) 0025H

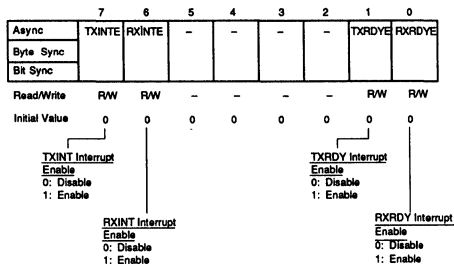


# G. Built-in Registers (cont.)

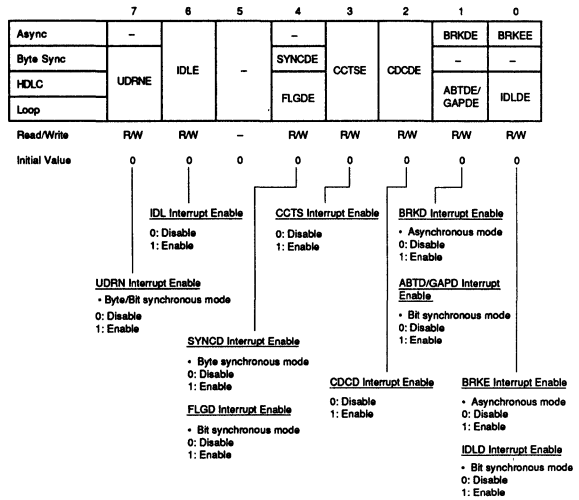
## MSCI

**Register Address Remarks**

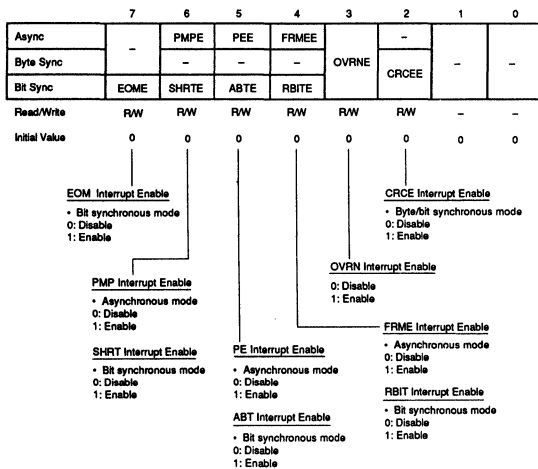
MSCI interrupt enable register 0 0026H  
(MIE0)



MSCI interrupt enable register 1 0027H  
(MIE1)



MSCI interrupt enable register 2 0028H  
(MIE2)



# G. Built-in Registers (cont.)

## MSCI

**Register**                      **Address**                      **Remarks**

MSCI frame interrupt enable register (MFIE)                      0029H

Async		7	6	5	4	3	2	1	0
Byte Sync	-	-	-	-	-	-	-	-	-
Bit Sync	EOMFE								
Read/Write	RW	-	-	-	-	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0	0

EOMF Interrupt Enable  
 • Bit synchronous mode  
 0: Disable  
 1: Enable

MSCI command register (MCMD)                      002AH

Async		7	6	5	4	3	2	1	0
Byte Sync	-	-	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0	
Bit Sync									
Read/Write	-	-	W	W	W	W	W	W	
Initial Value	-	-	-	-	-	-	-	-	-

Command

- Transmit Commands
- 000001: TX reset
- 000010: TX enable
- 000011: TX disable
- 000100: TX CRC initialization
- 000101: Exclusion from TX CRC calculation
- 000110: End of message
- 000111: Abort transmission
- 001000: MP bit on
- 001001: TX buffer clear
- Others: Reserved
- Receive Commands
- 010001: RX reset
- 010010: RX enable
- 010011: RX disable
- 010100: RX CRC initialization
- 010101: Message reject
- 010110: Search MP bit
- 010111: Exclusion from RX CRC calculation
- 011000: Forcing RX CRC calculation
- Other Commands
- 100001: Channel reset
- 110001: Enter search mode
- 000000: No operation

MSCI mode register 0 (MMD0)                      002BH

Async		7	6	5	4	3	2	1	0
Byte Sync		PRTCL2	PRTCL1	PRTCL0	AUTO		CRCCC	CRCL	CRCL0
Bit Sync									
Read/Write	RW	RW	RW	RW	-	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0	0

- Protocol Mode**
- 000: Asynchronous mode
- 001: Byte-sync, Mono-sync mode
- 010: Byte-sync, Bi-sync mode
- 011: Byte-sync, External synchronous mode
- 100: Bit-sync, HDLC mode
- 101: Bit-sync, Loop mode
- 110: Reserved
- 111: Reserved
- Auto-Enable**
- 0: Auto-enable reset
- 1: Auto-enable set
- CRC Calculation Code**
- Byte/Bit synchronous mode
- 0: Disable
- 1: Enable
- Stop Bit Length**
- Asynchronous mode
- 00: 1 bit
- 01: 1.5 bits
- 10: 2 bits
- 11: Reserved
- CRC Calculation Expression and Initial Value**
- Byte/Bit synchronous mode
- 0X: CRC-16
- 1X: CRC-CITT
- X0: Initial value = all 0s
- X1: Initial value = all 1s

# G. Built-in Registers (cont.)

## MSCI

**Register Address Remarks**

**MSCI mode register 1 (MMD1) 002CH**

	7	6	5	4	3	2	1	0
Async	BRATE1	BRATE0	TXCHR1	TXCHR0	RXCHR1	RXCHR0	PMPM1	PMPM0
Byte Sync	-	-	-	-	-	-	-	-
Bit Sync	ADDRS1	ADDRS0	-	-	-	-	-	-
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0

**Bit Rate**  
 • Asynchronous mode  
 00: 1/1 clock rate  
 01: 1/16 clock rate  
 10: 1/32 clock rate  
 11: 1/64 clock rate

**Transmit Character Length**  
 • Asynchronous mode  
 00: 8 bits/character  
 01: 7 bits/character  
 10: 6 bits/character  
 11: 5 bits/character

**Receive Character Length**  
 • Asynchronous mode  
 00: 8 bits/character  
 01: 7 bits/character  
 10: 6 bits/character  
 11: 5 bits/character

**Address Field Check**  
 • Bit asynchronous mode  
 00: Address field no-check  
 01: Single address 1  
 10: Single address 2  
 11: Dual address

**Parity/Multiprocessor Mode**  
 • Asynchronous mode  
 00: No parity/MP bit  
 01: MP bit appended (by command)  
 10: Even parity appended and checked  
 11: Odd parity appended and checked

**MSCI mode register 2 (MMD2) 002DH**

	7	6	5	4	3	2	1	0
Async	-	-	-	-	-	-	-	-
Byte Sync	NRZFM	CODE1	CODE0	DRATE1	DRATE0	-	CNCT1	CNCT0
Bit Sync	-	-	-	-	-	-	-	-
Read/Write	RW	RW	RW	RW	RW	-	RW	RW
Initial Value	0	0	0	0	0	0	0	0

**NRZ or FM Select**  
 • Byte/Bit synchronous mode  
 0: NRZ  
 1: FM

**Transmission Code Type**  
 • Byte/Bit synchronous mode  
 • NRZ  
 00: NRZ  
 01: NRZI  
 10: Reserved  
 11: Reserved  
 • FM  
 00: Manchester  
 01: FM1  
 10: FM0  
 11: Reserved

**ADPLL Operating Clock/Bit Rate**  
 • Byte/Bit synchronous mode  
 00: x8  
 01: x16  
 10: x32  
 11: Reserved

**Channel Connection**  
 00: Full duplex communications  
 01: Auto echo  
 10: Reserved  
 11: Local loop back

**MSCI control register (MCTL) 002EH**

	7	6	5	4	3	2	1	0
Async	TXRDYC	-	-	-	BRK	-	-	RTS
Byte Sync	-	-	UDFNC	IDLC	-	SYNCLD	-	-
Bit Sync	-	-	-	-	-	-	GOP	-
Read/Write	RW	-	RW	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	1

**TX Ready State Control**  
 0: TXRDY bit goes to 1 when the transmit buffer is empty  
 1: TXRDY bit goes to 1 when the transmit buffer is not full

**Send Break**  
 • Asynchronous mode  
 0: Off  
 1: On (break send)

**Idle State Control**  
 • Byte/Bit synchronous mode  
 0: Transmits a mark  
 1: Transmits an idle pattern

**Underrun State Control**  
 • Byte synchronous mode  
 0: Enters idle state immediately  
 1: Enters idle state after CRC transmission  
 • Bit synchronous mode  
 0: Enters idle state after aborting transmission  
 1: Enters idle state after FCS and flag transmission

**Request to Send**  
 0: RTSM line at low level  
 1: RTSM line at high level

**Go Active on Poll**  
 • Bit synchronous loop mode  
 0: Disable  
 1: Enable

**SYN Character Load Enable**  
 • Byte sync mode  
 0: Disable  
 1: Enable

## G. Built-in Registers (cont.)

### MSCI

**Register Address Remarks**

MSCI synchronous/address register 0 (MSA0) 002FH

	7	6	5	4	3	2	1	0
Async	-	-	-	-	-	-	-	-
Byte Sync	SA07	SA06	SA05	SA04	SA03	SA02	SA01	SA00
Bit Sync								
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	1

SYN Pattern for Reception/Address Field Check

• Byte synchronous mode

Mono-sync	SYN pattern for reception
Bi-sync	SYN pattern for transmission and reception (bits 7-0)
External-sync	Unused

• Bit synchronous mode

	Address field not checked	Unused
HDLC mode	Single address 1	Bits 7-0 of the secondary station address
	Single address 2	Unused
	Dual address	Bits 7-0 of the secondary station address
Loop mode	Address field not checked	Unused
	Single address 1	Bits 7-0 of the secondary station address
	4-bit address	Bits 7-4 of the secondary station address
	Dual address	Bits 7-0 of the secondary station address

MSCI synchronous/address register 1 (MSA1) 0030H

	7	6	5	4	3	2	1	0
Async	-	-	-	-	-	-	-	-
Byte Sync	SA17	SA16	SA15	SA14	SA13	SA12	SA11	SA10
Bit Sync								
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	1

SYN Pattern for Transmission/Address Field Check

• Byte synchronous mode

Mono-sync	SYN pattern for transmission
Bi-sync	SYN pattern for transmission and reception (bits 15-8)
External-sync	SYN pattern for transmission

• Bit synchronous mode

	Address field not checked	Unused
HDLC mode	Single address 1	Unused
	Single address 2	Bits 15-8 of the secondary station address
	Dual address	Bits 15-8 of the secondary station address
Loop mode	Address field not checked	Unused
	Single address 1	Unused
	4-bit address	Unused
	Dual address	Bits 15-8 of the secondary station address

MSCI idle pattern register (MIDL) 0031H

	7	6	5	4	3	2	1	0
Async	-	-	-	-	-	-	-	-
Byte Sync	IDL7	IDL6	IDL5	IDL4	IDL3	IDL2	IDL1	IDL0
Bit Sync								
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	1

Idle Pattern

## G. Built-in Registers (cont.)

### MSCI

Register	Address	Remarks
----------	---------	---------

MSCI time constant register (MTMC) 0032H

	7	6	5	4	3	2	1	0
Async								
Byte Sync								
Bit Sync								
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	1

Value loaded to the reload timer (1 – 256)

MSCI RX clock source register (MRXS) 0033H

	7	6	5	4	3	2	1	0
Async	-	RXCS2	RXCS1	RXCS0	RXBR3	RXBR2	RXBR1	RXBR0
Byte Sync								
Bit Sync								
Read/Write	-	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0

**Receive Clock Source**

- 000: RXCM line input
- 010: RXCM line input (noise suppression)
- 100: Internal baud rate generator (BRG) output
- 110: ADPLL output (BRG output for ADPLL operating clock)
- 111: ADPLL output (RXCM line input for ADPLL operating clock)
- Others: Reserved

**Receive Baud Rate**  
- Clock division ratio

- 0000: 1/1
- 0001: 1/2
- 0010: 1/4
- 0011: 1/8
- 0100: 1/16
- 0101: 1/32
- 0110: 1/64
- 0111: 1/128
- 1000: 1/256
- 1001: 1/512
- Others: Reserved

MSCI TX clock source register (MTXS) 0034H

	7	6	5	4	3	2	1	0
Async	-	TXCS2	TXCS1	TXCS0	TXBR3	TXBR2	TXBR1	TXBR0
Byte Sync								
Bit Sync								
Read/Write	-	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0

**Transmit Clock Source**

- 000: TXCM line input
- 100: Internal baud rate generator (BRG) output
- 110: Receiver clock
- Others: Reserved

**Transmit Baud Rate**  
- Clock division ratio

- 0000: 1/1
- 0001: 1/2
- 0010: 1/4
- 0011: 1/8
- 0100: 1/16
- 0101: 1/32
- 0110: 1/64
- 0111: 1/128
- 1000: 1/256
- 1001: 1/512
- Others: Reserved

Unused 0035H

Unused 0036H

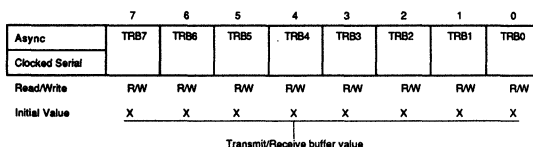
Unused 0037H

## G. Built-in Registers (cont.)

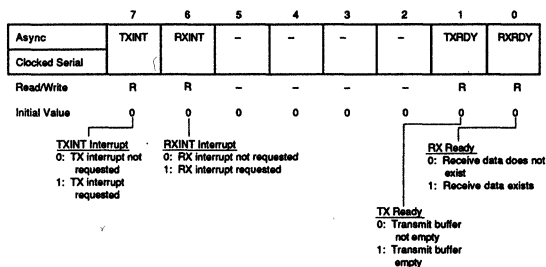
### ASCII/CSIO

Register	Address	Remarks
----------	---------	---------

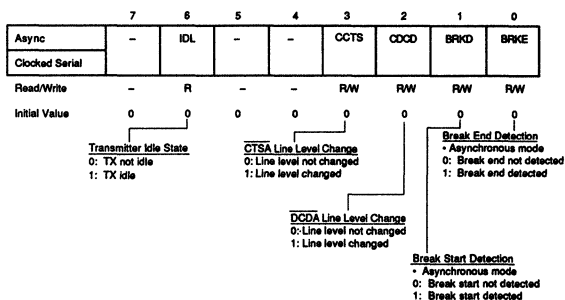
ASCII TX/RX buffer register (TRB) 0038H



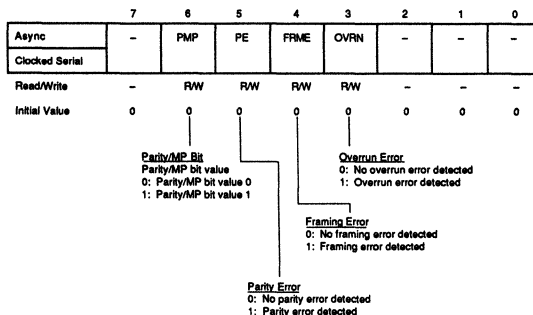
ASCII status register 0 (ST0) 0039H



ASCII status register 1 (ST1) 003AH



ASCII status register 2 (ST2) 003BH



## G. Built-in Registers (cont.)

### ASCII/CSIO

**Register Address Remarks**

**ASCII status register 3 (ST3) 003CH**

	7	6	5	4	3	2	1	0
Async	-	-	-	-	CTS	DCD	TXENBL	RXENBL
Clocked Serial	-	-	-	-	R	R	R	R
Read/Write	-	-	-	-	R	R	R	R
Initial Value	0	0	0	0	X	X	0	0

**CTS Input Line Level**  
 0: CTS line low level  
 1: CTS line high level

**DCDA Input Line Level**  
 0: DCDA line low level  
 1: DCDA line high level

**TX Enable**  
 0: Disable  
 1: Enable

**RX Enable**  
 0: Disable  
 1: Enable

Unused 003DH

**ASCII interrupt enable register 0 (IE0) 003EH**

	7	6	5	4	3	2	1	0
Async	TXINTE	RXINTE	-	-	-	-	TXRDYE	RXRDYE
Clocked Serial	-	-	-	-	-	-	-	-
Read/Write	RW	RW	-	-	-	-	RW	RW
Initial Value	0	0	0	0	0	0	0	0

**TXINT Interrupt Enable**  
 0: Disable  
 1: Enable

**RXINT Interrupt Enable**  
 0: Disable  
 1: Enable

**TXRDY Interrupt Enable**  
 0: Disable  
 1: Enable

**RXRDY Interrupt Enable**  
 0: Disable  
 1: Enable

**ASCII interrupt enable register 1 (IE1) 003FH**

	7	6	5	4	3	2	1	0
Async	-	IDLE	-	-	CCTS	CCDCE	BRKDE	BRKEE
Clocked Serial	-	-	-	-	-	-	-	-
Read/Write	-	RW	-	-	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0

**IDL Interrupt Enable**  
 0: Disable  
 1: Enable

**CCTS Interrupt Enable**  
 0: Disable  
 1: Enable

**CCDC Interrupt Enable**  
 0: Disable  
 1: Enable

**BRKD Interrupt Enable**  
 - Asynchronous mode  
 0: Disable  
 1: Enable

**BRKE Interrupt Enable**  
 - Asynchronous mode  
 0: Disable  
 1: Enable



## G. Built-in Registers (cont.)

### ASCII/CSIO

Register	Address	Remarks
----------	---------	---------

ASCII interrupt enable register 2 (IE2) 0040H

	7	6	5	4	3	2	1	0
Async	-	PMPE	PEE	FRMEE	OVRNE	-	-	-
Clocked Serial	-	-	-	-	-	-	-	-
Read/Write	-	RW	RW	RW	RW	-	-	-
Initial Value	0	0	0	0	0	0	0	0

**PMP Interrupt Enable**  
 0: Disable  
 1: Enable

**PE Interrupt Enable**  
 0: Disable  
 1: Enable

**FRM Interrupt Enable**  
 0: Disable  
 1: Enable

**OVRN Interrupt Enable**  
 - Asynchronous mode  
 0: Disable  
 1: Enable

Unused 0041H

ASCII command register (CMD) 0042H

	7	6	5	4	3	2	1	0
Async	-	-	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
Clocked Serial	-	-	-	-	-	-	-	-
Read/Write	-	-	W	W	W	W	W	W
Initial Value	-	-	-	-	-	-	-	-

**Command**

- Transmit commands
- 000001: TX reset
- 000010: TX enable
- 000011: TX disable
- 001000: MP bit on
- 001001: TX buffer clear
- Receive commands
- 010001: RX reset
- 010010: RX enable
- 010011: RX disable
- 010110: Search MP bit
- Other commands
- 100001: Channel reset
- 000000: No operation
- Others: Reserved

ASCII mode register 0 (MD0) 0043H

	7	6	5	4	3	2	1	0
Async	PRTCL2	PRTCL1	PRTCL0	AUTO	-	-	STOP1	STOP2
Clocked Serial	-	-	-	-	-	-	-	-
Read/Write	RW	RW	RW	RW	-	-	RW	RW
Initial Value	0	0	0	0	0	0	0	0

**Protocol Mode**  
 000: Asynchronous mode  
 110: Clocked serial mode  
 Other values are reserved

**Auto-Enable**  
 0: Auto-enable function not used  
 1: Auto-enable function used

**Stop Bit Length**  
 - Asynchronous mode  
 00: 1 bit  
 10: 2 bits  
 - Clocked serial  
 Reserved

ASCII mode register 1 (MD1) 0044H

	7	6	5	4	3	2	1	0
Async	BRATE1	BRATE0	TXCHR1	TXCHR0	RXCHR1	RXCHR0	PMPM1	PMPM0
Clocked Serial	-	-	-	-	-	-	-	-
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0

**Bit Rate**  
 - Asynchronous mode  
 00: 1/1 of clock rate  
 01: 1/16 of clock rate  
 10: 1/32 of clock rate  
 11: 1/64 of clock rate  
 - Clocked serial mode  
 These bits should be set to 00.

**Transmit Character Length**  
 00: 8 bits/character  
 01: 7 bits/character

**Receive Character Length**  
 00: 8 bits/character  
 01: 7 bits/character

**Parity/Multiprocessor Mode**  
 00: No parity/MP bit  
 01: MP bit appended (value specified by command)  
 10: Even parity appended and checked  
 11: Odd parity appended and checked

## G. Built-in Registers (cont.)

### ASCII/CSIO

Register	Address	Remarks
----------	---------	---------

ASCII mode register 2  
(MD2)

0045H

	7	6	5	4	3	2	1	0
Async	-	-	-	-	-	-	CNCT1	CNCT0
Clocked Serial	-	-	-	-	-	-	-	-
Read/Write	-	-	-	-	-	-	RW	RW
Initial Value	0	0	0	0	0	0	0	0

Channel Connection  
00: Full duplex  
01: Auto-echo  
10: Reserved  
11: Local loop-back

ASCII control register  
(CTL)

0046H

	7	6	5	4	3	2	1	0
Async	-	-	-	-	BRK	-	-	RTS
Clocked Serial	-	-	-	-	-	-	-	-
Read/Write	-	-	-	-	RW	-	-	RW
Initial Value	0	0	0	0	0	0	0	1

Break Send  
- Asynchronous mode  
0: Off (Normal operation)  
1: On (Break send)  
- Clocked serial mode  
Set this bit to 0

Request to Send  
0: RTSA low level output  
1: RTSA high level output

Unused

0047H

Unused

0048H

Unused

0049H

## G. Built-in Registers (cont.)

### ASCII/CSIO

**Register**                      **Address**              **Remarks**

ASCII time constant register    004AH  
(TMC)

	7	6	5	4	3	2	1	0
Async								
Clocked Serial	TMC7	TMC6	TMC5	TMC4	TMC3	TMC2	TMC1	TMC0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	1

Reload Timer Value (1 – 256)

ASCII RX clock source register    004BH  
(RXS)

	7	6	5	4	3	2	1	0
Async								
Clocked Serial	–	RXCS2	RXCS1	RXCS0	–	–	–	–
Read/Write	–	R/W	R/W	R/W	–	–	–	–
Initial Value	0	0	0	0	0	0	0	0

**Receive Clock Source**

- Asynchronous mode
- 000: RXCA line input
- 100: Internal baud rate generator (BRG) output
- Others: Reserved

**RX Master/Slave Mode Select**

- Clocked serial mode
- 000: Slave mode
- 100: Master mode
- Others: Reserved

ASCII TX clock source register    004CH  
(TXS)

	7	6	5	4	3	2	1	0
Async	–	TXCS2	TXCS1	TXCS0	TXBR3	TXBR2	TXBR1	TXBR0
Clocked Serial								
Read/Write	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

**Transmit Clock Source**

- Asynchronous mode
- 000: TXCA line input
- 100: Internal baud rate generator (BRG) output
- Others: Reserved

**TX Master/Slave Mode Select**

- Clocked serial mode
- 000: Slave mode
- 100: Master mode
- Others: Reserved

**Baud Rate**

- Clock division ratio
- 0000: 1/1    0101: 1/32
- 0001: 1/2    0110: 1/64
- 0010: 1/4    0111: 1/128
- 0011: 1/8    1000: 1/256
- 0100: 1/16    1001: 1/512
- Others: Reserved

Unused                              004DH

Unused                              004EH

Unused                              004FH

## G. Built-in Registers (cont.)

### Timer (channel 0)

Register	Address	Remarks
----------	---------	---------

Timer up-counter channel 0  
(TCNT channel 0)

0050H

	7	6	5	4	3	2	1	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Timer constant register  
channel 0 (TCONR channel 0)

0051H

	7	6	5	4	3	2	1	0
Read/Write	W	W	W	W	W	W	W	W
Initial Value	1	1	1	1	1	1	1	1

Timer control/status register  
channel 0 (TCSR channel 0)

0052H

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

**Compare Match Flag**  
 0: TCNT and TCONR are not equal  
 1: TCNT and TCONR are equal

**CMF Interrupt Enable**  
 0: Disable  
 1: Enable

**Timer Enable**  
 0: Count stop  
 1: Count start

**Timer Output Select**  
 00: Output fixed to 0  
 01: Toggled output  
 10: Output 0  
 11: Output 1

**Input Clock Select**  
 00: BC  
 01: BC/8  
 10: BC/128  
 11: External event count signal

Timer expand prescale register  
channel 0 (TEPR channel 0)

0053H

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

**Enable Expand Prescaler**  
 0: Clock is selected by the CKS1-0 bits in TCSR  
 1: Clock is selected by the ECKS2-0 bits in TEPR

**Expand Clock Input Select**  
 000: BC  
 001: BC/2  
 010: BC/4  
 011: BC/8  
 100: BC/16  
 101: BC/32  
 110: BC/64  
 111: BC/128

## G. Built-in Registers (cont.)

### Timer (channel 1)

Register	Address	Remarks																																				
Timer up-counter channel 1 (TCNT channel 1)	0054H	<table border="1" style="margin-left: auto; margin-right: auto;"> <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 style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td> </tr> </table> <p>Read/Write: R/W R/W R/W R/W R/W R/W R/W R/W Initial Value: 0 0 0 0 0 0 0 0</p>	7	6	5	4	3	2	1	0																												
7	6	5	4	3	2	1	0																															
Timer constant register channel 1 (TCO NR channel 1)	0055H	<table border="1" style="margin-left: auto; margin-right: auto;"> <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 style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td><td style="width: 20px; height: 20px;"></td> </tr> </table> <p>Read/Write: W W W W W W W W Initial Value: 1 1 1 1 1 1 1 1</p>	7	6	5	4	3	2	1	0																												
7	6	5	4	3	2	1	0																															
Timer control/status register channel 1 (TCSR channel 1)	0056H	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>Bit Name</td> <td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td></td> <td>CMF</td><td>ECMI</td><td>-</td><td>TME</td><td>TOS1</td><td>TOS0</td><td>CKS1</td><td>CKS0</td> </tr> <tr> <td>Read/Write</td> <td>R</td><td>R/W</td><td>-</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td> </tr> <tr> <td>Initial Value</td> <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> <p> <b>Compare Match Flag</b>            0: TCNT and TCONR are not equal            1: TCNT and TCONR are equal         </p> <p> <b>CMF Interrupt Enable</b>            0: Disable            1: Enable         </p> <p> <b>Timer Enable</b>            0: Count stop            1: Count start         </p> <p> <b>Timer Output Select</b>            00: Output fixed to 0            01: Toggled output            10: Output 0            11: Output 1         </p> <p> <b>Input Clock Select</b>            00: BC            01: BC/8            10: BC/128            11: External event count signal         </p>	Bit Name	7	6	5	4	3	2	1	0		CMF	ECMI	-	TME	TOS1	TOS0	CKS1	CKS0	Read/Write	R	R/W	-	R/W	R/W	R/W	R/W	R/W	Initial Value	0	0	0	0	0	0	0	0
Bit Name	7	6	5	4	3	2	1	0																														
	CMF	ECMI	-	TME	TOS1	TOS0	CKS1	CKS0																														
Read/Write	R	R/W	-	R/W	R/W	R/W	R/W	R/W																														
Initial Value	0	0	0	0	0	0	0	0																														
Timer expand prescale register channel 1 (TEPR channel 1)	0057H	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>Bit Name</td> <td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td></td> <td>EEP</td><td>-</td><td>-</td><td>-</td><td>-</td><td>ECKS2</td><td>ECKS1</td><td>ECKS0</td> </tr> <tr> <td>Read/Write</td> <td>R/W</td><td>-</td><td>-</td><td>-</td><td>-</td><td>R/W</td><td>R/W</td><td>R/W</td> </tr> <tr> <td>Initial Value</td> <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> <p> <b>Enable Expand Prescaler</b>            0: Clock is selected by the CKS1-0 bits in TCSR            1: Clock is selected by the ECKS2-0 bits in TEPR         </p> <p> <b>Expand Clock Input Select</b>            000: BC            001: BC/2            010: BC/4            011: BC/8            100: BC/16            101: BC/32            110: BC/64            111: BC/128         </p>	Bit Name	7	6	5	4	3	2	1	0		EEP	-	-	-	-	ECKS2	ECKS1	ECKS0	Read/Write	R/W	-	-	-	-	R/W	R/W	R/W	Initial Value	0	0	0	0	0	0	0	0
Bit Name	7	6	5	4	3	2	1	0																														
	EEP	-	-	-	-	ECKS2	ECKS1	ECKS0																														
Read/Write	R/W	-	-	-	-	R/W	R/W	R/W																														
Initial Value	0	0	0	0	0	0	0	0																														

## G. Built-in Registers (cont.)

### DMAC (channel 0)

Register	Address	Remarks															
Destination address register L channel 0/buffer address register L channel 0 (DARL channel 0/ BARL channel 0)	0058H	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>Unused</td> <td>DARB</td> <td>DARH</td> <td>DARL</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td>Unused</td> <td>DARB</td> <td>DARH</td> <td>DARL</td> </tr> <tr> <td>Chained-block transfer mode</td> <td>Unused</td> <td>BARB</td> <td>BARH</td> <td>BARL</td> </tr> </table>	Single-block transfer mode (dual address)	Unused	DARB	DARH	DARL	Single-block transfer mode (single address)	Unused	DARB	DARH	DARL	Chained-block transfer mode	Unused	BARB	BARH	BARL
Single-block transfer mode (dual address)	Unused	DARB	DARH	DARL													
Single-block transfer mode (single address)	Unused	DARB	DARH	DARL													
Chained-block transfer mode	Unused	BARB	BARH	BARL													
Destination address register H channel 0/buffer address register H channel 0 (DARH channel 0/ BARH channel 0)	0059H																
Destination address register B channel 0/buffer address register B channel 0 (DARB channel 0/ BARB channel 0)	005AH																
Source address register L channel 0 (SARL channel 0)	005BH	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>Unused</td> <td>SARB</td> <td>SARH</td> <td>SARL</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td>Unused</td> <td>SARB</td> <td>SARH</td> <td>SARL</td> </tr> <tr> <td>Chained-block transfer mode</td> <td>Unused</td> <td>CPB</td> <td>Unused</td> <td>Unused</td> </tr> </table>	Single-block transfer mode (dual address)	Unused	SARB	SARH	SARL	Single-block transfer mode (single address)	Unused	SARB	SARH	SARL	Chained-block transfer mode	Unused	CPB	Unused	Unused
Single-block transfer mode (dual address)	Unused	SARB	SARH	SARL													
Single-block transfer mode (single address)	Unused	SARB	SARH	SARL													
Chained-block transfer mode	Unused	CPB	Unused	Unused													
Source address register H channel 0 (SARH channel 0)	005CH																
Source address register B channel 0/chain pointer base channel 0 (SARB channel 0/ CPB channel 0)	005DH																

## G. Built-in Registers (cont.)

### DMAC (channel 0)

Register	Address	Remarks									
Current descriptor address register L channel 0 (CDAL channel 0)	005EH	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Chained-block transfer mode</td> <td>CDAH</td> <td>CDAL</td> </tr> </table>	Single-block transfer mode (dual address)	Unused	Unused	Single-block transfer mode (single address)	Unused	Unused	Chained-block transfer mode	CDAH	CDAL
Single-block transfer mode (dual address)	Unused	Unused									
Single-block transfer mode (single address)	Unused	Unused									
Chained-block transfer mode	CDAH	CDAL									
Current descriptor address register H channel 0 (CDAH channel 0)	005FH	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Chained-block transfer mode</td> <td>CDAH</td> <td>CDAL</td> </tr> </table>	Single-block transfer mode (dual address)	Unused	Unused	Single-block transfer mode (single address)	Unused	Unused	Chained-block transfer mode	CDAH	CDAL
Single-block transfer mode (dual address)	Unused	Unused									
Single-block transfer mode (single address)	Unused	Unused									
Chained-block transfer mode	CDAH	CDAL									
Error descriptor address register L channel 0 (EDAL channel 0)	0060H	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Chained-block transfer mode</td> <td>EDAH</td> <td>EDAL</td> </tr> </table>	Single-block transfer mode (dual address)	Unused	Unused	Single-block transfer mode (single address)	Unused	Unused	Chained-block transfer mode	EDAH	EDAL
Single-block transfer mode (dual address)	Unused	Unused									
Single-block transfer mode (single address)	Unused	Unused									
Chained-block transfer mode	EDAH	EDAL									
Error descriptor address register H channel 0 (EDAH channel 0)	0061H	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Chained-block transfer mode</td> <td>EDAH</td> <td>EDAL</td> </tr> </table>	Single-block transfer mode (dual address)	Unused	Unused	Single-block transfer mode (single address)	Unused	Unused	Chained-block transfer mode	EDAH	EDAL
Single-block transfer mode (dual address)	Unused	Unused									
Single-block transfer mode (single address)	Unused	Unused									
Chained-block transfer mode	EDAH	EDAL									
Receive buffer length L channel 0 (BFL channel 0)	0062H	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Chained-block transfer mode</td> <td>Memory to MSCI BFLH</td> <td>Unused BFL</td> </tr> </table>	Single-block transfer mode (dual address)	Unused	Unused	Single-block transfer mode (single address)	Unused	Unused	Chained-block transfer mode	Memory to MSCI BFLH	Unused BFL
Single-block transfer mode (dual address)	Unused	Unused									
Single-block transfer mode (single address)	Unused	Unused									
Chained-block transfer mode	Memory to MSCI BFLH	Unused BFL									
Receive buffer length H channel 0 (BFLH channel 0)	0063H	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Chained-block transfer mode</td> <td>Memory to MSCI BFLH</td> <td>Unused BFL</td> </tr> </table>	Single-block transfer mode (dual address)	Unused	Unused	Single-block transfer mode (single address)	Unused	Unused	Chained-block transfer mode	Memory to MSCI BFLH	Unused BFL
Single-block transfer mode (dual address)	Unused	Unused									
Single-block transfer mode (single address)	Unused	Unused									
Chained-block transfer mode	Memory to MSCI BFLH	Unused BFL									
Byte count register L channel 0 (BCRL channel 0)	0064H	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>BCPH</td> <td>BCRL</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td>BCPH</td> <td>BCRL</td> </tr> <tr> <td>Chained-block transfer mode</td> <td>BCPH</td> <td>BCRL</td> </tr> </table>	Single-block transfer mode (dual address)	BCPH	BCRL	Single-block transfer mode (single address)	BCPH	BCRL	Chained-block transfer mode	BCPH	BCRL
Single-block transfer mode (dual address)	BCPH	BCRL									
Single-block transfer mode (single address)	BCPH	BCRL									
Chained-block transfer mode	BCPH	BCRL									
Byte count register H channel 0 (BCRH channel 0)	0065H	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>BCPH</td> <td>BCRL</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td>BCPH</td> <td>BCRL</td> </tr> <tr> <td>Chained-block transfer mode</td> <td>BCPH</td> <td>BCRL</td> </tr> </table>	Single-block transfer mode (dual address)	BCPH	BCRL	Single-block transfer mode (single address)	BCPH	BCRL	Chained-block transfer mode	BCPH	BCRL
Single-block transfer mode (dual address)	BCPH	BCRL									
Single-block transfer mode (single address)	BCPH	BCRL									
Chained-block transfer mode	BCPH	BCRL									
Unused	0066H										
Unused	0067H										

## G. Built-in Registers (cont.)

### DMAC (channel 0)

Register Address Remarks

DMA status register channel 0 0068H  
(DSR channel 0)

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)								
Single-block transfer mode (single address)	EOT	-	-	-	-	-	DE	DWE
Chained-block transfer mode		EOB	BOF	COF				
Read/Write	RW	RW	RW	RW	-	-	RW	W
Initial Value	0	0	0	0	0	0	0	1

**End of Transfer**  
0: Transfer not completed  
1: Transfer completed

**Counter Overflow**  
• Chained-block transfer  
0: Error not detected  
1: Error detected

**Buffer Overflow/Underflow**  
• Chained block transfer  
0: Error not detected  
1: Error detected

**End of Frame Transfer**  
• Chained-block transfer  
0: Frame transfer not completed  
1: Frame transfer completed

**DMA Enable**  
0: Disable  
1: Enable

**DE Bit Write Enable**  
0: Enable  
1: Disable

DMA mode register A 0069H  
channel 0 (DMRA channel 0)

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)					-			
Single-block transfer mode (single address)	RSEL1	RSEL0	AMOD	TMOD	RT		CNTE	DMS
Chained-block transfer mode						NF		
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	1	0	0	0	0	0

**DMA Transfer Request Source**  
00: External line  
01: Reserved  
10: MSCI  
11: Reserved

**DMA Transfer Direction**  
• Single-block (single address)/  
Chained-block modes  
0: MSCI to memory  
1: Memory to MSCI

**DMA Transfer Mode**  
00: Single-block transfer (single address)  
01: Chained-block transfer  
10: Single-block transfer (dual address)  
11: Reserved

**DREQ Input Mode**  
• Single-block (dual address)  
0: DREQ level sensitive  
1: DREQ edge sensitive  
• Single-block (single address)/  
Chained-block modes  
Set this bit to 0.

**Frame-End Interrupt-Counter Enable/Disable**  
• Single-block transfer (single/dual address)  
Set this bit to 0.  
• Chained-block transfer  
0: Frame-end interrupt-counter disabled  
1: Frame-end interrupt-counter enabled

**Number of DMA Frames**  
• Chained-block transfers  
0: Single frame  
1: Multi-frame

DMA mode register B 006AH  
channel 0 (DMRB channel 0)

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)				DM1	DM0	SM1	SMD	MMOD
Single-block transfer mode (single address)	-	-	-	-	-	-	-	-
Chained-block transfer mode								
Read/Write	-	-	-	RW	RW	RW	RW	RW
Initial Value	0	0	0	1	1	0	0	0

**Destination (address increment or decrement)**  
• Single-block (dual address)  
00: Memory (+1)  
01: Memory (-1)  
10: Memory (fixed)  
11: IO (fixed)

**Source (address increment or decrement)**  
• Single-block (dual address)  
00: Memory (+1)  
01: Memory (-1)  
10: Memory (fixed)  
11: IO (fixed)

**Mode for Memory-to-Memory Transfers**  
• Single-block (dual address)  
0: Cycle steal mode  
1: Burst mode



## G. Built-in Registers (cont.)

### DMAC (channel 0)

Register	Address	Remarks																																													
Frame-end interrupt-counter channel 0 (FCT channel 0)	006BH	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> </tr> <tr> <td>Chained-block transfer mode</td> <td></td> <td></td> <td></td> <td></td> <td>FCT3</td> <td>FCT2</td> <td>FCT1</td> <td>FCT0</td> </tr> <tr> <td>Read/Write</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>R</td> <td>R</td> <td>R</td> <td>R</td> </tr> <tr> <td>Initial Value</td> <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> <p style="text-align: right;">Frame-End Interrupt-Counter Value</p>	Single-block transfer mode (dual address)									Single-block transfer mode (single address)	-	-	-	-	-	-	-	-	Chained-block transfer mode					FCT3	FCT2	FCT1	FCT0	Read/Write	-	-	-	-	R	R	R	R	Initial Value	0	0	0	0	0	0	0	0
Single-block transfer mode (dual address)																																															
Single-block transfer mode (single address)	-	-	-	-	-	-	-	-																																							
Chained-block transfer mode					FCT3	FCT2	FCT1	FCT0																																							
Read/Write	-	-	-	-	R	R	R	R																																							
Initial Value	0	0	0	0	0	0	0	0																																							

DMA interrupt enable register channel 0 (DIR channel 0)

Single-block transfer mode (dual address)								
Single-block transfer mode (single address)	EOTE	-	-	-	-	-	-	-
Chained-block transfer mode		EOME	BOFE	COFE				
Read/Write	R/W	R/W	R/W	R/W	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0

Transfer End Interrupt Enable  
0: Disable  
1: Enable

Counter Overflow Interrupt Enable  
• Chained-block transfer mode  
0: Disable  
1: Enable

Frame Transfer End Interrupt Enable  
• Chained-block transfer mode  
0: Disable  
1: Enable

Buffer Overflow/Underflow Interrupt Enable  
• Chained-block transfer mode  
0: Disable  
1: Enable

DMA command register channel 0 (DCR channel 0)

Single-block transfer mode (dual address)								
Single-block transfer mode (single address)	-	-	-	-	-	-	CMD1	CMD0
Chained-block transfer mode								
Read/Write	-	-	-	-	-	-	W	W
Initial Value	-	-	-	-	-	-	-	-

Command Specification  
01: Software abort  
10: Frame-end interrupt-counter-clear  
Others: Reserved

Command Name	Function
Software abort (01H)	Initializes the corresponding DMAC channel (see figure 6-2). All DMAC registers maintain their previous value.
Frame-end interrupt-counter-clear (02H)	Clears the frame-end interrupt-counter (FCT) of the corresponding DMAC channel to 0H and the EOM bit in the DSR to 0.

Unused 006EH

Unused 006FH

## G. Built-in Registers (cont.)

### DMAC (channel 1)

Register	Address	Remarks															
Destination address register L channel 1/buffer address register L channel 1 (DARL channel 1/BARL channel 1)	0070H	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>Unused</td> <td>DARB</td> <td>DARH</td> <td>DARL</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Chained-block transfer mode</td> <td>Unused</td> <td>BARB</td> <td>BARH</td> <td>BARL</td> </tr> </table>	Single-block transfer mode (dual address)	Unused	DARB	DARH	DARL	Single-block transfer mode (single address)					Chained-block transfer mode	Unused	BARB	BARH	BARL
Single-block transfer mode (dual address)	Unused	DARB	DARH	DARL													
Single-block transfer mode (single address)																	
Chained-block transfer mode	Unused	BARB	BARH	BARL													
Destination address register H channel 1/buffer address register H channel 1 (DARH channel 1/BARH channel 1)	0071H																
Destination address register B channel 1/buffer address register B channel 1 (DARB channel 1/BARB channel 1)	0072H																
Source address register L channel 1 (SARL channel 1)	0073H	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>Unused</td> <td>SARB</td> <td>SARH</td> <td>SARL</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Chained-block transfer mode</td> <td>Unused</td> <td>CPB</td> <td>Unused</td> <td>Unused</td> </tr> </table>	Single-block transfer mode (dual address)	Unused	SARB	SARH	SARL	Single-block transfer mode (single address)					Chained-block transfer mode	Unused	CPB	Unused	Unused
Single-block transfer mode (dual address)	Unused	SARB	SARH	SARL													
Single-block transfer mode (single address)																	
Chained-block transfer mode	Unused	CPB	Unused	Unused													
Source address register H channel 1 (SARH channel 1)	0074H																
Source address register B channel 1/chain pointer base channel 1 (SARB channel 1/ CPB channel 1)	0075H																
Current descriptor address register L channel 1 (CDAL channel 1)	0076H	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td></td> <td></td> </tr> <tr> <td>Chained-block transfer mode</td> <td>CDAH</td> <td>CDAL</td> </tr> </table>	Single-block transfer mode (dual address)	Unused	Unused	Single-block transfer mode (single address)			Chained-block transfer mode	CDAH	CDAL						
Single-block transfer mode (dual address)	Unused	Unused															
Single-block transfer mode (single address)																	
Chained-block transfer mode	CDAH	CDAL															
Current descriptor address register H channel 1 (CDAH channel 1)	0077H																

## G. Built-in Registers (cont.)

### DMAC (channel 1)

Register	Address	Remarks																																																						
Error descriptor address register L channel 1 (EDAL channel 1)	0078H																																																							
Error descriptor address register H channel 1 (EDAH channel 1)	0079H	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td></td> <td></td> </tr> <tr> <td>Chained-block transfer mode</td> <td>EDAH</td> <td>EDAL</td> </tr> </table>	Single-block transfer mode (dual address)	Unused	Unused	Single-block transfer mode (single address)			Chained-block transfer mode	EDAH	EDAL																																													
Single-block transfer mode (dual address)	Unused	Unused																																																						
Single-block transfer mode (single address)																																																								
Chained-block transfer mode	EDAH	EDAL																																																						
Receive buffer length L channel 1 (BFL channel 1)	007AH																																																							
Receive buffer length H channel 1 (BFLH channel 1)	007BH	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td>Unused</td> <td>Unused</td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td></td> <td></td> </tr> <tr> <td>Chained-block transfer mode</td> <td>Memory to MSC1</td> <td>Unused</td> </tr> <tr> <td>Chained-block transfer mode</td> <td>MSC1 to memory</td> <td>BFLH</td> </tr> <tr> <td></td> <td></td> <td>BFL</td> </tr> </table>	Single-block transfer mode (dual address)	Unused	Unused	Single-block transfer mode (single address)			Chained-block transfer mode	Memory to MSC1	Unused	Chained-block transfer mode	MSC1 to memory	BFLH			BFL																																							
Single-block transfer mode (dual address)	Unused	Unused																																																						
Single-block transfer mode (single address)																																																								
Chained-block transfer mode	Memory to MSC1	Unused																																																						
Chained-block transfer mode	MSC1 to memory	BFLH																																																						
		BFL																																																						
Byte count register L channel 1 (BCRL channel 1)	007CH																																																							
Byte count register H channel 1 (BCRH channel 1)	007DH	<table border="1"> <tr> <td>Single-block transfer mode (dual address)</td> <td></td> <td></td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td>BCRH</td> <td>BCRL</td> </tr> <tr> <td>Chained-block transfer mode</td> <td></td> <td></td> </tr> </table>	Single-block transfer mode (dual address)			Single-block transfer mode (single address)	BCRH	BCRL	Chained-block transfer mode																																															
Single-block transfer mode (dual address)																																																								
Single-block transfer mode (single address)	BCRH	BCRL																																																						
Chained-block transfer mode																																																								
Unused	007EH																																																							
Unused	007FH																																																							
DMA status register channel 1 (DSR channel 1)	0080H	<table border="1"> <tr> <td></td> <td>7</td> <td>6</td> <td>5</td> <td>4</td> <td>3</td> <td>2</td> <td>1</td> <td>0</td> </tr> <tr> <td>Single-block transfer mode (dual address)</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Single-block transfer mode (single address)</td> <td>EOT</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>DE</td> <td>DWE</td> </tr> <tr> <td>Chained-block transfer mode</td> <td></td> <td>EOM</td> <td>BOF</td> <td>COF</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Read/Write</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>RW</td> <td>-</td> <td>-</td> <td>RW</td> <td>W</td> </tr> <tr> <td>Initial Value</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> </tr> </table> <p> <b>End of Transfer</b>          0: Transfer not completed          1: Transfer completed       </p> <p> <b>Counter Overflow</b>          0: Chained-block transfer          0: Error not detected          1: Error detected       </p> <p> <b>Buffer Overflow/Underflow</b>          0: Chained block transfer          0: Error not detected          1: Error detected       </p> <p> <b>End of Frame Transfer</b>          0: Chained-block transfer          0: Frame transfer not completed          1: Frame transfer completed       </p> <p> <b>DMA Enable</b>          0: Disable          1: Enable       </p> <p> <b>DE Bit Write Enable</b>          0: Enable          1: Disable       </p>		7	6	5	4	3	2	1	0	Single-block transfer mode (dual address)									Single-block transfer mode (single address)	EOT	-	-	-	-	-	DE	DWE	Chained-block transfer mode		EOM	BOF	COF					Read/Write	RW	RW	RW	RW	-	-	RW	W	Initial Value	0	0	0	0	0	0	0	1
	7	6	5	4	3	2	1	0																																																
Single-block transfer mode (dual address)																																																								
Single-block transfer mode (single address)	EOT	-	-	-	-	-	DE	DWE																																																
Chained-block transfer mode		EOM	BOF	COF																																																				
Read/Write	RW	RW	RW	RW	-	-	RW	W																																																
Initial Value	0	0	0	0	0	0	0	1																																																

## G. Built-in Registers (cont.)

### DMAC (channel 1)

Register Address Remarks

DMA mode register A channel 1 0081H

(DMRA channel 1)

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)								
Single-block transfer mode (single address)	RSEL1	RSEL0	AMOD	TMOD	RT		CNTE	DMS
Chained-block transfer mode						NF		
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	1	0	0	0	0	0

**DMA Transfer Request Source**  
 00: External line  
 01: Reserved  
 10: MSCI  
 11: Reserved

**DMA Transfer Direction**  
 • Single-block (single address)  
 Chained-block modes  
 0: MSCI to memory  
 1: Memory to MSCI

**DMA Transfer Mode**  
 00: Single-block transfer (single address)  
 01: Chained-block transfer  
 10: Single-block transfer (dual address)  
 11: Reserved

**DREO Input Mode**  
 • Single-block (dual address)  
 0: DREO level sensitive  
 1: DREO edge sensitive  
 • Single-block (single address)  
 Chained-block modes  
 Set this bit to 0.

**Frame-End Interrupt-Counter Enable/Disable**  
 • Single-block transfer (single/dual address)  
 Set this bit to 0.  
 • Chained-block transfer  
 0: Frame-end interrupt-counter disabled  
 1: Frame-end interrupt-counter enabled

**Number of DMA Frames**  
 • Chained-block transfers  
 0: Single frame  
 1: Multi-frame

DMA mode register B channel 1 0082H

(DMRB channel 1)

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)				DM1	DM0	SM1	SM0	MMOD
Single-block transfer mode (single address)	-	-	-	-	-	-	-	-
Chained-block transfer mode								
Read/Write	-	-	-	RW	RW	RW	RW	RW
Initial Value	0	0	0	1	1	0	0	0

**Destination (address increment or decrement)**  
 • Single-block (dual address)  
 00: Memory (+1)  
 01: Memory (-1)  
 10: Memory (fixed)  
 11: IO (fixed)

**Source (address increment or decrement)**  
 • Single-block (dual address)  
 00: Memory (+1)  
 01: Memory (-1)  
 10: Memory (fixed)  
 11: IO (fixed)

**Mode for Memory-to-Memory Transfers**  
 • Single-block (dual address)  
 0: Cycle steal mode  
 1: Burst mode

Frame-end interrupt-counter channel 1 (FCT channel 1) 0083H

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)								
Single-block transfer mode (single address)	-	-	-	-				
Chained-block transfer mode					FCT3	FCT2	FCT1	FCT0
Read/Write	-	-	-	-	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

Frame-End Interrupt-Counter Value

## G. Built-in Registers (cont.)

### DMAC (channel 1)

Register	Address	Remarks
----------	---------	---------

DMA interrupt enable register channel 1 (DIR channel 1)	0084H	
--	-------	--

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)	EOTE	-	-	-	-	-	-	-
Single-block transfer mode (single address)		EOME	BOFE	COFE	-	-	-	-
Chained-block transfer mode								
Read/Write	RW	RW	RW	RW	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0

**Transfer End Interrupt Enable**  
 0: Disable  
 1: Enable

**Counter Overflow Interrupt Enable**  
 - Chained-block transfer mode  
 0: Disable  
 1: Enable

**Frame Transfer End Interrupt Enable**  
 - Chained-block transfer mode  
 0: Disable  
 1: Enable

**Buffer Overflow/Underflow Interrupt Enable**  
 - Chained-block transfer mode  
 0: Disable  
 1: Enable

DMA command register channel 1 (DCR channel 1)	0085H	
---	-------	--

	7	6	5	4	3	2	1	0
Single-block transfer mode (dual address)								
Single-block transfer mode (single address)	-	-	-	-	-	-	CMD1	CMD0
Chained-block transfer mode								
Read/Write	-	-	-	-	-	-	W	W
Initial Value	-	-	-	-	-	-	-	-

**Command Specification**  
 01: Software abort  
 10: Frame-end interrupt-counter-clear  
 Others: Reserved

Command Name	Function
--------------	----------

Software abort (01H)	Initializes the corresponding DMAC channel (see figure 6-2). All DMAC registers maintain their previous value.
Frame-end interrupt-counter-clear (02H)	Clears the frame-end interrupt-counter (FCT) of the corresponding DMAC channel to 0H and the EOM bit in the DSR to 0.

Unused	0086H
--------	-------

Unused	0087H
--------	-------

Reserved	0088H
----------	-------

	00DFH



Our Standards Set Standards

Hitachi America, Ltd.  
Semiconductor and IC Division  
Hitachi Plaza  
2000 Sierra Point Parkway, Brisbane, CA 94005-1819  
1-415-589-8300

---